Desenvolupament de SW encastat per un microsatèl·lit

Daniel Ávila Gámez

Resum– El present informe descriu el procés de desenvolupament d'un software encastat amb l'objectiu de determinar la posició i la orientació d'un cubesat en temps real per executar-se a un microcontrolador i utilitzant una unitat de mesura inercial de baix cost. Per aconseguir-ho es crea un entorn de desenvolupament utilitzant el System Workbench, Matlab i Simulink que permet testejar la solució amb una aproximació hardware-in-the-loop.

Paraules clau– cubesat, IMU, SGP4, WMM, filtre de Kalman, hardware-in-the-loop

Abstract– The present report describes the process of real time embdedded software development for the purpose of determining the position and orientation of a cubesat to run on a microcontroller and using a low cost inertial measurement unit. To achieve this, a development environment is created using the System Workbench, Matlab and Simulink that will test the solution using the hardware-in-the-loop approach.

Keywords– cubesat, IMU, SGP4, WMM, Kalman filter, hardware-in-the-loop

1 Introducció

Un sistema encastat es composa normalment de sensors, actuadors i un microcontrolador on s'executa el software que relaciona les entrades obtingudes dels sensors amb les sortides cap els actuadors. El present informe descriu el desenvolupament d'un software encastat que s'executa en un microcontrolador per llegir i processar les dades d'una unitat de mesura inercia (IMU) de baix cost gràcies a la fabricació MEMS[12], consistent en la integració en un sol chip miniaturitzat de varis sensors.

Posteriorment s'integrarà en un projecte més gran, el C3SatP[19] on participa l'Institut d'Estudis Espacials de Catalunya (IEEC) i que consisteix en el disseny i fabricació d'un sistema de control per un un nanosatèl.lit[16], un cubesat. Aquests petits satèl·lits estan revolucionant el sector aeroespacial donant accessibilitat a empreses més modestes a l'espai, ja que mentre que el cost de posada en òrbita d'un satèl·lit tradicional ronda els 500 milions de dòlars, els cubesats no superen el milió, al tenir un tamany molt reduït, una vida útil més curta i el cost del transport a l'espai és compartit entre molts cubesats, que viatjan en el mateix coet, obrint el sector a multitud d'aplicacions civils noves. Davant aquesta predicció sorgeixen les primeres necessitats de disposar d'un sistema de control que gestioni l'accés als diferents components i faci una abstracció del hardware per facilitar el desenvolupament de software que implementi una solució de negoci sense necessitat de tenir un coneixement profund de l'arquitectura específica del cubesat. Aquesta definició ens porta a la mateixa situació viscuda amb el naixement dels primers sistemes operatius per computadors de propòsit general, que es va repetir posteriorment per els dispositius mòvils, i cobrir aquesta necessitat és l'objectiu del projecte C3SatP, que dissenya actualment el software d'un OBC (On Board Computer), l'equivalent al sistema operatiu i els drivers necesaris per cubesats, seguint l'estàndar de fabricació proposat per Jordi Puig-Suari i Robert Twiggs[21].

L'OBC és en sí mateix un sistema encastat amb un microcontrolador ARM Cortex-M4 dissenyat específicament per ser instal·lat a un cubesat i gestionar la resta de components. Es composa d'un sistema operatiu de temps real ja existent per sistemes encastats, com és el FreeRTOS[3], i s'afegeixen mòduls per gestionar la resta de sistemes com el de comunicació, d'energia o de navegació.

El projecte que ens ocupa es situa al sistema de navegació i consisteix en el desenvolupament d'un software capaç d'estimar en temps real la posició i la orientació del cubesat utilitzant els mínims recursos possibles tant en costos com en el nombre de dispositius utilitzats, ja que l'addició d'un

• E-mail de contacte: daniel.avilag@e-campus.uab.cat
• Mención realitzada: Enginyeria de Computadors
• Treball tutoritzat per: Màrius Montón Macián (Departament de Microelectrònica i Sistemes Electrònics)
• Curs 2019/20

nou dispositiu implica el consum de més energia i la multiplicació de la font de possibles errors en el sistema que no es podran reparar, atès que estarà orbitant a uns 500 km d'altitud a l'anomenada Low Earth Orbit (LEO), a una velocitat idealment constant d'uns 8 km/s, donant unes 15 voltes a la Terra. Idealment perquè sobre el cubesat actuarà la força de fricció de l'atmosfera encara present de manera tènue, que es traduirà en una lenta pèrdua d'altitud fins la desintegració final quan acabi el seu període d'operació.

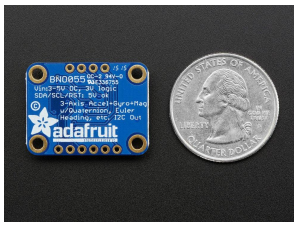

Fig. 1: Unitat de mesura inercial BNO055 de Bosch

Basat en aquesta premisa, s'estudiarà la viabilitat d'utilitzar una IMU de baix cost, el BNO055 de Bosch[14], que es presenta a la figura 1. Les IMUs incorporen un acceleròmetre, un giròscop i un magnetòmetre, tots ells amb 3 graus de llibertat (3DOF), oferint dades sobre l'acceleració, velocitat angular i camp magnètic, respectivament, per cadascun dels tres eixos que formen l'espai tridimensional; també poden incorporar un sensor de temperatura que pot ajudar a interpretar les dades provinents dels altres sensors quan aquestes presentin una variació amb la temperatura. Alguns integren un microcontrolador propi oferint la possibilitat d'obtenir dades dels sensors ja processades, o accedir a les dades en cru (raw data), o una combinació d'ambdues.

A les següents seccions s'analitza el problema a resoldre i es descriu com es dóna solució als mateixos partint de l'estat de l'art, per establir uns objectius basats en la elecció d'una solució en particular. A continuació s'explica la metodologia utilitzada per implementar la solució, el desenvolupament realitzat i s'analitzen els resultats dels experiments fets, per finalitzar amb les conclusions i el treball futur.

2 ANÀLISI DEL PROBLEMA

Per resoldre el problema de la determinació de la posició, la opció més intuïtiva és integrar doblement la mesura feta per l'acceleròmetre aplicant les lleis de Newton i coneixent la posició i velocitat inicials, propagar la dinàmica del sistema d'un instant al següent. És a dir, coneixent la posició, velocitat i acceleració a un instant determinat (estat del sistema), es pot calcular la posició i la velocitat a l'instant posterior. Però aquesta solució té molts problemes. Per una banda les condicions inicials són desconegudes, ja que tot el sistema està desconnectat fins que es llenci a l'òrbita i passi un temps de seguretat i no hi ha certesa de la posició una vegada arrenqui. Per altra banda, cada vegada que es fa una integració es comet un error. L'estat del sistema en un instant s'utilitza com a condició inicial per calcular l'estat a l'instant posterior, i per tant, si l'error té una deriva (la seva mitjana és diferent de zero), aquest s'acumularà en el temps.

Aquesta deriva en el temps es pot corregir utilitzant una segona font de mesura que sigui independent de les mesures en instants anteriors, i per tant, no tingui errors de deriva, com el GPS. GPS i acceleròmetre combinen molt bé per resoldre la posició ja que mentre el primer corregeix els errors de deriva, el segon corregeix els errors de precisió de l'altre. Addicionalment, amb un GPS es soluciona també la carència de condicions inicials. Si es fusionen aquestes dues fonts d'informació, per exemple, amb un filtre de Kalman[15], es pot aconseguir la millor estimació possible de la posició. Suposant que l'error en les dues mesures segueix una funció de densitat de probabilitat Gaussiana, aquestes es poden combinar per obtenir una altra funció Gaussiana amb una desviació estàndard més petita que les altres dos individualment, la millor estimació. Aquesta solució suposaria la incorporació d'un nou dispositiu al projecte, que és una de les decisions que volem evitar.

A més, altra font d'error prové de la resolució de l'acceleròmetre, la mínima acceleració que el sensor és capaç de mesurar. L'acceleròmetre del sensor utilitzat té una resolució de 14 bits amb un rang de +/- 2g. Si fem la divisió podem obtenir la sensibilitat, d'uns 250 μGs, mentre que l'acceleració deguda a la força de fricció de l'atmosfera a les òrbites LEO és de desenes de nano-Gs[24], quatre ordres de magnitud inferiors. Per tant es conclou que no es pot utilitzar aquest acceleròmetre per estimar la posició del cubesat.

