Marc Arbones Clua

Metodologies per al seguiment de trajectories de robots mobils

TREBALL DE FI DE GRAU

Dirigit Albert Oller Pujol

Grau d'Enginyeria Electrònica Industrial i Automàtica

UNIVERSITAT ROVIRA I VIRGILI

Tarragona

2015

1. Índex

Taula de figures

2. Introducció

En aquest treball farem una exploració a totes les opcions que tenim quan volem realitzar el seguiment de la posició d'un mòbil. Estudiarem totes aquestes possibilitats i veurem els seus avantatges i al mateix moment els seus desavantatges. D'aquest n'hi haurà que no podrem implantar ja que no tenim el material necessari, però la resta els estudiarem i els posarem a prova amb diversos experiments per veure la seva fiabilitat.

Per a la realització de tot el projecte treballarem amb el robot educatiu Lego Mindstorms que resulta una eina de relativament baix cost amb la qual es poden realitzar diversos projectes educatius com de baixa complexitat, no iniciats a la programació, fins als que tenen anys d'experiència en aquest camp, a causa de la versatilitat que aporta aquest producte. Per a poder programar estudiarem les prestacions que aporten cada llenguatge i entorn i entre aquestes triarem la que resulti més beneficiosa pel nostre treball.

Tot aquest estudi és per trobar una manera fiable de seguir el mòbil, però l'objectiu final és que no es quedi només amb això, sinó que aplicar aquests mètodes per a projectes més complexos com a programes de recerca de trajectòries, les quals necessiten saber la posició del mòbil el més exacte possible per a poder fer els càlculs de la trajectòria més optima en cada moment.

3. Objectius

- Trobar els diferents mètodes pel seguiment de la trajectòria
- Interacció entre l'ordinador i el robot Mindstorms
- Estudiar els mètodes i implementar-los al robot Mindstorms
- Realitzar proves dels mètodes i veure la seva fiabilitat
- Extreure les conclusions i proposar millores o implementacions a projectes més complexos.

4. Informació preliminar

Matlab

Matlab prové de la unió de les paraules matriu ("Matrix") i laboratori ("Laboratory"), és una potent eina de càlcul i simulació dirigida a enginyers o per l'àmbit científic, ja que permet el processat i el maneig de grans quantitats de dades de manera simple i ràpida. A més a més de les nombroses aplicacions d'ampliació fetes per usuaris o altres empreses que augmenten la capacitat i versatilitat de l'aplicació. Així com eines de processat d'imatge, de filtratge de senyals, de control, de computació financera, etc.

Dins de Matlab tenim la Toolbox Simulink que permet fer el muntatge dels sistemes per blocs i simplificar la programació de manera considerable per tal de facilitar el processament de les dades al poder comprimir els sistemes en blocs de processament i així anar creant encapsulaments que faciliten la programació visual quan es treballa amb senyals de dades.

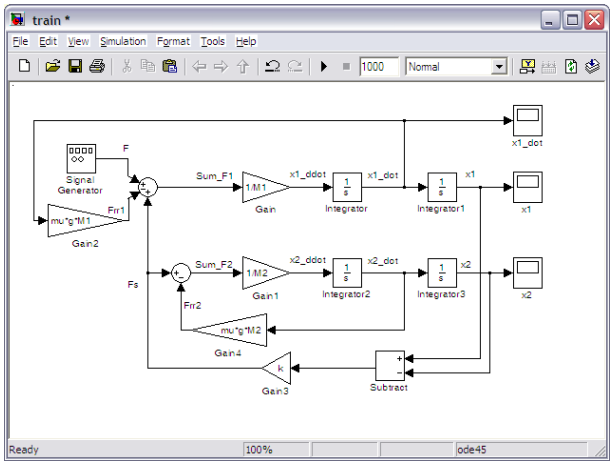


Figura 1 Captura de l'entorn Simulink

Lego Mindstorms

Lego Mindstorms és un kit de robòtica educativa distribuït per LEGO en col·laboració inicial amb el MIT. Hi ha hagut diverses versions; la primera generació RCX va ser la versió de prova per veure la acceptació del públic infantil, però gracies a l'interès del públic adult aficionat a la programació i a la robòtica, el producte va tenir millor acollida de la esperada amb el sorgiment de llenguatges de programació, apart del proporcionat per LEGO, i de diferents components fets per altres empreses. Això va portar a una remodelació millorant les capacitats amb la versió NXT(Versió 1.0, 1.1, 2.0 y 2.1). Actualment aquesta es la més estesa i en la qual hi ha més varietat de sensors i complements de terceres empreses. Hi ha una versió més moderna (EV3) però encara no té el mateix suport.

És un producte bastant complert amb una gran varietat de sensors i complements, com ja he dit, això li dona una gran versatilitat en la realització de diferents projectes, amb una ràpida recerca per la web podem veure com hi ha des de màquines que resolen el cub de Rubik, robots que pugen escales, fins a models de plantes de producció totalment automatitzades. Ja no és només una joguina perquè els nens aprenguin, sinó que es una eina on adults o estudiants superiors poden realitzar projectes sorprenents i poden dur-los a terme d'una manera econòmica on no seria necessària la inversió i el disseny d'altres components.

Llenguatges/Entorns de programació

Apart del llenguatge de programació proporcionat per LEGO, que és un llenguatge de programació d'entorn gràfic, hi ha hagut diferents desenvolupadors que han creat els seus entorns per a persones que vulguin profunditzar o vulguin crear projectes més complexos o amb interacció amb altres aparells.

- BrickOS

  És una llibreria d'instruccions i programes que permeten al programador ingressar de forma directa a la BIOS del bloc i instal·lar un petit sistema operatiu, per a poder ser programat en C, C++ i assembler.

- LejOS

  A diferència del BrickOS no s'ha d'instal·lar cap sistema operatiu, sinó que instal·la una màquina virtual de Java i així el bloc pot ser programat en Java

- Robot C

  Entorn de programació que utilitza una modificació de C amb les pròpies instruccions per a poder fer funcionar els diferents sensors i actuadors.

- BirxxC

  Entorn de programació basat en NXC(Not-Exactly-C) un llenguatge basat en C però incloent funcions pròpies semblant a l'entorn Robot C, però lliure.

- Matlab/Simulink

  Amb les llibreries es pot fer funcionar el bric a través de comandes per la consola en llenguatge de programació Matlab, i amb la *toolbox* corresponent, també es pot fer des de l'entorn Simulink. Aquestes llibreries o *add-ons* con Matlab les anomena, són proveïdes per l'instal·lador oficial de Matlab.

  Aquest serà el mètode que usaré per fer les proves experimentals ja que Matlab és una potent eina la qual em servirà per analitzar posteriorment les dades de manera ràpida i eficient. En la imatge podem veure tots els blocs que afegeix la llibreria del NXT.

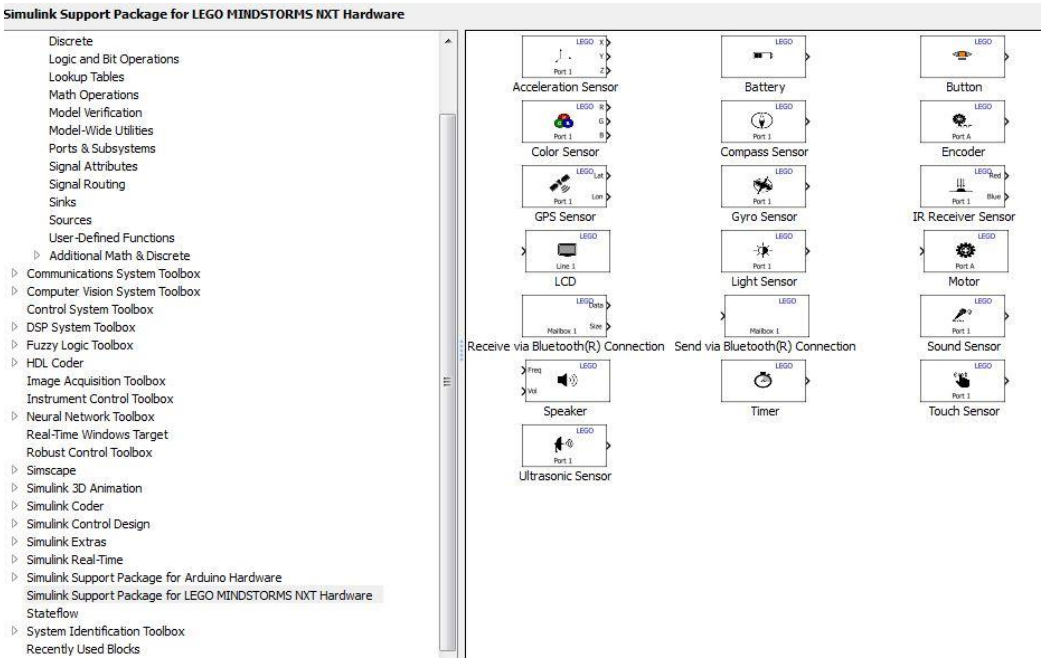


Figura 2 Captura dels blocs proporcionats per la llibreria NXT

I podria seguir aquesta llista fins omplir varies fulles, la gent ha adaptat gairebé tots els llenguatges de programació o n'ha creat modificacions per poder programar els blocs NXT i EV3. Jo només he nombrat els més usats o amb més suport.

Bloc

O també anomenat Brick, es el centre computacional del robot, d'aquí surten totes les comunicacions amb els sensors i motors a través d'unes connexions, en aquest bloc també hi ha la possibilitat de ficar un bateria, per defecte el bloc NXT funciona amb piles AA.

En la següent taula podem veure les característiques de cada model:

EV3: NXT: RCX:

Figura 4 Fotografia del bloc EV3

Figura 5 Fotografia del bloc NXT

Figura 3 Fotografia del bloc RCX

	EV3	**NXT**	**RCX**
Data de sortida	September 2013	July 2006	1998
Pantalla	178×128 pixel Monochrome LCD	100×64 pixel Monochrome LCD	segmented Monochrome LCD
Processador principal	TI Sitara AM1808 (ARM926EJ-S core) @300 MHz	Atmel AT91SAM7S256 (ARM7TDMI core) @48 MHz	Hitachi H8/300 @16 MHz
Memòria principal	64 MB RAM 16 MB Flash microSDHC Slot	64 KB RAM 256 KB Flash	32 KB RAM 16 KB ROM
USB Host Port	Si	No	No
WiFi	Connexió opcional via USB port	No	No
Bluetooth	Si	Si	No
Nombre d'entrades	4	4	3
Nombre de sortides	4	3	3

Com es pot observar, a cada versió se li han afegit noves característiques o se n'han millorat d'altres per a que siguin més potents i eficients, ara mateix el model NXT 2.0 és probablement el model amb més suport, el EV3 va sortir només fa 2 anys i hi ha entorns que no s'han actualitzat encara per poder donar suport, i hi ha sensors que encara no funcionen

en aquest model. Però es molt possible que en uns anys el EV3 acabi sent més popular, ja que té característiques molt millors que el seu model anterior.

Sensors

Els sensors són extensions que permeten passar dades del món físic al digital per a que el bloc les pugui processar i interaccionar amb el món extern. Tots van connectats a través dels cables als ports d'entrada del bloc.

En aquest apartat hi ha una gran varietat, així que només nombraré els mes típics o els que he usat durant el projecte.

- Sensor de llum/ de color; reconeix diferents colors i lluminositat.

Figura 6 Sensor de llum

- Sensor tàctil; detecta quan alguna objecte està polsant.

Figura 7 Sensor tàctil

- Sensor d'ultrasons, tecnologia que permet saber la distància d'un objecte que té davant.

Figura 8 Sensor d'ultrasons

- Sensors de sons; detecta els sons en decibels.

Figura 9 Sensor de soroll

- Sensors de infrarojos; permeten el seguiment d'una balisa que emet infrarojos .

Figura 10 Sensor IR

Third-Parties:

Sensors no oficials de LEGO:

- Acceleròmetre(HITechnic); llegeix l'acceleració a la que està sotmès el sensor en els 3 eixos.

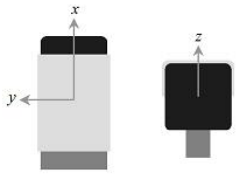


Figura 11 Acceleròmetre: https://www.hitechnic.com/sensors

- Gyroscop(HITechnic); llegeix el rati de rotació que pateix el sensor en un eix.

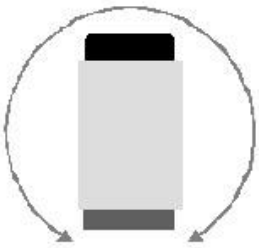


Figura 12 Giroscopi: https://www.hitechnic.com/sensors

- Sensor baromètric(HITechnic); permet la lectura de pressió atmosfèrica i la temperatura ambient.

Figura 13 Sensor baromètric: https://www.hitechnic.com/sensors

- Sensor de força(HITechnic); permet llegir la força aplicada en un punt.

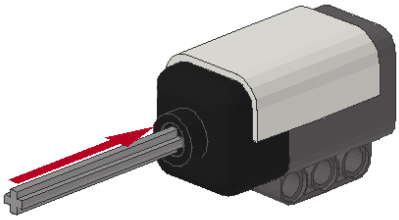


Figura 14 Sensor de força: https://www.hitechnic.com/sensors

- IMU; brúixola, acceleròmetre i giroscopi en el mateix sensor és conegut com IMU(Inertial Mesurment Unit).

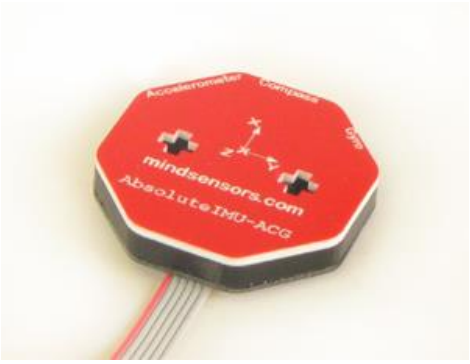


Figura 15 IMU: http://www.mindsensors.com/

En aquest apartat hi ha infinitat de sensors diferents, com per exemple Webcams, adaptadors a altres kit semblants, arrays de leds, bruixoles, etc. Com es pot observar la gran varietat només reafirma el que ja he dit de la forta acollida que ha tingut el producte en els aficionats del camp.

Però durant el desenvolupament del meu projecte nomes faré us del giroscopi (Figura 12 Giroscopi: https://www.hitechnic.com/sensors) i el acceleròmetre (Figura 11 Acceleròmetre: https://www.hitechnic.com/sensors).

Motors

Figura 16 Fotografies de tots el motors compatibles amb el robot Mindstorms

Com es pot veure de motors també hi ha molts tipus diferents i de característiques diferents, si es vol veure un comparació detallada de tots, recomano visitar aquesta pàgina: http://www.philohome.com/motors/motorcomp.htm. Jo només faré l'explicació de les característiques del que usaré pel meu projecte.

Figura 17 Fotografia del motor basic del NXT

En el kit bàsic del NXT només tenim un tipus de motor; és un motor de mitjana potència amb un seguit d'engranatges per mantenir la relació par-velocitat. Dins hi ha un motor, un encoder, la relació d'engranatges i el controlador de la potència del motor i la connexió amb el brick. Tot això està muntat sobre una carcassa de plàstic amb els diferents suports compatibles amb totes les peces de la companyia Lego, d'aquesta manera poder ser integrat en les construccions fàcilment.

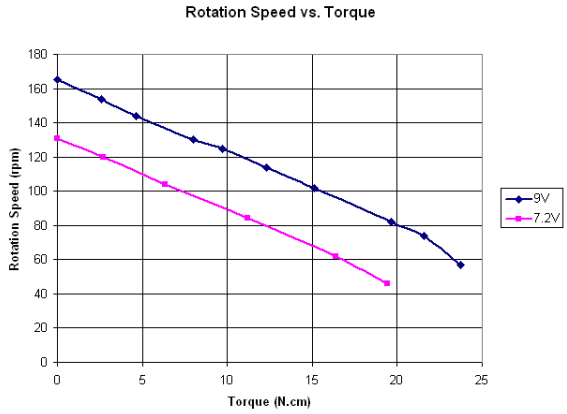


Figura 18 Grafica velocitat de gir-parell

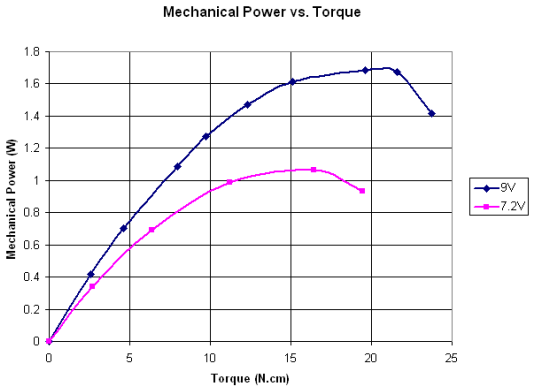


Figura 19 Grafica potència- parell

En les gràfiques anteriors podem veure la potència-parell i la de velocitat de rotació-parell. Hi ha dos maneres d'alimentar el motor, però no es configurable a través del programari. Dependran de l'estat de les piles i la bateria.

Figura 20 Fotografia del interior del motor

Aquí tenim una fotografia de l'interior del motor on podem veure la relació d'engranatges que resulta:

10:30:40		= 1:4
9:27		= 1:3
	10:20	= 1:2
	10:13:20	= 1:2
Total		1:48

Connectors

La construcció dels cables és semblant a la dels RJ12, utilitzats per a la connexió de xarxes i mòdems, però amb un petita modificació a la posició del "latch" per evitar la confusió, ja que els cables porten diferents voltatges i estan preparats per suportar diferents condicions.

Aquests cables permeten la connexió entre tots els sensors i motors amb el brick. Dins del kit inicial hi ha cables de diferent llargària per evitar que els cables molestin quan es fan connexions curtes i d'altres suficientment llargs per donar mobilitat.