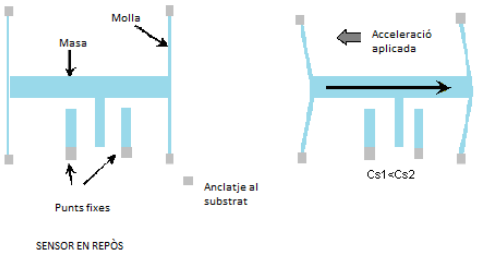

Fig. 2: Principi de funcionament de l'acceleròmetre

Mentre que la posició d'un cos rígid s'expressa com un vector en un sistema de coordenades, la orientació s'expressa com la relació existent entre dos sistemes de coordenades diferents. Al cos rígid s'annexa un nou sistema de coordenades BF (body-fixed) que rota amb ell i la orientació ve determinada pels angles existents entre els dos sistemes de coordenades; integrant la velocitat angular proporcionada per un giroscopi s'obtenen els angles, coneixent la posició angular inicial. En el cas trivial, quan el cos no ha rotat, els dos sistemes de coordenades estan superposats. A la figura 3 es pot visualitzar la idea, cada vector unitari del nou sistema de coordenades es pot expressar com a una combinació lineal dels vectors del sistema original, anomenat inercial. Transformant les equacions en una multiplicació matricial, es pot veure que la matriu de rotació és una matriu de canvi de base o matriu de cosinus directors (DCM). Aquesta matriu és de 3x3, i expressa una rotació a l'espai tridimensional, per tant, existeixen 6 restriccions en quant a la seva composició, per exemple, els elements de la matriu nomes prenen valors entre -1 i 1. Aquestes restriccions es tradueixen en equacions que completen el sistema d'equacions lineals per tal que quedi unívocament determinat. A més, gràcies a aquestes restriccions es pot calcular la inversa de

la matriu fent la transposada, ja que es tracta d'una matriu ortonormal, essent una operació molt més simple d'executar per un processador que el càlcul de la inversa.

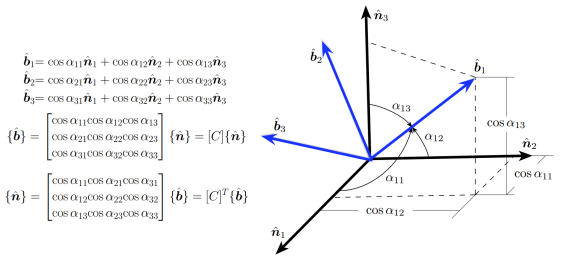

Fig. 3: Rotació entre dos sistemes de coordenades

La DCM és la mare de totes les parametritzacions d'orientació, i a partir d'ella s'han generat la resta de sistemes de descripció de la orientació, com el angles de Euler, el quaternions, el angles principals de rotació (PRV) o els paràmetres de Rodrigues, entre d'altres[25]. Cada manera d'expressar la rotació té la seva corresponent traducció a DCM i un nombre de components inferior a 9, generalment 3 o 4. Els que tenen 3 components, pateixen singularitats que donen lloc a la pèrdua d'un grau de llibertat quedant el sistema indeterminat per algunes orientacions en particular. S'utilitzen perquè existeixen aplicacions on no és necessari tenir els tres graus de llibertat; per exemple, els angles de Euler en una determinada configuració tenen una singularitat al segon eix als 90º (gimbal-lock[18]). Si s'utilitzen a un automòbil o un vaixell existiran problemes per determinar la orientació quan el cotxe o el vaixell estiguin orientats en direcció vertical mirant al cel, clar que quan es doni aquest cas existirà un problema major que determinar la orientació. És per aquest motiu que a l'espai s'utilitzen parametritzacions de 4 components, com els quaternions, perquè la nau sí pot estar orientada a qualsevol direcció; a canvi, en tenir 4 component s'assumeix només 1 restricció en comptes de les 6 de les DCM, i és un dels motius de la seva utilització per implementar software, perquè menys restriccions es tradueix en menys complexitat i cost computacional.

La IMU BNO055 ofereix directament els quaternions com a sortida processada ja que disposa d'un microcontrolador Arm Cortex-M0+ de 32 bits que proporciona un processament de les dades sensades. Aquests quaternions relacionen el sistema de coordenades fixat al sensor (BF) amb un sistema de coordenades local com el Nord-Est-Down (NED) que es mostra a la figura 4; és local perquè considera el pla tangent a la Terra a la posició del sensor i fixa les direccions de dos eixos (cap al nord i cap al centre de la Terra) completant la tercera la triada seguint la regla de la mà dreta. L'algoritme que utilitza la IMU per determinar la orientació és un filtre de Kalman que fusiona les dades de dos o tres sensors (es pot triar) per calcular els quaternions.

Però de nou existeix un problema si es vol aplicar a un cubesat. Per determinar la direcció dels eixos en coordenades NED per calcular els quaternions que el relacionen amb les coordenades BF, és necessari saber on es situa el Nord (magnetòmetre) i el centre de la Terra (acceleròmetre), i el problema torna a aparèixer amb l'acceleròmetre de la mateixa manera que per determinar la posició. A la figura 2 es mostra el principi de funcionament de l'acceleròmetre, que llegirà una acceleració de 9.8 m/s^2 (1 g) quan estigui en repòs a la Terra; d'aquesta manera es pot determinar la direcció del centre de la Terra. Però en el cubesat el sensor estarà en caiguda lliure i no es podrà determinar aquesta direcció, que conforma la condició inicial sobre la que fer la integració de la velocitat angular per obtenir els angles que descriuen la rotació. A més, la integració de la velocitat angular introdueix un error de deriva en la posició angular obtinguda.

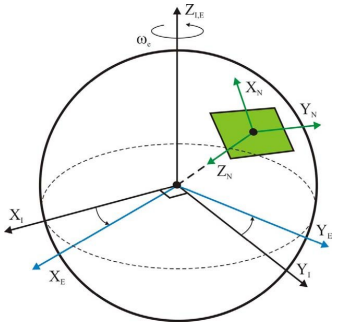

Fig. 4: Sistemes de coordenades ECI (negre), ECEF (blau) i NED (verd)

3 Estat de l'art

En moltes missions s'ha abordat el problema de la posició anomenat "orbit determination problem"sense necessitat de cap sensor. Coneixent la posició i la velocitat del cubesat en un instant determinat, es pot propagar la òrbita als instants posteriors si es té un model de les forces que actuen a la posició del cubesat. Per tant, es necessita una posició i velocitats inicials i el model per calcular les posicions i velocitats en instants posteriors.

La NORAD[5] (North American Aerospace Defense Command), proveeix de varis models que es poden fer servir per propagar la òrbita del satèl·lit[17]. A la universitat de Dundee, el dr. Paul Crawford va publicar a GitLab un codi d'exemple que implementa el model SGP4 per propagar la òrbita d'un satèl·lit[6]. L'algoritme de propagació d'òrbita és iteratiu, calcula la posició i velocitat actual en base a la anterior, i per tant, acumula un error que s'incrementa a mesura que la predicció s'allunya en el temps de les condicions inicials. Després de dues setmanes, l'error a la posició estimada superaria els 40 km. Així que les posicions i velocitats (condicions inicials) s'han d'anar actualitzant periòdicament.

Les condicions inicials no les proporciona l'algoritme, però la mateixa NORAD publica diàriament les dades de tots els satèl·lits que passen prop del seu camp de visió, a la web CelesTrak [1], en un format de dues línies i 80 columnes conegut com a TLE[8]. Quan s'estableix comunicació amb el cubesat, es pot enviar la informació de la seva posició i la velocitat periòdicament i d'aquesta manera el cubesat pot actualitzar les seves condicions inicials. Moltes missions de tot el món depenen de la NORAD per conèixer la posició dels seus satèl·lit, i molts es pregunten què passaria si deixés de compartir aquesta informació. Aquest fet ha motivat l'estudi d'alternatives com la utilització del projecte GENSO[22], una xarxa col·laborativa de compartició d'in-

[1] `https://www.celestrak.com/NORAD/elements/`

formació de localització de satèl·lits, per obtenir les dades necessàries per estimar la posició del satèl·lit[20].

El denominat "attitude determination problem"ha sigut object d'estudi intensiu durant els últims 60 anys. L'anomenat TRIAD Method proposat per Harold D. Black el 1964[13] calcula la orientació a partir de dos vectors coneguts en els dos sistemes de coordenades que es volen relacionar a la posició on es troba el sensor. Es necessita un mínim de dos vectors a cada sistema de referència (quatre en total) per determinar la orientació. Per exemple, si el magnetòmetre sensa el camp magnètic en només un dels eixos i rota sobre el mateix eix tants graus com es vulgui, la mesura no canviarà, serà la mateixa, per això és necessari un segon vector. Normalment s'utilitza un "sun sensor"per obtenir aquest segon vector que permet determinat unívocament la orientació, encara que existeixen més dispositius que posicionen objectes i poden utilitzar-se per aconseguir un dels dos vectors necessàris, com els "star trackers"o els "Earth sensors", inclús es poden utilitzar els panells solars per determinar la direcció del Sol deduint el vector de Pointing de la potència rebuda en els panells[11].