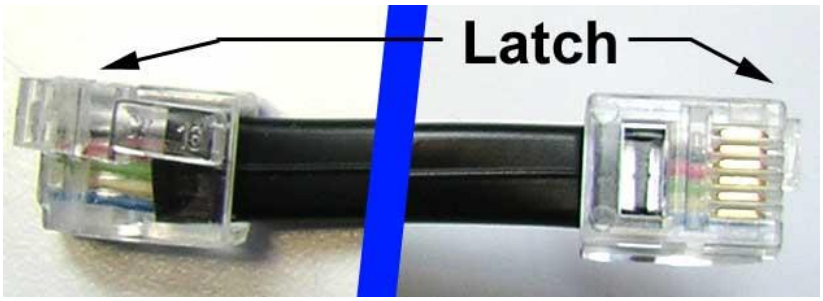


Figura 21Fotografia dels conector dels cables

5. Robòtica

La robòtica és una branca de la tecnologia que estudia i dissenya la creació de màquines que poden moure o realitzar tasques per si soles, tasques repetitives, tasques perilloses per a l'ésser humà, i interactuant amb el seu entorn a través de sensors i poder prendre decisions en les seves accions.

És un camp interdisciplinari que uneix estudis de mecànica, d'electrònica analògica, d'electrònica digital, informàtica, matemàtica i física.

Dins de la robòtica els diferents tipus de robots és classifiquen segons la seva morfologia;

- **Robots manipuladors:**

Figura 22 Exemple de robot manipualdor

O també coneguts com robots industrials, són els que estan més presents a l'àmbit de la industria. La majoria tenen l'aparença d'un braç, que estan dissenyats per manipular càrregues o eines. Poden fer moviments amb gran tonatge i amb gran precisió i rapidesa.

Aquests poden ser més o menys complexos segons els graus de llibertat que tinguin, des de 3 graus per la paletització fins a 6 graus de llibertat per la pintura o soldadura. Hi ha diverses empreses que es dediquen al disseny i distribució d'aquest tipus de robots i cada una té el seu model característic.

- **Robots mòbils:**

Els robots mòbils són tots aquells que estan dissenyats per a que es puguin desplaçar, es poden classificar segons el medi on ho facin:

Aquàtic; els robots d'exploració de les profunditats marines, on portar un humà seria complicat.

Aeri; la proliferació dels drons durant els últims anys està fent que aquest medi, on abans només hi havia aparells principalment militars o científics, ara estigui aconseguint que gairebé qualsevol persona aficionada pugui tenir un dron volador, normalment estan impulsats per hèlixs i en una configuració de cuadracopter.

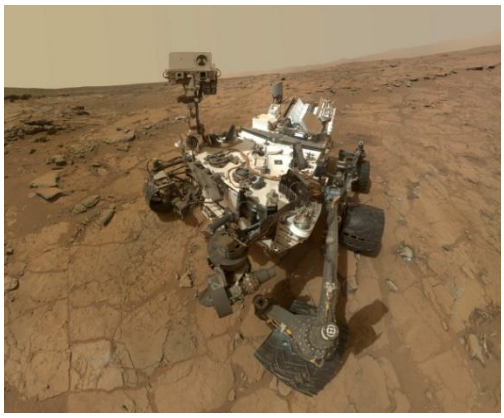


Figura 23 Curiosity robot d'exploracio espaciol de la NASA

Terrestre; aquests són els més coneguts i amb més ampli catàleg. Tenim des dels petits robots de joguina, passant per robots de neteja, fins als ròvers d'exploració espacial com el "Curiosity", que està actualment explorant la superfície de Mart. Aquests es poden moure de diferents maneres: amb potes, rodes d'eruga com els tancs, o rodes com els cotxes.

- **Androides:**

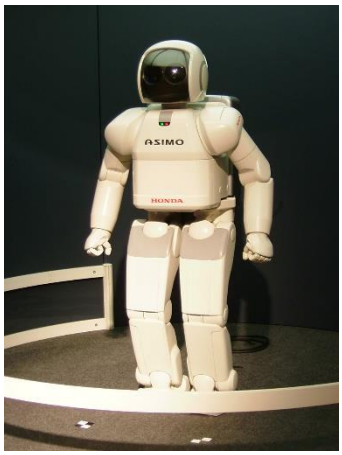


Figura 24 Robot androide ASIMO

Són el grup de robots el disseny del qual està pensat per a que s'assemblin en lo màxim possible als humans i siguin capaços de realitzar les mateixes accions que nosaltres. És el camp en el que menys avanç es té, ja que el moviment bípede és complicat de reproduir, i l'intent de donar intel·ligència per a que puguin realitzar les accions, està en desenvolupament actualment.

En aquest treball ens centrarem en els robots mòbils, més concretament els terrestres, que és desplacen amb rodes. Que depenent de com estiguin col·locades i de quin tipus siguin, poden canviar totalment el moviment del robot, aquestes disposicions són anomenades configuracions.

6. Configuracions

Pels robots mòbils hi ha el que són anomenades configuracions, que fan referència a com està muntat els sistema de desplaçament del robot. Hi ha diferents característiques de muntatge que defineixen cada una de les configuracions.

Components motrius

Hi ha diferents tipus de rodes i cada una té les seves característiques;

- Roda fixa:

  És la roda més bàsica, normalment està lligada a un motor i són les que donen moviment.

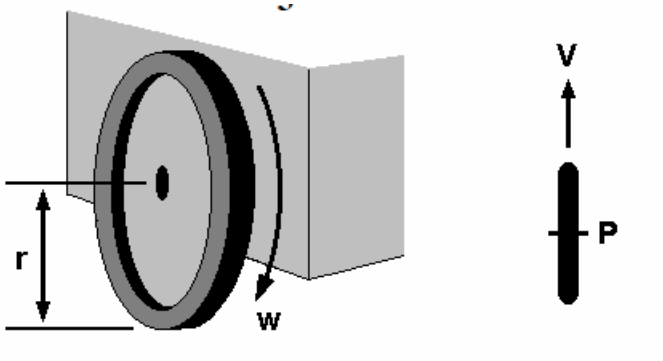


Figura 25 Roda fixa: http://www.esi2.us.es/~vivas/ayr2iaei/LOC_MOV.pdf

- Roda amb centre orientable:

  Ens permet orientar el robot si la roda està controlada i sinó pot actuar com a roda de suport.

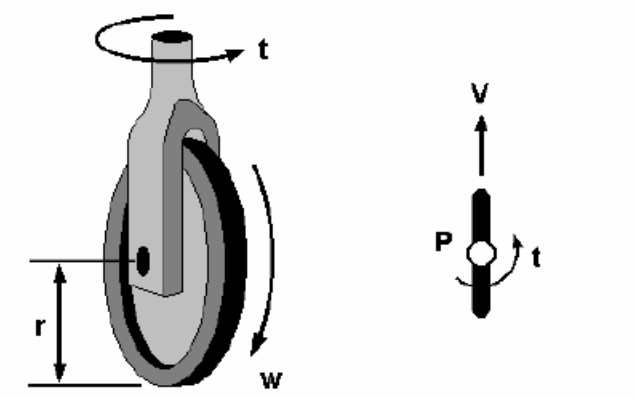


Figura 26 Roda amb centre orientable: http://www.esi2.us.es/~vivas/ayr2iaei/LOC_MOV.pdf

- Roda amb centre orientable desplaçat:

  Pot servir per donar direcció al robot, però també es pot utilitzar per donar suport al robot que segons el moviment, aquesta roda girarà i s'orientarà si no està controlada.

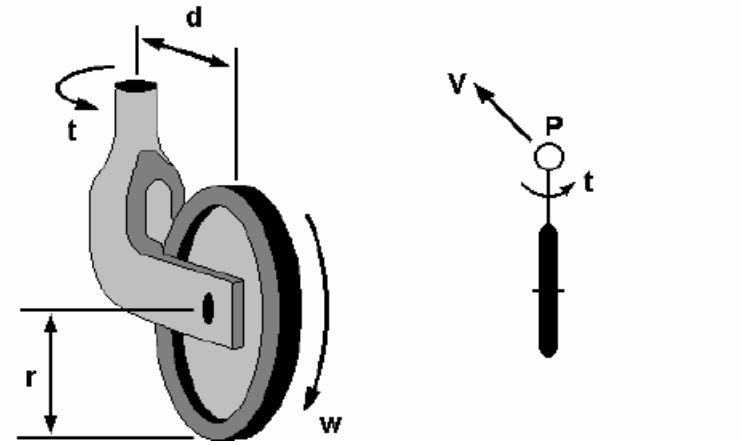


Figura 27 Roda amb centre orientable desplaçat: http://www.esi2.us.es/~vivas/ayr2iaei/LOC_MOV.pdf

- Roda omnidireccional o sueca:

Aquesta roda permet el moviment en gairebé totes les direccions. A diferència de les altres que només poden anar endavant i cap endarrere, i girar. Aquest tipus donen molta versatilitat, però de la mateixa manera també compliquen molt els models cinemàtics a l'hora de calcular.