L'algoritme TRIAD es tracta d'un mètode determinista que no té en compte els errors en la mesura. Posteriorment van sorgir altres mètodes que sí tenen en compte el caràcter estocàstic del problema, com el QUEST Method, una descripció del problema i el mètode es pot trobar, per exemple, a la tesi de Henri Christian Kjellberg, de la universitat de Texas a Austin, al capítol 5[7].

De cada sensor que s'utlitzi per posicionar un objecte es necessita ubicar aquest mateix objecte en coordenades inercials. Per exemple, si es mesura el camp magnètic amb el magnetòmetre es pot determinar la direcció del vector en les coordenades BF del sensor; després, coneguda la posició actual del cubesat (SGP4 + TLE), es pot determinar aquest mateix camp en coordenades NED si es troba un model amb el que es pugui calcular el valor del camp magnètic donada una posició. Aquest model existeix, el World Magnetic Model[10] n'és un exemple encara que hi han d'altres.

Per obtenir el segon vector no es pot utilitzar el giroscopi, ja que no situa cap objecte conegut, només mesura la velocitat angular.

De fet, amb 2 vectors el sistema queda sobre-determinat, doncs per cada vector obtenim dues de les tres dimensions necessàries. Revisant la literatura, el 2011, en Jason David Searcy, a la seva tesi de màster per la Universitat de Ciència i Tecnologia Missouri, aprofita aquest fet per implementar un mètode que aconsegueix obtenir la orientació del cubesat utilitzant com a segon vector (que anomena pseudo-vector) la derivada del camp magnètic tant en coordenades inercials com en BF[26]. En aquesta tesi demostra que es pot obtenir els quaternions que descriuen la rotació del cubesat coneixent la seva posició, amb la qual calcula el camp magnètic i la seva derivada en coordenades inercials gràcies al World Magnètic Model; amb la mesura del magnetòmetre obté el camp magnètic en coordenades BF i calcula la seva derivada utilitzant un filtre de Kalman, per finalment, amb un segon filtre de Kalman calcular la orientació i la velocitat angular utilitzant com a mesura el resultat del primer filtre, aconseguint un error de 2 graus amb les dades d'una missió real, la M-SAT. El seu treball es basa en un treball anterior fet per Natanson i Challa[23], que intentaven obtenir la orientació d'un satèl·lit perdut del qual disposaven de les dades del magnetòmetre i les velocitats angulars. Aleshores van crear un mètode que van anomenar DADMOD (Deterministic Attitude Determination using Magnetometer-Only Data) on aconsegueixen determinar la orientació calculant la derivada fent diferències finites sobre les mesures del camp magnètic i coneixent la acceleració angular.

4 OBJECTIUS

L'objectiu principal del present projecte TFG és implementar un software en temps real que s'executi al microcontrolador STM32F446ZE[28] basat en un Arm Cortex-M4 de STMicroelectronics connectat a un sensor inercial de baix cost, model BNO055 de Bosch per resoldre els següents problemes:

- Determinació de la posició. Utilitzant l'algoritme de propagació basat en el model SGP4 i les efemèrides proporcionades per la NORAD.
- Determinació de la orientació. Utilitzant el magnetòmetre present a la IMU i l'algoritme basat en dos filtres de Kalman en cascada.

Com a subobjectiu cal destacar la creació d'un entorn de treball que permeti arribar a la solució de manera que es pugui validar i que aquest entorn sigui auto-contingut en el sentit que permeti donar continuïtat al desenvolupament del sistema de navegació per altres professionals que participin, una vegada el TFG finalitzi.

5 METODOLOGIA

La metodologia utilitzada ha estat la divisió en tasques que s'han anat completant iterativament i on la majoria implementen un petit projecte individual que unit a la resta creen l'entorn de treball on poder desenvolupar la solució i testejar-la de manera ràpida. Aquest entorn consisteix en un conjunt d'eines utilitzades per acabar fent un desenvolupament de tipus hardware-in-the-loop (HIL)[9], davant la impossibilitat de provar el sistema a l'espai.

A l'annex A.1 es llista tot el software utilitzar per composar l'entorn de treball i fer el desenvolupament de la solució i a l'annex A.2 es mostra el diagrama de Gannt de la planificació del projecte.

5.1 Entorn de treball

Per programar el microcontrolador s'ha utilitzat el System Workbench for STM32[29], que és un IDE creat per l'empresa francesa AC6 basat en Eclipse que proporciona eines visuals per activar i configurar els diferents mòduls hardware presents a la placa de desenvolupament NUCLEO-F446ZE[27] que es mostra a la figura 5, subministrada per l'ICE per fer el TFG. Aquesta placa de desenvolupament incorpora, a més del microcontrolador STM32F446ZE, mòduls per poder introduir software que s'executi al microcontrolador, comunicació USART, butons, LEDs, rellotges, mòduls que permeten depurar el software, entre d'altres.

Per realitzar el disseny dels algoritmes i la validació dels resultat s'ha utilitzat Matlab i Simulink[1]. El departament

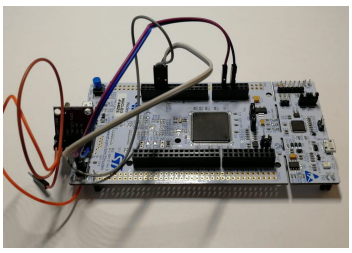

Fig. 5: Placa de desenvolupament NUCLEO-F446ZE connectada a la IMU BNO055

MathWorks Aerospace Products Team, ha desenvolupat un add-on, l'Aerospace Blockset Cubesat Simulation Library[2] per Simulink, que es mostra a la figura 6 que precisament simula l'evolució d'un cubesat donats uns paràmetres inicials (data, posició, rotació, velocitat lineal i velocitat angular inicials, entre d'altres), i permet visualitzar el cubesat orbitant al voltant de la Terra, que no només està dibuixada sinó que també està modelada, així com el sol i la lluna, fent els moviments de rotació segons la data.

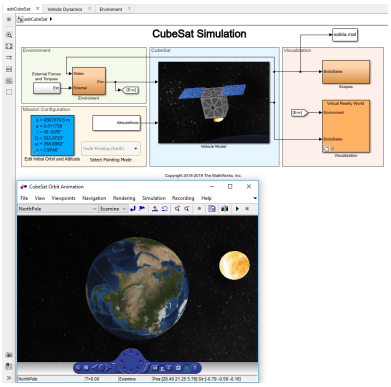

Fig. 6: Aerospace blockset cubesat simulation library

Les primeres tasques han estat orientades a la familiarització amb el microcontrolador i el sensor, realitzant petits projectes per configurar el sensor, llegir les dades a través de I2C[4] i comunicar-les a través del port sèrie per poder llegir-les i analitzar-les des de Matlab. A la figura 7 es mostra un diagrama de Simulink que processa les dades rebudes del sensor i simula gràficament el moviment de translació i rotació de la IMU en temps real, aprofitant el model del cubesat del projecte del add-on instal·lat. El model visual és de gran ajuda per verificar la correcta alineació dels eixos de coordenades, ja que amb un simple moviment del sensor es pot veure la resposta a la simulació, sense haver d'interpretar un conjunt de dades o una gràfica representant l'evolució dels quaternions.

Una vegada l'entorn està creat, s'ha procedit a buscar una font de dades d'una missió d'un satèl·lit real que contingui les posicions, orientacions i el camp magnètic sensat per poder validar els algoritmes desenvolupats. La Agència Espacial Europea (ESA) posa a disposició del públic les dades d'algunes de les missions. S'ha utilitzat la missió SWARM[3] per fer les proves del sistema. Es proporcionen dades de les

[2] https://es.mathworks.com/matlabcentral/fileexchange/70030-aerospace-blockset/cubesat/-simulation-library

[3] https://earth.esa.int/web/guest/missions/esa-operational-eo-missions/swarm

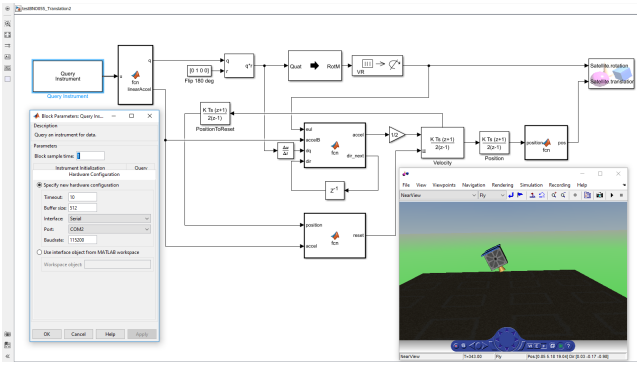

Fig. 7: Comunicació IMU -> Microcontrolador -> Simulink

lectures de tots els dispositius presents al satèl·lit, un fitxer per cada dia, i les posicions i rotacions amb un període de mostreig d'un segon. Els arxius es proporcionen en format CDF, de Wolfram Research. Com que Matlab no té support per aquest tipus d'arxius s'ha escrit un script en python aprofitat que existeix una llibreria[4] per llegir aquests fitxers per traduir el fitxer a format csv i d'aquesta manera utilitzar-lo en Matlab.