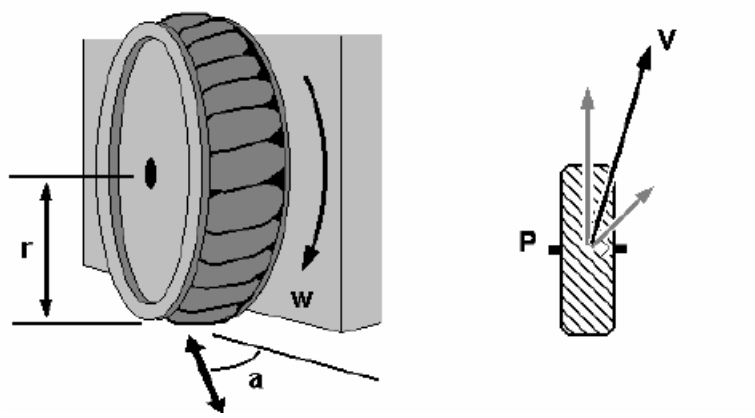


Figura 28 Roda omnidireccional: http://www.esi2.us.es/~vivas/ayr2iaei/LOC_MOV.pdf

- Roda d'eruga:

Aquestes són utilitzades, com els trens de rodes, per augmentar la estabilitat del robots i el seu poder de tracció però resulten bastant pesades i complicades de controlar.

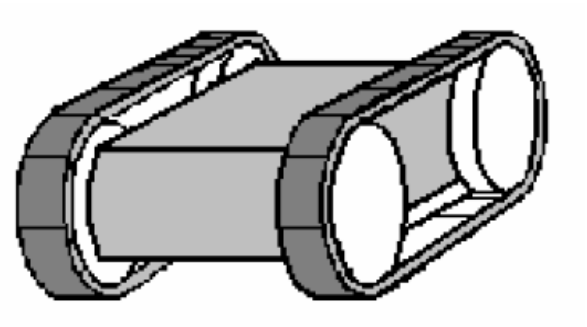


Figura 29 Roda d'eruga: http://www.esi2.us.es/~vivas/ayr2iaei/LOC_MOV.pdf

- Roda esfèrica:

El robot se suporta sobre una o varies esferes, i de diferents maneres es transmet moviment sobre aquestes esferes i permet al robot moure's.

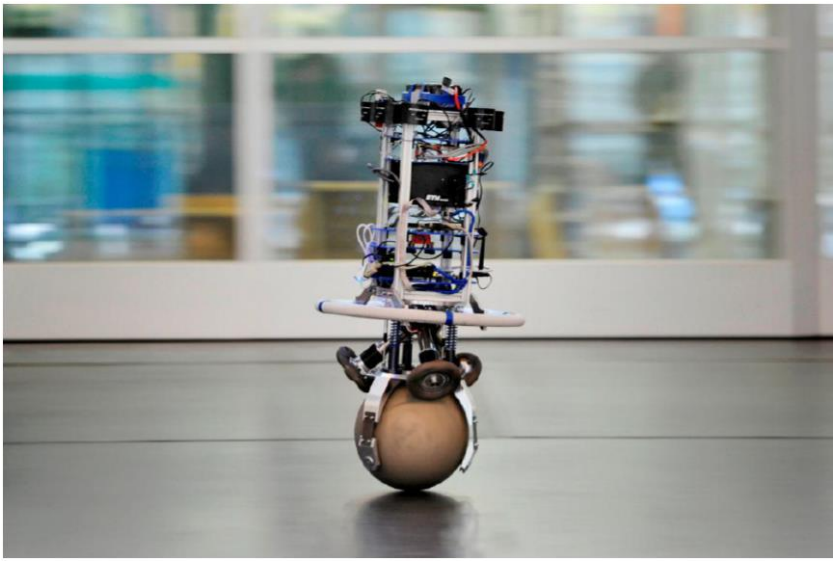


Figura 30 http://www.youtube.com/watch?feature=player_embedded&v=sB9IowB8nx8

- Potes:

Són versàtils, capaces de superar desnivells (pujar i baixar escales) però el control és molt complicat. Ara mateix hi ha pocs models que hagin donat prestacions acceptables. La majoria de configuracions intenten imitar a animals com per exemple aranyes, centpeus, guepards, humans, etc.

Figura 31 Exemple de robot impulsat per potes: http://www.superrobotica.com/S300160.htm

Segons la funció que realitzin les rodes es poden dividir també en;

- Roda motriu: la que proporciona la força motriu del robot i fa que es desplaci
- Roda directriu: rodes d'orientació controlable
- Roda fixa: que giren sobre el seu eix sense tracció motriu
- Roda boja o solta: rodes orientables no controlades

Configuracions

Els tipus de rodes anteriorment mencionats es poden muntar de diferents maneres i amb diferent nombre, cada configuració tindrà característiques diferents:

- Model diferencial; és probablement el model més simple de tots, ja que només requereix dues rodes fixes independents i la cinemàtica resulta bastant simple. Però te diferents problemes com les restriccions no holònomes i poca estabilitat, tot i que això últim es pot solucionar afegint una tercera roda boja, continua tenint dificultats superant els desnivells.
- Model tricicle clàssic; està compost de dos rodes fixes que poden no ser independents, i una roda directriu de centre orientable que es la que ens permet donar la direcció al robot. És un dels models més estables i es pot adaptar fàcilment als desnivells del terra.
- Model Ackerman: aquest correspondria al model dels cotxes, dues rodes fixes en la part posterior i dues orientables en la part frontal. La tracció pot ser a 2 rodes o a 4. Té una cinemàtica bastant complexa.
- Configuració sincrònica: totes les rodes, generalment tres, són de direcció i de tracció, permet fer girs sense tenir que girar tot el xassís del robot, resulta bastant necessari per a robots que hagin de seguir punts o hagin de mantenir una orientació.
- Configuracions especials: són les que utilitzen cadenes o un tren de rodes, aporten més estabilitat o més versatilitat per superar desnivells, un exemple són els robots d'exploració espacial.

Configuració escollida

Inicialment es volien fer diferents configuracions i poder comparar-les, però per la complexitat que ja va suposar fer només un model i la falta de temps, vam decidir quedar-nos amb un sol.

Per a realitzar el meu treball he escollit la configuració diferencial, de dos rodes fixes amb un motor per a cada una, disposades de tal manera per poder repartir el pes. I una roda fixa al darrera per donar estabilitat.

Dins de la configuració diferencial hi ha infinitat de maneres de construcció, però jo he escollit una construcció amb un centre de gravetat baix, i que em permetia col·locar els sensors d'acceleració i el giroscopi gairebé al centre de gir del robot, així evitava molts problemes que em donaven altres models on la col·locació dels sensors era complicada o poc estable, i donava lloc a vibracions més fortes.

Restriccions de moviment

Primer em de definir les restriccions no holònomes; hi ha robots que només es poden moure instantàniament en algunes direccions, principalment endavant i endarrere, però no ho pot fer en altres direccions com cap a dreta i esquerra, a causa de la disposició de les seves rodes i del tipus. Per això no es poden tractar les equacions de la mateixa manera per a totes les configuracions, sinó que cada un tindrà un grup d'equacions que relacionaran l'orientació de les rodes i la seva velocitat amb la forma que es mourà el robot.

Per a trobar això primer de tot s'ha de veure quines restriccions aporten cada roda.

Cada roda té un tipus de restriccions;

- Roda fixa:

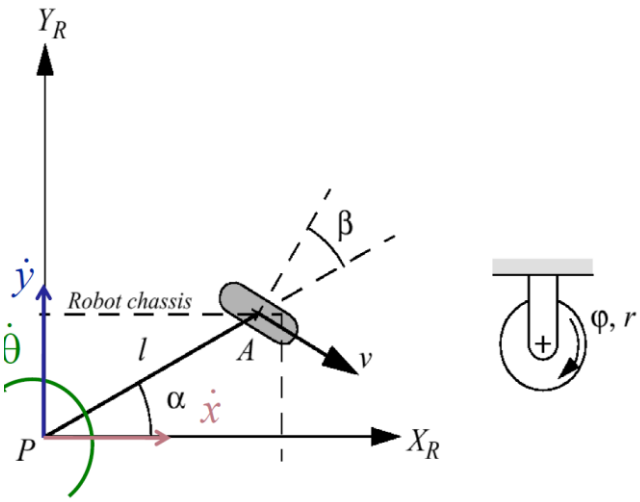


Figura 32 Diagrama roda fixa

En el pla de la roda:

$$[sen(\alpha+\beta) \quad -\cos(\alpha+\beta) \quad -lcos(\beta)]R(\theta)\dot{\xi} - r\dot{\varphi} = 0$$

Equació 1

En el pla ortogonal de la roda:

$$[cos(\alpha+\beta) \quad sen(\alpha+\beta) \quad lsen(\beta)]R(\theta)\dot{\xi} = 0$$

Equació 2

- Roda amb centre orientable:

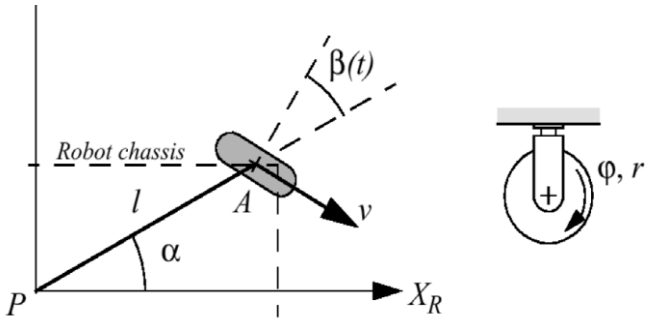


Figura 33 Diagrama roda orientable

En el pla de la roda:

$$[sen(\alpha+\beta) \quad -\cos(\alpha+\beta) \quad -lcos(\beta)]R(\theta)\dot{\xi} - r\dot{\varphi} = 0$$

Equació 3

En el pla ortogonal de la roda:

$$[cos(\alpha+\beta) \quad sen(\alpha+\beta) \quad lsen(\beta)]R(\theta)\dot{\xi} = 0$$

Equació 4

La diferència amb la roda anterior és que en aquest cas la β és variable durant el temps.

- Roda amb centre orientable desplaçat:

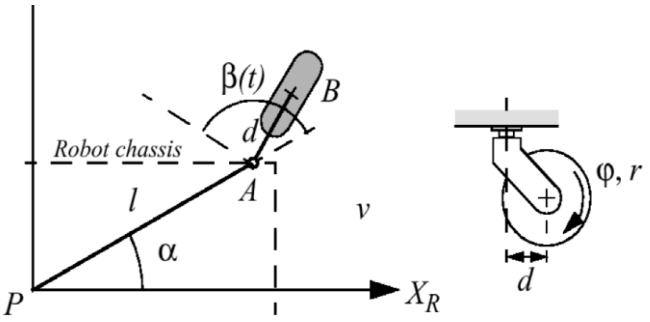


Figura 34 Diagrama roda orientable amb centre desplaçat

En el pla de la roda:

$$[sen(\alpha+\beta) \quad -\cos(\alpha+\beta) \quad -lcos(\beta)]R(\theta)\dot{\xi} - r\dot{\varphi} = 0$$

Equació 5

En el pla ortogonal de la roda:

$$[cos(\alpha+\beta) \quad sen(\alpha+\beta) \quad d + lsen(\beta)]R(\theta)\dot{\xi} + d\dot{\beta} = 0$$

Equació 6

- Roda omnidireccional:

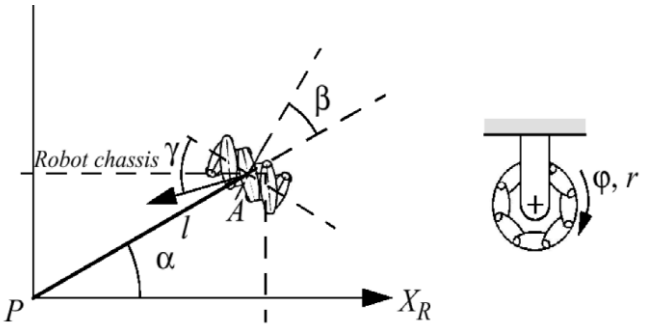


Figura 35 Diagrama omnidireccional

En el pla de la roda:

$$[sen(\alpha+\beta+\gamma) \quad -\cos(\alpha+\beta+\gamma) \quad -lcos(\beta+\gamma)]R(\theta)\dot{\xi} - r\,cos(\gamma)\dot{\varphi} = 0$$

Equació 7

A partir d'aquestes equacions es poden treure, a través de un mètode estandarditzat, gairebé totes les equacions de totes les configuracions possibles. En l'apartat (8. Particularització de les equacions) faré un exemple amb el model diferencial.

7. Cinemàtica

La cinemàtica és l'estudi del moviment dels cossos en l'espai durant el temps, en el nostre cas serà un mòbil que només es mourà en un pla.

Ara definirem els conceptes bàsics;

La posició; és el punt en l'espai cartesià que defineix on està el mòbil, en el nostre cas nomes tindrà dues coordenades (x,y).

$$P = (x_p, y_p)$$

Equació 8

La velocitat; és la variació de posició d'un mòbil en un període de temps, d'una altra manera podem dir que és la derivada de la posició respecte el temps.

$$\bar{V} = \frac{\Delta P}{\Delta t}$$

Equació 9

$$V = \lim_{\Delta t \to 0} \frac{\Delta P}{\Delta t} = \frac{dP}{dt} = \dot{P}$$

Equació 10

L'acceleració; és la variació de velocitat d'un mòbil en un període de temps, d'una altra manera podem dir que és la derivada de la velocitat respecte el temps. I que al mateix moment és la segona derivada de la posició.

$$\bar{A} = \frac{\Delta V}{\Delta t}$$

Equació 11

$$A = \lim_{\Delta t \to 0} \frac{\Delta V}{\Delta t} = \frac{dV}{dt} = \dot{V} = \ddot{P}$$

Equació 12

Si tenim l'acceleració durant un el temps podrem obtenir la posició i la velocitat del mòbil integrant.

Quedaria com:

$$v(t) = v(0) + \int_0^t a(t)\, dt$$

Equació 13

I de la mateix manera si tornem a integrar:

$$p(t) = p(0) + \int_0^t \left(v(0) + \int_0^t a(t)\, dt \right) dt$$

Equació 14

Amb aquestes fórmules les introduirem al Matlab i amb les dades obtingudes dels diferents sensors intentarem extreure la trajectòria.

Matemàtica discreta

Totes aquestes equacions estan escrites per a temps continu, on la diferència del temps entre elles és infinitesimal. Però malauradament la realitat és totalment diferent, la majoria dels sensors usats són digitals i això implica que tenen un temps de mostreig, el qual farà que passéssim a un camp de les matemàtiques que és diu matemàtica discreta. Aquest camp s'encarrega de l'anàlisi de conjunts numerables o comptables, no és com a la matemàtica continua on es pot trobar sempre una aproximació més precisa entre dos nombres, en la matemàtica discreta són elements totalment comptables, com serien els nombres naturals. Un exemple serien els nombres natural 2 i 3, entre ells no existeix cap nombre natural més.

Llavors les equacions anteriors s'han de discretitzar amb un concepte conegut com temps de mostreig(ts); aquest temps és el que tarda un sensor en recollir una dada i poder tornar a fer-ho assegurant que aquesta serà la correcta. Aquest temps s'ha d'ajustar al temps de computació de totes les operacions que ha de fer el robot i també al temps del conjunt d'accions que està fent el robot. Per exemple no és necessari que el robot estigui comprovant cada mil·lèsima de segon si ha arribat una peça si les peces arriben cada minut, es un cost innecessari. Com a contrapartida s'ha d'ajustar a sistemes crítics on el retard més petit pot costar vides humanes. Per aquestes raons escollir el temps de mostreig pot ser una decisió molt complicada i ha de ser meditada.

La derivació en temps discret queda com:

$$v_k = \frac{p_k - p_{k-1}}{ts}$$

Equació 15

On p_{t-1} significa la posició en el estat anterior i p_t la posició en l'estat actual.

La integració quedaria com:

$$p_k = p_{k-1} + v_k * ts$$

Equació 16

8. Particularització de les equacions

Tenint les equacions anteriors i sabent la configuració del nostre robot podem extreure les equacions que relacionen la velocitat lineal de les rodes i la velocitat angular i lineal de tot el robot. Primer ho farem de manera geomètrica i després farem pel mètode estandarditzat.


Tenim:

2 rodes motrius fixes paral·leles i una fixa darrera.

L = longitud de la roda

B = distancia entre les 2 rodes

R = radi de la roda

La roda fixa del darrere no aporta res a la cinemàtica ni aplica cap restricció, simplement esta allí com a suport, com podria estar una roda esfèrica, tal com incorpora de sèrie el model EV3.

Per el mètode geomètric;

Sabem que per cinemàtica bàsica del moviment circular, la velocitat lineal del mòbil serà la velocitat angular per el radi $v = \omega * R$. Si apliquem aquesta equació per a cada roda obtenim:

$$v_r = \omega(R + \frac{B}{2})$$

Equació 17

$$v_l = \omega(R - \frac{B}{2})$$

Equació 18

Sumant les dos equacions anteriors:

$$v_r + v_l = 2\omega R = 2v$$

Equació 19

I restant-les tenim:

$$v_r - v_l = \omega B$$

Equació 20

Si aïllem la velocitat lineal i la velocitat angular del mòbil, obtenim aquestes respecte a la velocitat lineal de cada roda.

$$v = \frac{v_r + v_l}{2}$$

Equació 21

$$\omega = \frac{v_r - v_l}{B}$$

Equació 22

I si anem més enllà i diem que la velocitat lineal de cada roda és la velocitat angular pel radi, suposem que les rodes tenen el mateix radi, obtenim la relació de les rotacions de rodes i el moviment del mòbil.

$$v = \frac{(\omega_r + \omega_l)r}{2}$$

Equació 23

$$\omega = \frac{(\omega_r - \omega_l)r}{B}$$

Equació 24

Ara seguint les equacions de les restriccions i el mètode estandarditzat, podem obtenir les equacions per al model diferencial.

Primer de tot faré l'explicació, definim el vector de postura del robot i la matriu de rotació.

$$\xi = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$$

Equació 25

$$R(\theta) = \begin{bmatrix} \cos(\theta) & sen(\theta) & 0 \\ -sen(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Equació 26

La suma total de rodes de totes les classes:

$$N = N_f + N_{co} + N_{cd} + N_o$$

Equació 27

el subíndex f representa les rodes fixes, *co* les rodes de centre orientable, *cd* rodes de centre desplaçat orientable i *o* les rodes omnidireccionals.