6 Desenvolupament

6.1 Estimar la posició

Per estimar la posició s'ha descarregat l'algoritme de la NORAD en C i s'ha adaptat per executar-se al microcontrolador. Aquesta adaptació ha consistit en la substitució de les crides del sistema d'entrada i sortida que estaven escrites per una arquitectura x86 per les corresponents a les mateixes en FreeRTOS. Posteriorment s'ha executat partint d'un arxiu TLE descarregat de la web de Celestrack que situa la posició i velocitat d'un satèl·lit real. Les dades sobre la posició generades s'han transmès a través del port sèrie i s'ha alimentat la simulació gràfica de Simulink, comprovant que descriuen una òrbita estable. A la figura 8 es pot observar el model que llegeix les posicions generades per l'algoritme SGP4 executant-se en temps real al microcontrolador a través del port sèrie i les utilitza per situar el cubesat a la simulació gràfica a la part inferior dreta. A l'esquerra es grafica l'evolució de les components del vector posició i es pot comprovar que la òrbita es estable.

6.2 Estimar la orientació

Per estimar la orientació es necessiten 2 vectors en coordenades BF i els mateixos vectors en coordenades ECEF (Earth-Centered Earth-Fixed), que es tracta d'un sistema de referència amb origen al centre de la Terra però que rota amb ella. El primer vector és el camp magnètic i el segon la seva derivada. Partint del camp magnètic mesurat amb el magnetòmetre en BF, s'implementa un primer filtre de Kalman per calcular la seva derivada. Per altra banda, a partir de la data i la posició es calcula el mateix camp magnètic i la seva derivada en coordenades ECEF. Finalment, amb els quatre vectors s'alimenta el segon filtre de Kalman que estima la orientació i la velocitat angular, sense necessitat

[4] https://github.com/MAVENSDC/cdflib

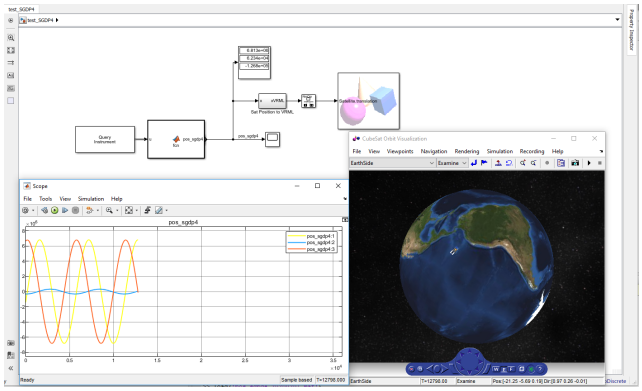

Fig. 8: Posicions generades a partir d'un fitxer TLE amb l'algoritme SGP4 executant-se en al microcontrolador

d'utilitzar el giroscopi. A la figura 9 es mostra el model simulink generat, composat per subsistemes en una estructura jeràrquica que es pot seguir a la barra superior a mesura que es va entrant a cada subsistema. Cada subsistema s'ha nombrat seguint la numeració de les equacions implementades seguint la tesi de màster referenciada a la secció 3 per facilitar el seu seguiment, així com les configuracions dels filtres.

Per implementar un filtre de Kalman primer s'ha de definir el problema amb una equació d'estat que descrigui el comportament dinàmic del sistema, així com una equació de mesura que descrigui la relació entre variables d'estat i la sortida dels sensors. Les equacions han de ser descrites en temps discret. A més, s'han de conèixer les propietats estadístiques del soroll del sistema (l'error degut a pertorbacions no capturades per el model del comportament dinàmic del sistema) i del soroll de mesura, del sensor. El soroll ha de ser Gaussià, sinó el filtre no aplica. El sistema ha de ser lineal, i en el cas de no ser-ho, es pot utilitzar la versió extesa (EKF) que a cada pas linealitza utilitzant l'aproximació en sèrie de Taylor.

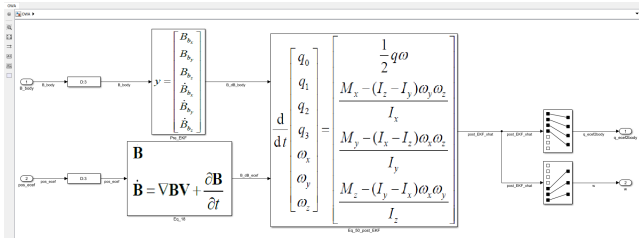

Fig. 9: Model que estima la velocitat angular i la orientació

6.2.1 Pre-filtre

L'objectiu del pre-filtre és calcular la derivada del camp magnètic mesurat amb el magnetòmetre, per introduir la mesura del camp i la derivada com a entrada del segon filtre. Rep com a entrada el camp magnètic mesurat i a la sortida genera la derivada utilitzant un procés de Markov, suposant que la tercera derivada és igual a zero. Simulink proporciona un block que implementa el filtre de Kalman. A la figura 10 es pot veure la configuració del pre-filtre, i als llistats 1 i 2 les funcions de transició d'estats i de mesura, respectivament.

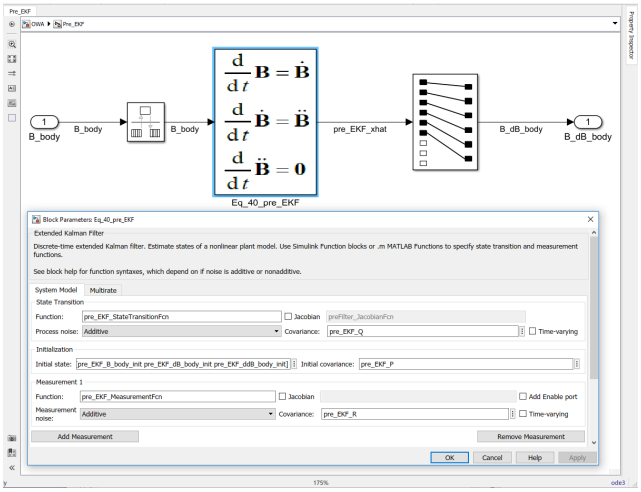

Fig. 10: Model simulink del pre-filtre

Listing 1: Funció de transició d'estat - pre_EKF_StateTransitionFcn

```
function x = pre_EKF_StateTransitionFcn(x)
F = [0 0 0 1 0 0 0 0 0;  ...
     0 0 0 0 1 0 0 0 0;  ...
     0 0 0 0 0 1 0 0 0;  ...
     0 0 0 0 0 0 1 0 0;  ...
     0 0 0 0 0 0 0 1 0;  ...
     0 0 0 0 0 0 0 0 1;  ...
     0 0 0 0 0 0 0 0 0;  ...
     0 0 0 0 0 0 0 0 0;  ...
     0 0 0 0 0 0 0 0 0;  ...
    ];
x = x + F*x;
end
```

Listing 2: Funció de Mesura - pre_EKF_MeasurementFcn

```
function y = pre_EKF_MeasurementFcn(x)
H = [1 0 0 0 0 0 0 0 0;  ...
     0 1 0 0 0 0 0 0 0;  ...
     0 0 1 0 0 0 0 0 0;  ...
    ];
y = H*x;
end
```

Al llistat 3 es mostra la inicialització del filtre, condicions inicials i matrius de covariància.

Listing 3: Script d'inicialització

```
% Pre_filtre
pre_EKF_Q = diag(repmat(1e-1,1,9));
pre_EKF_R = diag(repmat(1e-9,1,3));
pre_EKF_P = diag([repmat(1e4,1,3) repmat(1e6,1,6)]);


% Condicions inicials, derivada de les 3 primeres mostres
dB_body_2 = [    (B_CRF.Data(3,1)-B_CRF.Data(1,1))    ...
                 (B_CRF.Data(3,2)-B_CRF.Data(1,2))    ...
                 (B_CRF.Data(3,3)-B_CRF.Data(1,3))] * 0.5;

dB_body_3 = [    (B_CRF.Data(4,1)-B_CRF.Data(2,1))    ...
                 (B_CRF.Data(4,2)-B_CRF.Data(2,2))    ...
                 (B_CRF.Data(4,3)-B_CRF.Data(2,3))] * 0.5;

dB_body_4 = [    (B_CRF.Data(5,1)-B_CRF.Data(3,1))    ...
                 (B_CRF.Data(5,2)-B_CRF.Data(3,2))    ...
                 (B_CRF.Data(5,3)-B_CRF.Data(3,3))] * 0.5;

ddB_body_3 = [   (dB_body_4(1)-dB_body_2(1))      ...
                 (dB_body_4(2)-dB_body_2(2))      ...
                 (dB_body_4(3)-dB_body_2(3))] * 0.5;

pre_EKF_B_body_init = B_CRF.Data(3,:);
pre_EKF_dB_body_init = dB_body_3;
pre_EKF_ddB_body_init = ddB_body_3;
```

6.2.2 World Magnetic Model

Abans d'implementar el segon filtre és necessari generar el vector de camp magnètic i la seva derivada en coordenades ECEF. Partint d'una data i una posició en coordenades geodèsiques (altitud, latitud i longitud)[2], el World Magnetic Model calcula el camp magnètic per aquella data i posició. Posteriorment es calcula la derivada aplicant diferències finites en el temps i l'espai, ja que el camp magnètic varia en ambdues dimensions. A la figura 11 es mostra el subsistema que s'encarrega de fer els càlculs.