Les restriccions de moviment es poden representar amb la següent forma general:

$$J_1(\beta_{co}, \beta_{cd})R(\theta)\dot{\xi} - J_2\dot{\varphi} = 0$$

Equació 28

$$C_1(\beta_{co}, \beta_{cd})R(\theta)\dot{\xi} - C_2\dot{\beta}_{cd} = 0$$

Equació 29

De les equacions anteriors es té que

$$J_1(\beta_{co}, \beta_{cd}) = \begin{bmatrix} J_{1f} \\ J_{1co}(\beta_{co}) \\ J_{1cd}(\beta_{cd}) \\ J_{1o} \end{bmatrix}$$

Equació 30

on $J_{1f}, J_{1co}(\beta_{co}), J_{1cd}(\beta_{cd}), J_{1o}$ són matrius de dimensions$(N_f \times 3), (N_{co} \times 3), (N_{cd} \times 3)$ i $(N_o \times 3)$ respectivament i la seva estructura surt de les equacions de restricció(Equació 1, Equació 3, Equació 5, Equació 7Equació 4) de cada roda. J_2 és una matriu diagonal de dimensió (N × N) formada per els radis de les rodes.

Per altra banda, de l'equació obtenim

$$C_1(\beta_{co}, \beta_{cd}) = \begin{bmatrix} C_{1f} \\ C_{1co}(\beta_{co}) \\ C_{1cd}(\beta_{cd}) \end{bmatrix}$$

Equació 31

on C_{1f}, $C_{1co}(\beta_{co})$ i $C_{1cd}(\beta_{cd})$ són matrius de dimensions dimensions$(N_f \times 3), (N_{co} \times 3)$ i $(N_{cd} \times 3)$ respectivament formades per les equacions(Equació 2, Equació 4, Equació 6).

$$C_2 = \begin{bmatrix} 0 \\ 0 \\ C_{2cd} \end{bmatrix}$$

Equació 32

Per últim C_{2cd} és una matriu diagonal de $(N_{cd} \times N_{cd})$ amb les distancies *d* de les rodes de centre desplaçat.

Per a seguir el mètode primer de tot s'ha de fer un taula amb totes les posicions respectives de les rodes. Com ja he dit anteriorment la roda del darrera no realitza cap treball en el moviment, simplement és un suport.

Rodes	**α**	**β**	**L**
1f	-π/2	π	B/2
2f	π/2	0	B/2

I substituïm els valors per a formar les equacions:

$$J_1 = \begin{bmatrix} sen(-\frac{\pi}{2}+\pi) & -\cos(-\frac{\pi}{2}+\pi) & -\frac{B}{2}\cos(\pi) \\ sen(\frac{\pi}{2}) & -\cos(\frac{\pi}{2}) & -\frac{B}{2}\cos(0) \end{bmatrix}$$

Equació 33

$$J_1 = \begin{bmatrix} 1 & 0 & \frac{B}{2} \\ 1 & 0 & -\frac{B}{2} \end{bmatrix}$$

Equació 34

Com les rodes són paral·leles podem reduir la C_1:

$$C_1 = \begin{bmatrix} \cos(-\frac{\pi}{2}+\pi) & \text{sen}(-\frac{\pi}{2}+\pi) & l\, sen(\pi) \end{bmatrix}$$

Equació 35

$$C_1 = [0 \quad 1 \quad 0]$$

Equació 36

$$J_2 = \begin{bmatrix} r & 0 \\ 0 & r \end{bmatrix}$$

Equació 37

Apliquem el que tenim a les equacions anteriors i s'obté:

$$\begin{bmatrix} \begin{bmatrix} 1 & 0 & \frac{B}{2} \\ 1 & 0 & -\frac{B}{2} \end{bmatrix} \\ [0 \quad 1 \quad 0] \end{bmatrix} R(\theta)\dot{\xi} = \begin{bmatrix} J_2 \begin{bmatrix} \dot{\varphi}_l \\ \dot{\varphi}_r \end{bmatrix} \\ 0 \end{bmatrix}$$

Equació 38

Aïllem el vector de velocitats:

$$\dot{\xi} = R(\theta)^{-1} \begin{bmatrix} 1 & 0 & \frac{B}{2} \\ 1 & 0 & -\frac{B}{2} \\ 0 & 1 & 0 \end{bmatrix}^{-1} \begin{bmatrix} \begin{bmatrix} r\dot{\varphi}_l \\ r\dot{\varphi}_r \end{bmatrix} \\ 0 \end{bmatrix}$$

Equació 39

$$\dot{\xi} = R(\theta)^{-1} \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 1 \\ \frac{1}{B} & \frac{1}{B} & 0 \end{bmatrix} \begin{bmatrix} \begin{bmatrix} r\dot{\varphi}_l \\ r\dot{\varphi}_r \end{bmatrix} \\ 0 \end{bmatrix}$$

Equació 40

I només amb un simple cop d'ull es pot veure com hem arribat a la mateixa solució que per geomètric, tot i que resulta molt complexa per un model tant simple. Quan tenim models més complicats i la solució geomètrica o per observació no resulta tant obvia, aquest és un mètode fiable.

9. Metodologies per al seguiment de posició

Aquí faré una petita explicació dels diferents mètodes que hi ha actualment per a poder seguir la trajectòria de un robot mòbil.

Encoders

Aquest mètode es basa en que si sabem la velocitat angular de les rodes podem obtenir la seva velocitat lineal multiplicant pel radi. I amb el model cinemàtic caracteritzat del robot podem obtenir la seva posició en tot moment.

Dins dels motors del NXT ve incorporat un encoder, el qual podem utilitzar per veure quantes voltes ha fet la roda.

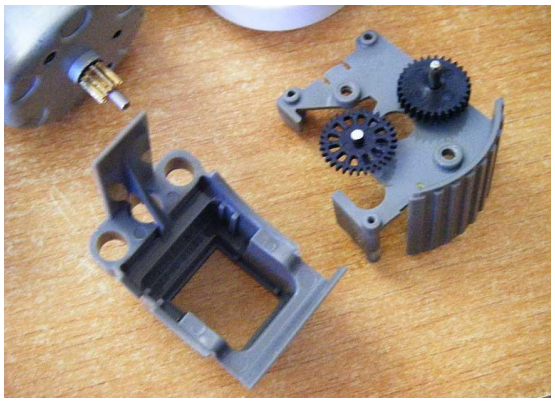


Figura 36 Fotografia del encoder del motor NXT

El encoder es òptic i té 12 ranures, la raó de reducció dels engranatges entre el motor i el encoder es de 10:32. Si sabem que la relació motor roda és de 1:48; llavors una volta de la roda són 48*10/32 = 15 voltes de l'engranatge de l'encoder, 15*12 = 180 ranures per volta de la roda. Però usant els dos salts de les ranures obtenim 360. La resolució final serà de 1 grau.

Es pot derivar respecte el temps aquest compte de graus i així obtenim la velocitat angular de cada roda, utilitzant les equacions deduïdes anteriorment podem obtenir la velocitat lineal i angular de tot el robot.

$$v(t) = \frac{r(w_L(t) + w_R(t))}{2}$$

Equació 41

$$w(t) = \frac{r(w_L(t) + w_R(t))}{b}$$

Equació 42

Tenint això a través de la cinemàtica directa del model diferencial podem obtenir la posició del robot.

Avantatges: bastant precís ja que la resolució del sensor resulta suficient per obtenir resultats totalment fiables. No necessita gran capacitat de processament ja que els càlculs són simples. Els encoders són un afegit bastant comú als motors, així que és bastant fàcil implementar-ho.

Desavantatges: aquest mètode segueix les rotacions de les rodes, però això no vol dir que una rotació de les rodes impliqui moviment, pot ser que les rodes derrapin a causa d'una acceleració gran o un terra on la tracció sigui baixa. I també hi ha el problema que el robot es quedi encallat en alguna posició on una roda quedi flotant o que hagi xocar contra una paret, aquest mètode no pot detectar això i si passa, els resultats seran els mateixos que si el robot hagués seguit normalment i no hi hagués paret o pertorbació en la seva trajectòria.

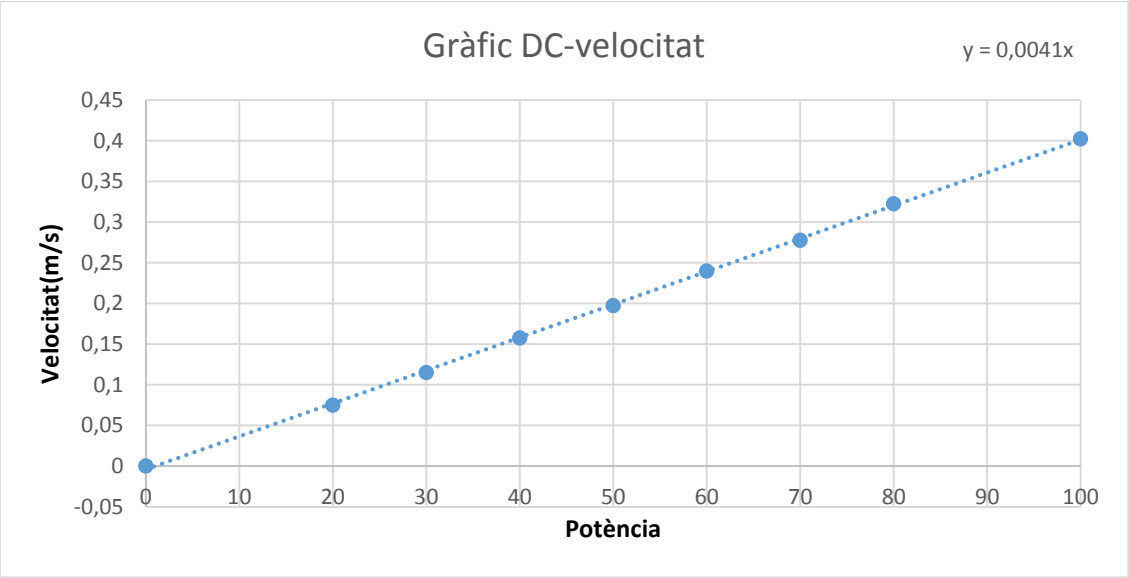
Teòric

Agafant els valors que introduïm als motors podem fer una predicció de com s'ha mogut el robot. La potència als motors es introduïda en tant per cent(100% - 0%). Experimentalment podem obtenir a quina velocitat es mou el robot depenent de la potència del motor, i aplicant cinemàtica poden deduir la posició del robot.

Per obtenir aquesta relació es poden fer un seguit de proves, per exemple moure el robot durant 4 segons en línia recta i veure quina distancia ha fet, i això prova-ho en diferents potències.

Potència(DC)	Distancia(m)	Velocitat(m/s)	K_motor(v/DC)
20	0.3	0.075	0.00375
30	0.46	0.115	0.003833
40	0.63	0.1575	0.003938
50	0.79	0.1975	0.00395
60	0.96	0.24	0.004
70	1.11	0.2775	0.003964
80	1.29	0.3225	0.004031
100	1.61	0.4025	0.004025

Fent la regressió lineal ens dóna una K_motor de 0.0041 però aquest valor depèn molt de la superfície, de la bateria del robot, etc.



Obtenim una constant anomenada K_motor que és la relació entre la velocitat lineal del robot i la potència enviada.

Avantatges: fidel a les comandes enviades al robot i a la trajectòria que hem predefinit en les comandes enviades al robot.

Desavantatges: la velocitat és una estimació de la realitat, la relació no és del tot lineal i varia segons el pes del robot, la càrrega de la bateria i la superfície on està. I com ja he dit, és totalment fidel a les comandes donades al robot, però que això no implica que sigui el desplaçament real del robot.

Sensors inercials

Aquest mètode es basa en la idea que ja he dit anteriorment de que si es pot saber l'acceleració a la qual està sotmès el robot, podem saber en tot moment a quina velocitat es mou i en quina posició estarà.

Els acceleròmetres ens proporcionen una lectura de l'acceleració a la qual està sotmès el robot, en el nostre cas en els tres eixos. Si amb això afegim un giroscopi per saber l'orientació del robot podem seguir la trajectòria.

Apliquem la formula (Equació 14), utilitzant matemàtica discreta, ja que estem treballant en sensors que tenen un temps de mostreig:

$$v_k = v_{k-1} + a_k * ts$$

Equació 43

$$p_k = p_{k-1} + v_{k-1} * ts + \frac{1}{2} a_k * ts^2$$

Equació 44

On ts serà la diferencia entre el temps actual i el temps de l'anterior mostra:

$$ts = t_k - t_{k-1}$$

Equació 45

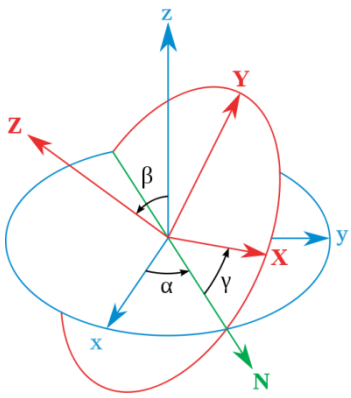


Figura 37 Rotacions en l'espai: www.wikipedia.org

L'acceleració l'haurem de dividir en els dos eixos de moviment del robot ja que és un vector sobre el pla, quan el robot giri també ho farà el sensor i els eixos del sensor, per això s'han d'aplicar les conegudes com rotacions de Euler per transferir aquests eixos desplaçats als nostres eixos bases.

En el nostre cas com només treballem en un pla de dos eixos només podem tenir rotacions en l'eix z.

Un altre mètode per calcular aquestes rotacions seria les *Quaternions*, que permeten una computació molt més simple i ràpida, i també l'extracció dels angles de manera simple, però està més orientat a les rotacions en l'espai i per fer-ho en un pla, queda com un mètode massa complex comparat al de Euler.

$$R(\theta) = \begin{bmatrix} \cos(\theta) & sen(\theta) & 0 \\ -sen(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Equació 46

Per a trobar l'angle i aplicar les rotacions jo utilitzaré el sensor del giroscopi.

Avantatges: és un mètode que agafa mesures del que realment està passant amb el robot, amb això podem detectar quan hi ha pertorbacions externes en el moviment del robot.

Desavantatges: els acceleròmetres són uns sensors amb molt de soroll, i bastant complicats de filtrar perquè també reben totes les vibracions del robot.

Com estem en un sistema on les acceleracions són bastant petites comparades amb el soroll del sensor, això encara complica més les coses.

I amb el mètode de la integració, el més petit error té bastanta importància i es va ampliant i acumulant durant el procés, i això provoca que cada error se sumi a l'anterior arribant a deixar inútils les mesures si no es fan correccions.

Seguiment per càmera

Aquest mètode es basa en els software de reconeixement visual. Es processen les imatges que venen d'una càmera i es fa un seguiment del moviment de l'objecte que es vol seguir. Això és bastant usat pel control del tràfic en certs encreuaments a les ciutats grans, i així poder controlar-lo millor segons el moviment que hi hagi i aplicar les mesures necessàries per millorar la fluïdesa d'aquest.

Avantatge: si es programa adequadament és bastant precís i fa un seguiment exacte de la posició del mòbil.

Desavantatges: la visió artificial encara és una mica rudimentària, l'afecten molt els canvis de il·luminació. Són sistemes principalment fixes, i per seguir el moviment del robot seria necessària una càmera enfocant en tot moment al robot i un complex algoritme per que

puguis recalcular la distància que hi ha en cada moment. I si es vol que el sistema sigui autònom no podem fer que una càmera la segueixi a tots llocs.

Seguiment per balises

Aquest mètode es bassa en la col·locació de balises pel trajecte del robot i per infrarojos i ultrasons, o altres tipus de comunicació inalàmbric, aquestes informen d'on està posicionat el robot quan aquest està proper a alguna d'aquestes balises. Aquest mètode és usualment combinat amb algun dels anteriors. També es pot fer per triangular la posició del robot com si fos un GPS.

Desavantatges: Es necessita un trajecte preestablert o un espai tancat.

GPS

El GPS o *Global Positioning System*, és un sistema de navegació per satèl·lit que permet saber la posició del mòbil amb una precisió d'uns metres en gairebé qualsevol lloc de la Terra.

Per saber la posició de l'objecte es fa la triangulació utilitzant 3 satèl·lits.

Avantatges: Resulta bastant fiable en distancies grans.

Desavantatges: Si es vol utilitzar per seguir moviments petits(<3metres) resulta poc precís. El temps de mostreig és bastant gran ±1 segon o més, si es necessiten grans prestacions pot resultar bastant molest. I òbviament necessita tota un xarxa de satèl·lits.

10.Fusió de mètodes

Ja que tots els mètodes anteriors tenen els seus avantatges i els seus defectes, una solució pot ser la fusió de diferents mètodes. Així aconseguim cobrir on alguns fallen amb les fortaleses d'altres.

Estimació per Kalman

Un dels mètodes més utilitzats per dur a terme aquest procés és a través d'un "Kalman Filter", també conegut com a *Linear Quadratic Estimation*. El procés que es duu a terme en un filtre Kalman és el que veiem a continuació:

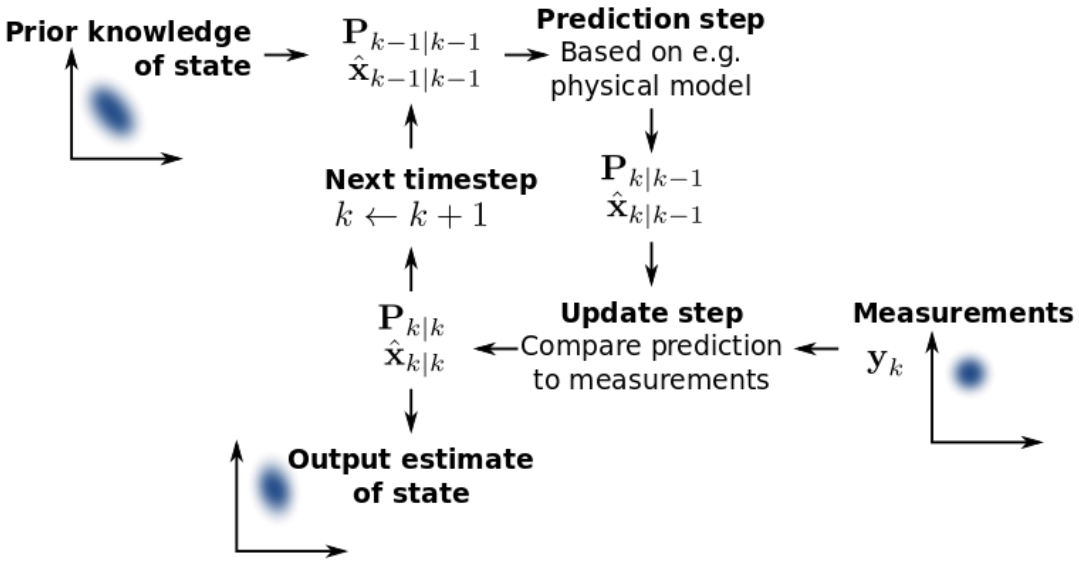


Figura 38 Explicació esquematitzada del funcionament del filtre Kalman: http://en.wikipedia.org/wiki/Kalman_filter.