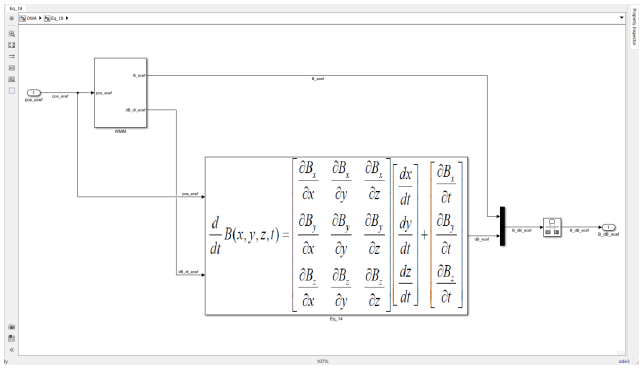

Fig. 11: WMM. Càlcul del camp magnètic i la seva derivada temporal i espacial

A la figura 12 es mostra el subsistema simulink que rep una posició en coordenades ECEF, la transforma a coordenades geodèsiques per alimentar el WMM que genera el camp magnètic en coordenades NED per una data configurada. Aquest camp es transforma a coordenades ECEF multiplicant per la DCM corresponent. A continuació s'alimenta el subsistema que calcula la derivada temporal, que es mostra obert a la part inferior de la figura, on s'aplica un retard de dues mostres per aplicar la diferenciació central.

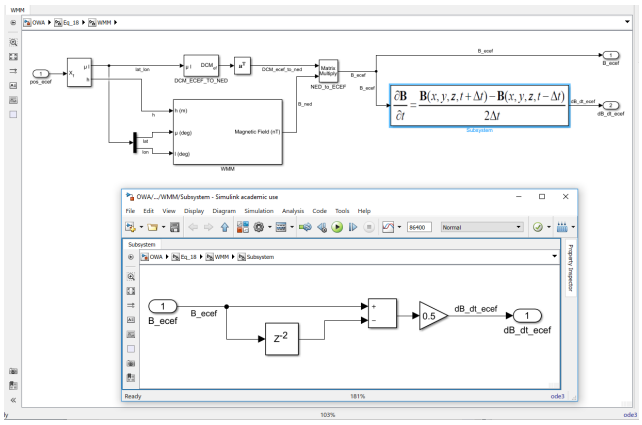

Fig. 12: WMM. Càlcul del camp magnètic i la seva derivada temporal

Per calcular la derivada espacial es calcula de nou el camp magnètic augmentant i disminuint una unitat cadascuna de les components del vector posició en coordenades ECEF i aplicant de nou la diferenciació central, com es mostra a la figura 13.

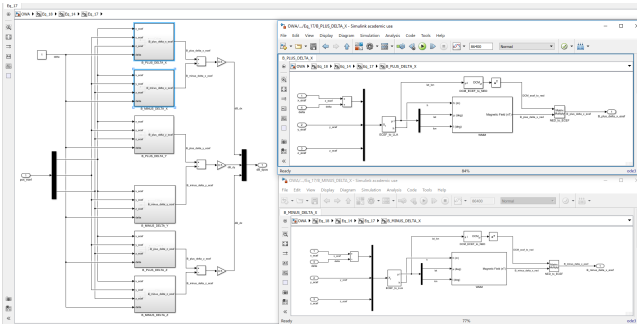

Fig. 13: WMM. Càlcul de la derivada espacial del camp magnètic.

6.2.3 Post-filtre

Una vegada tenim els vectors de camp magnètic i les seves derivades en coordenades BF i ECEF es procedeix a implementar el segon filtre de Kalman de la mateixa manera que s'ha fet amb el primer. Les funcions de transició d'estat i mesura es mostren als llistats 4 i 5, respectivament, on es pot veure a la funció de transició d'estats com s'aplica la restricció normalitzant el quaternió fent que el sistema quedi determinat.

Listing 4: Funció de transició d'estat - post_EKF_StateTransitionFcn

```
function x = post_EKF_StateTransitionFcn(x)
q = quatnormalize([x(1) x(2) x(3) x(4)]);
Wx=x(5); Wy=x(6); Wz=x(7);
Ix = 0.8918222; Iy = 0.8753646; Iz = 0.6176641;

F = 0.5* ...
[0  -Wx -Wy -Wz -q(2) -q(3) -q(4);  ...
 Wx  0   Wz -Wy  q(1) -q(4)  q(3);  ...
 Wy -Wz  0   Wx  q(4)  q(1) -q(2);  ...
 Wz  Wy -Wx  0  -q(3)  q(2)  q(1);  ...
 0   0   0   0   0  2*(Iy-Iz)*Wz/Ix   2*(Iy-Iz)*Wy/Ix;   ...
 0   0   0   0   2*(Iz-Ix)*Wz/Iy   0   2*(Iz-Ix)*Wx/Iy;  ...
 0   0   0   0   2*(Ix-Iy)*Wy/Iz  2*(Ix-Iy)*Wx/Iz   0;   ...
];

x1 = x + F*x;
x = [quatnormalize([x1(1) x1(2) x1(3) x1(4)]) ...
     x1(5) x1(6) x1(7)]';
end
```

Listing 5: Funció de Mesura - post_EKF_MeasurementFcn

```
function y = post_EKF_MeasurementFcn(x, B_dB_ecef)
q = quatnormalize([x(1) x(2) x(3) x(4)]);
w = [0 x(5) x(6) x(7)];

q_c = quatconj(q);

Q = [   q(1) -q(2) -q(3) -q(4);  ...
        q(2)  q(1) -q(4)  q(3);  ...
        q(3)  q(4)  q(1) -q(2);  ...
        q(4) -q(3)  q(2)  q(1)];
dq = (0.5*Q*w')';
dq_c = quatconj(dq);

Bi = [0 B_dB_ecef(1) B_dB_ecef(2) B_dB_ecef(3)];
dBi = [0 B_dB_ecef(4) B_dB_ecef(5) B_dB_ecef(6)];

y1 = quatmultiply(quatmultiply(q_c, Bi), q);
y2 = quatmultiply(quatmultiply(dq_c, Bi), q) + ...
     quatmultiply(quatmultiply(q_c, dBi), q) + ...
     quatmultiply(quatmultiply(q_c, Bi), dq);

y = [y1(2) y1(3) y1(4) y2(2) y2(3) y2(4)];

end
```

Mentre que la funció de transició d'estat segueix fidelment la tesi, la funció de mesura s'ha implementat de manera diferent, ja que amb les instruccions de la tesi el sistema

donava un error a la orientació molt gran, encara que les dues implementacions deurien d'ésser equivalents. L'autor no dóna explícitament la matriu de mesura H com al pre-filtre però indica que partint de l'equació 2 i utilitzant l'equació 1, substituir i utilitzar el resultat per implementar la funció de mesura. Al llistat 6 es mostra aquesta implementació que possiblement tingui un error de concepte que no s'ha pogut determinar. En canvi, la present implementació s'ha fet calculant la derivada del quaternió prèviament utilitzant l'equació 4 que relaciona la derivada del quaternió amb la velocitat angular i el propi quaternió sense derivar, extreta del llibre Analytical Mechanics of Space Systems, de Hanspeter Schaub i John L. Junkins[25].

$$\begin{bmatrix} B_{bf} \\ B_{bf} \end{bmatrix} = \begin{bmatrix} q^c B_{ecef} q \\ \dot{q^c} B_{ecef} q + q^c \dot{B}_{ecef} q + q^c B_{ecef} \dot{q} \end{bmatrix} \quad (1)$$

$$\dot{q} = \frac{1}{2} q w \quad (2)$$

$$\begin{bmatrix} B_{bf} \\ B_{bf} \end{bmatrix} = \begin{bmatrix} q^c B_{ecef} q \\ \frac{1}{2} q^c w B_{ecef} q + q^c \dot{B}_{ecef} q + q^c B_{ecef} \frac{1}{2} q w \end{bmatrix} \quad (3)$$

$$\dot{q} = \frac{1}{2} \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{bmatrix} \begin{bmatrix} 0 \\ w_1 \\ w_2 \\ w_3 \end{bmatrix} \quad (4)$$

Listing 6: Funció de Mesura implementant l'equació 3

```
function y = post_EKF_MeasurementFcn(x, B_dB_ecef)
q = quatnormalize([x(1) x(2) x(3) x(4)]);
w = [0 x(5) x(6) x(7)];

q_c = quatconj(q);

Bi = [0 B_dB_ecef(1) B_dB_ecef(2) B_dB_ecef(3)];
dBi = [0 B_dB_ecef(4) B_dB_ecef(5) B_dB_ecef(6)];

y1 = quatmultiply(quatmultiply(q_c, Bi), q);
y2_1 = quatmultiply( ...
        quatmultiply( ...
            quatmultiply(0.5*q_c, w), Bi), q);
y2_2 = quatmultiply(quatmultiply(q_c, dBi), q);
y2_3 = quatmultiply( ...
        quatmultiply( ...
            quatmultiply(q_c, Bi)*0.5, q), w);
y2 = y2_1 + y2_2 + y2_3;

y = [y1(2) y1(3) y1(4) y2(2) y2(3) y2(4)];

end
```

6.2.4 Traducció a C

Tots els blocs que han estat utilitzats a simulink per implementar la solució es poden traduir automàticament a codi C per executar-se a un microcontrolador ARM, amb la instal·lació de l'add-on Embedded Coder de Simulink. Una vegada generat el codi es pot visualitzar i si s'ha activat l'opció corresponent es generen comentaris davant de cada bloc que actuen com a link mostrant el bloc al qual es refereix la traducció, com es pot veure a la figura 14.

A la taula 1 es recull el tamany en KB de les dos solucions, separades per l'espai que ocupen a la memòria Flash, de la que el microcontrolador disposa de 512 KB i l'espai

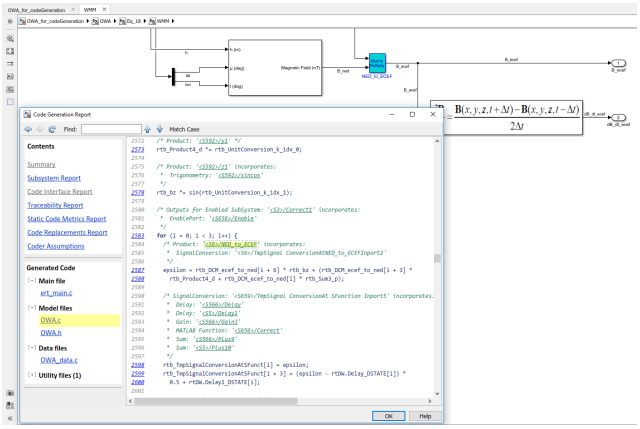

Fig. 14: Codi C generat de la solució.

Algoritme	Flash	SRAM	Total Space	Time Step (ms)
SGP4	60 KB	5 KB	65 KB	0.4 ms
OWA	75 KB	75 KB	150 KB	105 ms
Total	135 KB	80 KB	215 KB	105.4 ms

TAULA 1: TAMANY DEL CODI COMPILAT PER EL MICROCONTROLADOR I MS/STEP DE CADA ALGORITME.

que ocupa a la memòria SRAM, de la que el microcontrolador disposa de 128 KB. També s'inclou el temps que triga cada iteració en executar-se, un total de 105.4 ms. El període de mostreig és d'un segon, i es necessita tenir el resultat de la iteració abans d'aquest temps.

6.3 Test

Abans de provar la validesa de la solució a l'hora d'obtenir una bona orientació s'han realitzat tests a blocs individuals per comprovar la seva correcta implementació en la mesura del possible, atès que no es disposa d'un ground-truth de totes les dades per fer les comprovacions. A la figura 15 es mostra un exemple, per comprovar la correcta implementació del subsistema que calcula el camp magnètic en coordenades ECEF mitjançant el WMM, s'han introduït la data i les posicions de la missió SWARM, de la que es disposa el camp magnètic i s'ha comparat amb el generat per el model, comprovant que l'error relatiu mitjà és inferior al 2%.

S'ha comprovat que el resultat de la implementació feta al microcontrolador en codi C sigui el mateix que a les simulacions. Per fer-ho s'ha creat un entorn de test bastat en el paradigma hardware-in-the-loop, consistent en un projecte de test que s'executa al microcontrolador i espera rebre les dades sobre el camp mesurat des del terminal en comptes de la IMU, les dades son enviades des de Matlab, que a la seva vegada espera rebre la resposta per avaluar-la, per el mateix terminal.

7 RESULTATS

Per avaluar la bondat de les solucions s'han fet simulacions amb les dades reals obtingudes de la missió SWARM i també s'han generat òrbites amb paràmetres diferents aprofitant el projecte inclòs a l'Aerospace Blockset Cubesat Si-

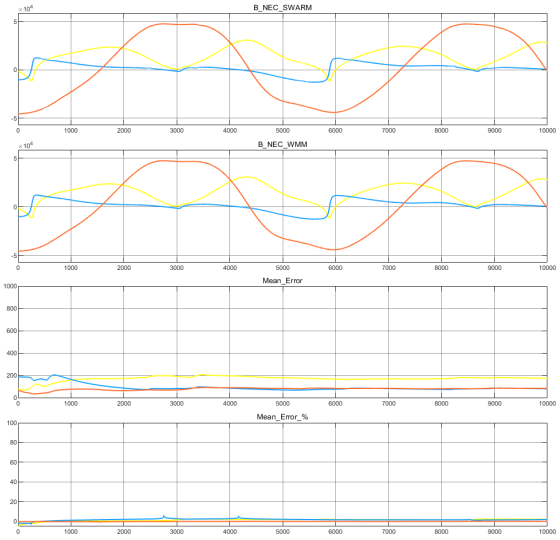

Fig. 15: Comparació entre el camp magnètic de la missió SWARM i el generat amb el WMM durant 10000 segons

mulation Library per provar com es comportaria el software davant canvis als paràmetres de la òrbita o del cubesat, per exemple, amb una velocitat angular o una inclinació de la òrbita diferents.

7.1 Òrbites simulades

Test 1

- Òrbita: Polar, inclinació 85 graus
- Orientació: lliure
- Velocitat angular inicial (Wx,Wy,Wz): 0 graus/s

A la figura 16 es pot veure el resultat. L'algoritme convergeix a un error mitjà per sota dels 20 graus després de 20 minuts i al cap de 4 hores baixa dels 10 graus per situar-se als 7.5 graus a les 10 hores, encara que existeixen 3 períodes d'uns 15 minuts on l'error instantani es dispara.

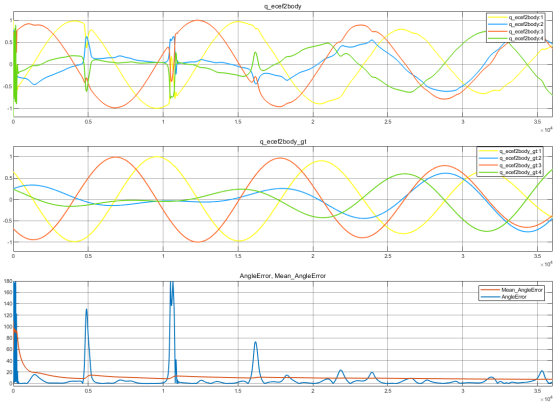

Fig. 16: Òrbita simulada. Comparació dels quaternions i error d'orientació en graus.

Test 2

- Òrbita: Polar, inclinació 85 graus
- Orientació: lliure
- Velocitat angular (Wx,Wy,Wz): (0.1,0.5,0.1) graus/s

A la figura 17 es pot veure el resultat. L'algoritme convergeix a un error mitjà per sota dels 20 graus després de 6 minuts i al cap de 40 minuts baixa dels 4 graus i s'estaciona fins que finalitza la simulació, a les 10 hores. Seria el resultat ideal.

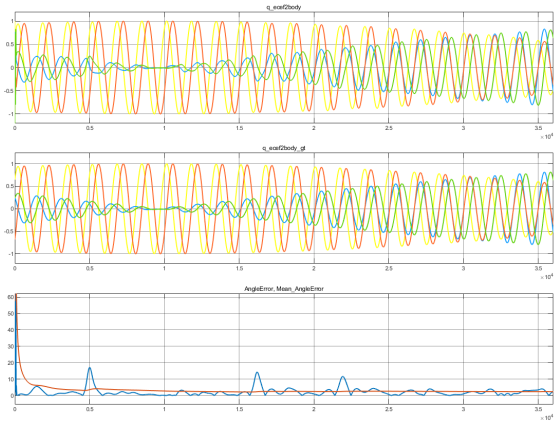

Fig. 17: Òrbita simulada. Comparació dels quaternions i error d'orientació en graus.

Test 3

- Òrbita: Polar, inclinació 85 graus
- Orientació: cap a la Terra (Nadir)
- Velocitat angular (Wx,Wy,Wz): (0.1,0.5,0.1) graus/s

A la figura 18 es pot veure el resultat. L'algoritme convergeix a un error mitjà per sota dels 20 graus després de 8 minuts i a partir de les 3 hores es situa per sota dels 10 graus fins als 7 graus al final de la simulació. S'observa que l'error instantani oscil·la al voltant del zero i arriba a màxims de 30 graus, la qual cosa no és desitjable. S'observen 4 períodes d'uns 15 minuts on l'algoritme divergeix donant un gran error a la orientació.

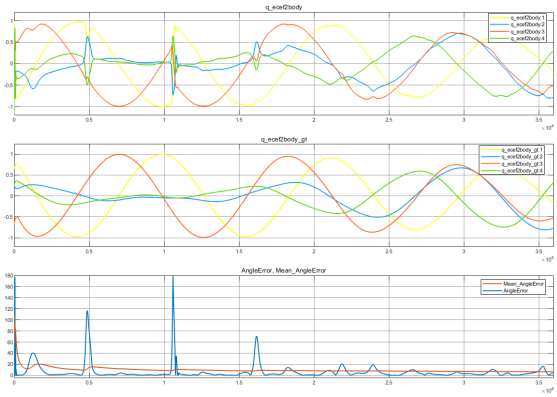

Fig. 18: Òrbita simulada. Comparació dels quaternions i error d'orientació en graus.

Test 4

- Òrbita: Polar, inclinació 85 graus
- Orientació: cap al sol
- Velocitat angular (Wx,Wy,Wz): (0.1,0.5,0.1) graus/s

A la figura 19 es pot veure el resultat. Com el cubesat està orientat al Sol, es pot observar com els quaternions varien molt lentament ja que la velocitat angular és pràcticament nul·la. L'algoritme convergeix a un error mitjà per sota dels 20 graus després de 9 minuts i al cap de 2 hores baixa dels 4 graus i s'estaciona fins que finalitza la simulació, a les 10 hores. No s'observa cap anomalia en l'error instantani. Seria el comportament ideal.

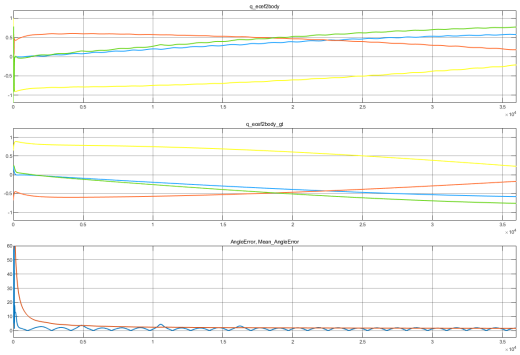

Fig. 19: Òrbita simulada. Comparació dels quaternions i error d'orientació en graus.

7.2 Missió SWARM

S'ha realitzat la simulació de la òrbita del satèl·lit A de la missió SWARM durant el dia 1 de gener de 2016 a una freqüència d'1 Hz, de la qual es tenen els quaternions mesurats a bord que descriuen la rotació des de les coordenades BF a ECEF, així com les posicions a cada instant i el camp magnètic en coordenades NED. S'ha transformat el camp magnètic a coordenades ECEF per aplicar la rotació per mitjà dels quaternions i generar el camp magnètic en coordenades BF, el que hauria de llegir el sensor. Posteriorment a aquest camp s'ha afegit soroll blanc Gaussià de magnitud 1.4 muT, extret del datasheet del sensor BNO055, per simular el sensor. A continuació s'ha fet la simulació, a la figura 20 es mostra el resultat per una simulació de 10 hores. S'observa com l'algoritme convergeix lentament i triga una hora en obtenir un error per sota de 20 graus; l'error instantani oscil·la al voltant del zero i arriba a màxims de 30 graus, la qual cosa no és desitjable. S'han provat diverses configuracions en les condicions inicials i no s'ha aconseguit una millora significativa. A la tesi seguida per implementar l'algoritme s'explica que l'algoritme pot arribar a obtenir un error de 2 graus però en determinades condicions, especialment en quant a la velocitat angular que és un dels paràmetres crítics.

- Òrbita: Polar, inclinació 87.35 graus
- Orientació: Cap a la Terra (Nadir)
- Velocitat angular (Wx,Wy,Wz): (0.01,-0.05,0.01) graus/s (approx)

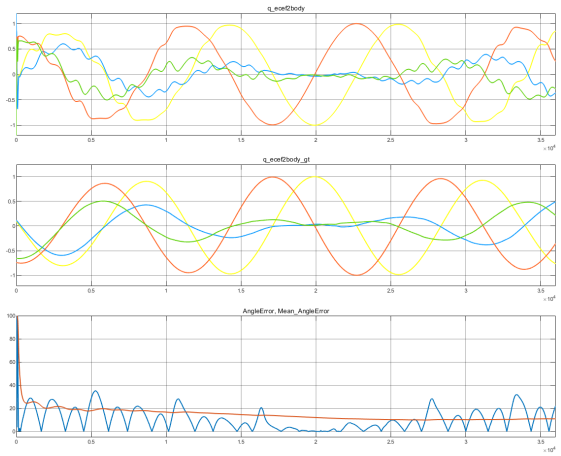

Fig. 20: Missió SWARM. Comparació dels quaternions i error d'orientació en graus.

8 Conclusions i treball futur

S'ha creat l'entorn necessari per poder desenvolupar el sistema de navegació del projecte C3SatP utilitzant l'aproximació hardware-in-the-loop. També s'ha pogut verificar positivament la viabilitat d'estimar la posició i la orientació d'un cubesat en òrbita utilitzant components de baix cost amb una precisió raonable front els costos dels dispositius utilitzats tradicionalment a la indústria aeroespacial destinada només a entitats governamentals amb un cost molt elevat, encara que han quedat proves per realitzar, sobretot en quant al tunning dels paràmetres per situar els límits d'estabilitat del sistema i saber quan aplica i quan no, i donar resposta a per què algunes de les proves no donen el resultat desitjat. També s'han de trobar aquests límits en quant a l'execució en un entorn compartit amb molt components.

Com a línia futura en el desenvolupament del sistema de navegació s'haurien de completar les següents tasques:

- Aplicar tècniques d'enginyeria de rendiment per optimitzar al màxim el codi generat mantenint la integritat del mateix.
- Estudiar l'impacte que l'error en la determinació de la posició fet amb SGP4+TLE té en l'algoritme de determinació de la orientació, ja que la sortida del primer és una de les entrada del segon.
- Creació d'un entorn per poder tunnejar l'algoritme de determinació de la orientació ad-hoc per una missió en concret.
- Estudiar les possibilitats d'implementació dels actuadors necessàris per poder provocar canvis en la orientació del cubesat.

Per altra banda, com a resultat secundari, s'ha pogut comprovar la importància d'estar constantment actualitzat en les noves eines i tecnologies de desenvolupament que permeten aconseguir fer implementacions de sistemes cada cop més complexos per professionals cada vegada més allunyats del camp d'especialització, enriquint cada vegada més la professió d'Enginyer Informàtic.

AGRAÏMENTS

Agrair als meus tutors Màrius Montón i Lluis Gesa per donar-me la oportunitat de contribuir en un projecte tan fascinant com aquest, i per deixar-me prendre decisions i donar-me ànims constantment. A Ramón Vilanova per fer-me un lloc a la seva apretada agenda i revisar el treball per donar-me la seguretat d'estar en el camí correcte. A la meva parella, per encertar quan deia que ho aconseguiria. I a Jason David Searcy i Henry J. Pernicka, per una tesi tan ben explicada.

REFERÈNCIES

[1] MathWorks. URL https://es.mathworks.com/.

[2] Earth-Fixed Coordinate Systems, . URL http://www.aerojockey.com/papers/meng/node7.html.

[3] FreeRTOS Kernel, . URL https://www.st.com/en/embedded-software/freertos-kernel.html.

[4] I2c Info – I2c Bus, Interface and Protocol, . URL https://i2c.info/.

[5] North American Aerospace Defense Command (NORAD), . URL https://www.norad.mil/.

[6] Paul Crawford / dundee_sgp4, . URL https://gitlab.com/pscrawford/dundee_sgp4.

[7] (PDF) Design of a CubeSat Guidance, Navigation, and Control Module, . URL https://www.researchgate.net/publication/273134690_Design_of_a_CubeSat_Guidance_Navigation_and_Control_Module.

[8] Two-line element (TLE) format - Tracker - satellite tracking software for Windows, . URL http://www.stltracker.com/resources/tle.

[9] What Is Hardware-in-the-Loop? - National Instruments, . URL https://www.ni.com/en-us/innovations/white-papers/17/what-is-hardware-in-the-loop-.html.

[10] World Magnetic Model | NCEI, . URL https://www.ngdc.noaa.gov/geomag/WMM/.

[11] E. Babcock. CubeSat Attitude Determination via Kalman Filtering of Magnetometer and Solar Cell Data. 2011. URL https://pdfs.semanticscholar.org/a939/c790b74424bd7b060a85762bfb28739cdcf5.pdf.

[12] Rachid Bekhti. What's the difference between MEMS and IMUs?, November 2017. URL https://variense.com/blog/difference-between-mems-and-imus/.

[13] HAROLD D. BLACK. A passive system for determining the attitude of a satellite. *AIAA Journal*, May 2012. doi: 10.2514/3.2555. URL https://arc.aiaa.org/doi/abs/10.2514/3.2555.

[14] BOSCH. BNO055. URL https://www.bosch-sensortec.com/bst/products/all_products/bno055.

[15] Ramsey Faragher. Understanding the Basis of the Kalman Filter Via a Simple and Intuitive Derivation [Lecture Notes]. *IEEE Signal Processing Magazine*, 29(5):128–132, September 2012. ISSN 1558-0792. doi: 10.1109/MSP.2012.2203621. URL https://ieeexplore.ieee.org/document/6279585.

[16] Bannister Global. Guía básica de nanosatélites. URL https://alen.space/es/guia-basica-nanosatelites/.

[17] Felix R. Hoots and Ronald L. Roehrich. Models for Propagation of NORAD Element Sets:. Technical report, Defense Technical Information Center, Fort Belvoir, VA, December 1980. URL http://www.dtic.mil/docs/citations/ADA093554.

[18] David Hosier. Avoiding Gimbal Lock in a Trajectory Simulation. URL https://apps.dtic.mil/dtic/tr/fulltext/u2/1055301.pdf.

[19] Institut d'Estudis Espacials de Catalunya. C3SATP. URL http://www.ieec.cat/en/content/30/success-cases.

[20] Ivo Klinkert. Homepage of Ivo Klinkert - Satellite Orbit Determination with GENSO, Master Project Thesis. URL https://ivok.home.xs4all.nl/about/orbitdetermination.html.

[21] Wenschel Lan. CubeSat Design Specification. URL https://static1.squarespace.com/static/5418c831e4b0fa4ecac1bacd/t/56e9b62337013b6c063a655a/1458157095454/cds_rev13_final2.pdf.

[22] Kyle Leveque. Global Educational Network for Satellite Operations (GENSO). 2007. URL https://pdfs.semanticscholar.org/6ab8/54f62c757ee97471412f3a26b5f8405d4130.pdf.

[23] G. A. Natanson and M. S. Challa. Magnetometer-only attitude and rate determination for a gyro-less spacecraft. URL https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19950011131.pdf.

[24] Marcin D Pilinski. An Innovative Method for Measuring Drag on Small Satellites. URL https://digitalcommons.usu.edu/cgi/viewcontent.cgi?article=1311&context=smallsat.

[25] Hanspeter Schaub and John L. Junkins. *Analytical Mechanics of Space Systems, Fourth Edition*. American Institute of Aeronautics and Astronautics, Inc., Washington, DC, July 2018. ISBN 978-1-62410-521-0. doi: 10.2514/4.105210. URL https://arc.aiaa.org/doi/book/10.2514/4.105210.

[26] Jason D Searcy. Magnetometer-only attitude determination with application to the M-SAT mission. URL https://scholarsmine.mst.edu/cgi/viewcontent.cgi?article=7891&context=masters_theses.

[27] STMicroelectronics. NUCLEO-F446ZE, . URL https://www.st.com/en/evaluation-tools/nucleo-f446ze.html.

[28] STMicroelectronics. STM32F446ZE, . URL https://www.st.com/en/microcontrollers-microprocessors/stm32f446ze.html.

[29] STMicroelectronics. SW4STM32, . URL https://www.st.com/en/development-tools/sw4stm32.html.

APÈNDIX

A.1 Secció d'Apèndix

El software utilitzat en el projecte és el següent:

- Hardware
  - Placa de desenvolupament NUCLEO-F446ZE de STMElectronic
  - IMU BNO0055 de Bosch
- Programació del microcontrolador
  - System Workbench for STM32 V1.24.0
  - STM32CubeF4 v1.24.0
  - STM32CubeMX For Eclipse v5.5.0
- Simulacions i tests
  - Matlab R2019a
  - Simulink 9.3
  - Simulink Add-Onns
    - Aerospace Blockset. Proporciona blocs per simular vehicles a l'espai.
    - Simulink 3D Animation. Simulacions en un espai 3D.
    - Aerospace Blockset CubeSat Simulation Library. Proporciona blocs per modelar y simular un CubeSat en òrbita.
    - Robotics System Toolbox. Proporciona blocs per traduir sistemes de coordenades.
    - MATLAB Coder. Per traduir codi matlab a c/c++
    - Simulink Coder. Per generar codi c/c++ des de un model en simulink
    - Embedded Coder. Per generar codi ANSI/ISO C per executar en qualsevol microcontrolador.
  - Python p3.7 amb el paquet cdflib

A.2 Secció d'Apèndix

A la figura 21 es mostra el diagrama de Gantt una vegada el projecte ha quedat finalitzat.

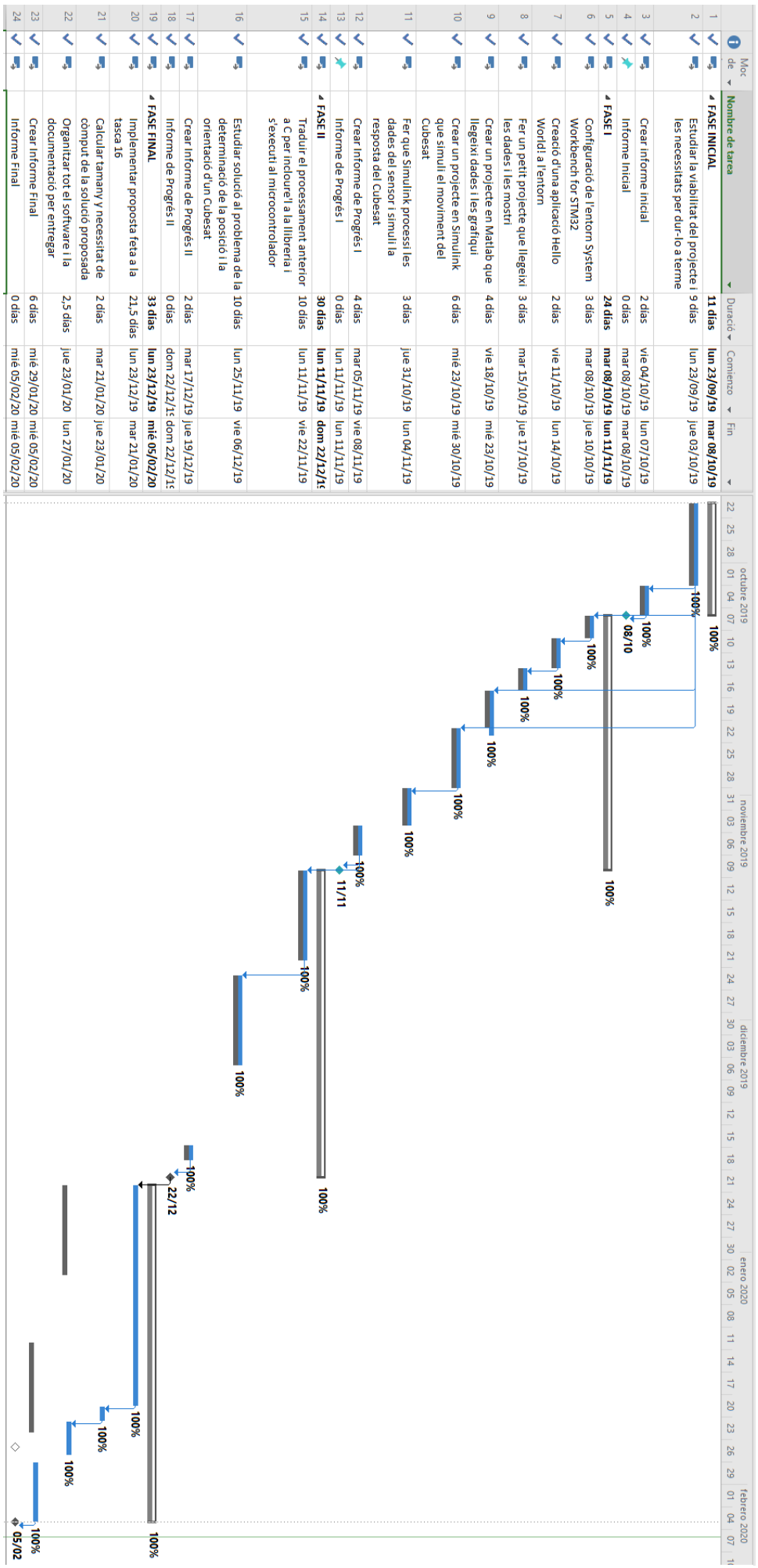

Fig. 21: Diagrama Gantt del projecte finalitzat