Com es pot veure, és un mètode recursiu que intenta aproximar el valor present del model intentant predir quin serà l'estat futur, posteriorment comparant amb les mesures reals i anar regulant l'error per cada vegada intentar obtenir mesures més acurades.

Pel filtratge, encara que no requereix que el soroll sigui de tipus Gaussià, el seu funcionament serà molt millor en el cas que ho sigui.

Tot i que s'anomena filtre, pot ser usat per altres coses apart de filtrar senyals, una d'elles és fusionar dos models, normalment és un teòric i l'altre es basa en les mesures reals obtingues.

També pot ser usat per combinar sensors iguals, com per exemple 3 sensors que mesurin la temperatura d'una caldera, amb aquest mètode podem obtenir una mesura on eliminem el soroll dels sensors individuals i els seus errors deguts a les seves característiques de construcció que puguin tenir cada un, i obtenir una mesura totalment acurada de la temperatura real.

Un exemple de la fusió de mètodes, és l'estimació de la càrrega d'una bateria, es combina el model teòric de la bateria amb l'entrada de corrent, i es compara amb el voltatge que treu la bateria, d'aquesta manera es pot evitar l'error d'acumulació durant la integració del corrent i fer aproximacions sense saber l'estat inicial de la bateria.

Ara passem a la explicació del procés:

x[n] → vector d'estats actual.

x[n+1] → vector d'estats futur.

$\bar{x}[n+1]$ → vector d'estat estimat.

u[n] → vector d'entrades.

y[n] → vector de mesures.

A → matriu dinàmica.

B → matriu de control.

C → matriu d'observació.

w → error de procés.

n → error de mesura.

Q → matriu de covariància en l'error de procés.

R → matriu de covariància en l'error de mesura.

$L[n+1]$ → guany de Kalman.

$P[n]$ → error de covariància.

Tenim un sistema així:

Predicció d'estat

$$\bar{x} = Ax + Bu + w$$

Equació 47

Predicció del sensor

$$\bar{y} = C\bar{x} + n$$

Equació 48

Tindrem dos tipus d'errors: el de procés i el de mesura, aquest mètode utilitza la iteració durant el temps per anar regulant certs paràmetres interns per intentar fer més acurada l'estimació i tractar d'eliminar aquests errors.

El filtre a cada iteració realitza els següents passos.

1. Estimació del vector d'estats futur:

$$\bar{x}[n+1] = Ax[n] + Bu[n]$$

Equació 49

2. Propagació de l'error de covariància:

$$\bar{P}[n+1] = A * P[n] * A^T + Q$$

Equació 50

3.Calcul del guany:

$$L[n+1] = \bar{P}[n+1] * C^T (C * \bar{P}[n+1] * C^T + R)^{-1}$$

Equació 51

4.Estimacio de la mesura:

$$\bar{y}[n] = C * \bar{x}[n+1]$$

Equació 52

5.Calcul del vector d'estat:

$$x[n+1] = \bar{x}[n+1] + L[n+1](y[n] - \bar{y}[n])$$

Equació 53

6.Correció de l'error:

$$P[n+1] = (I_n - L[n+1] * C) * \bar{P}[n+1]$$

Equació 54

El primer que fa a partir de la combinació de les entrades i l'estat anterior, és una predicció de quin ha de ser el següent estat(Equació 49), posteriorment es troba la nova matriu de l'error de covariància(Equació 50) amb l'error del procés Q i la matriu dinàmica. Seguidament es troba l'anomenat guany de Kalman amb l'error de la mesura R i la matriu d'observació(Equació 51). Estima quina hauria de ser la mesura segons l'estat predit i les entrades(Equació 52). I amb l'error de la mesura estimada i la real mesurada pel guany calculat trobem el nou vector d'estats(Equació 53). I per últim es fa l'actualització de la matriu d'error de covariància(Equació 54) que com més iteracions hi hagi més precisa serà.

Tipus de fusions

Per poder trobar quina combinació em donava millor resultat vaig proposar diferents mètodes i fent proves vaig esbrinar quin era el més útil i quin era el que donava millors resultats.

Acceleròmetre + Encoders

El primer tipus és la combinació dels sensors inercials i els encoders, l'objectiu és utilitzar l'acceleració obtinguda amb els sensors inercials; introduir-la en el model cinemàtic del filtre i comparar aquests resultats amb la velocitat lineal i la posició obtinguda pels encoders.

Teòric + Acceleròmetre

En aquest farem una combinació amb el model teòric, les dades introduïdes als motors, i les dades obtingudes a través dels sensors inercials.

Acceleròmetre + GPS

Un dels mètodes més usats és combinar els sensors inercials, imprecisos i sorollosos, amb els sistema GPS, precís però lent, fent el model amb les dades de l'acceleració i creuant-les amb les de posició del GPS. Aquest mètode tot i ser el més fiable, malauradament no tinc la possibilitat de posar-lo en prova al no tenir disponible cap sistema GPS, i la impossibilitat de muntar un sistema GPS en el temps necessari.

11.Proves experimentals

Explicació de les proves i l'entorn de programació

Blocs usats

En aquest apartat faré una petita explicació de quins blocs he usat, de com funcionen i perquè he triat aquest i no un altre, així posteriorment quan faci la explicació del seu funcionament conjunt, ja no hagi de fer la explicació de la funció de cada bloc, ja que la majoria seran combinacions d'aquests.

Començaré per els blocs de la llibreria de Lego;

Figura 39 Bloc timer

Timer:

Aquest en retornarà el temps en mil·lisegons del robot de Lego, el temps de mostreig com en tots els altres blocs pot ser escollit lliurement.

Figura 40 Bloc motor

Motor:

Aquest en permet donar comandes als motors connectats al robot, escollin el port corresponent, i els valors van de 0 a 100.

Figura 41 Bloc giroscopi

Giroscopi:

Ens retorna el valor del giroscopi en radiants/segon, el temps de mostreig pot ser escollit segons les necessitats i els límits del sensor.

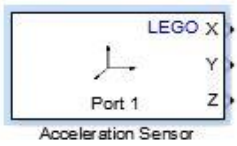


Figura 42 Bloc acceleròmetre

Acceleròmetre:

El bloc de l'acceleròmetre té 3 sortides, una per a cada eix, els valors van de -255 fins 255, per a fer la conversió es mira que el valors de la gravetat és d'uns 200, llavors si fem la conversió queda que cada increment és 0,04903325m^2/s.

Figura 43 Bloc encoder

Encoder:

Aquest ens proporcionarà en comptes el valor de rotació del motor, tal com ja he explicat en l'apartat corresponent.

Integració:

Per fer la integració tenim varies opcions, descartem totes les de temps continu, ja que com he dit, treballarem en discret.

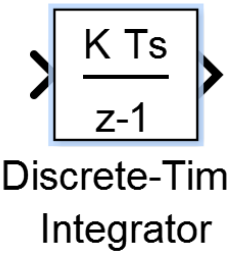


Figura 44 Bloc integrador discret

Aquest bloc ens permetrà fer la integració de totes les dades que entrin, definint prèviament el temps de mostreig d'aquestes dades.

Dins d'aquest hi ha diferents opcions de realitzar això;

- Forward Euler
- Backward Euler
- Trapezoïdal

Totes aquestes són diferents maneres de realitzar la integració, la de Euler fa la bàsica, que aprenem a Batxillerat, d'integrar com si sota la corba hi hagués un rectangle, la diferència entre la Forward i la Backward és que una agafa el valor anterior i l'altra el posterior, en el moment d'escollir "l'altura" del rectangle. La trapezoïdal considera que el rectangle no omple completament l'àrea sota la corba i cobreix el espai buit con si fos un triangle. En les següents fotografies podem veure la diferencia.

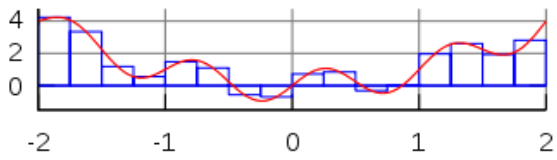


Figura 45 Exemple d'integració pel mètode Euler

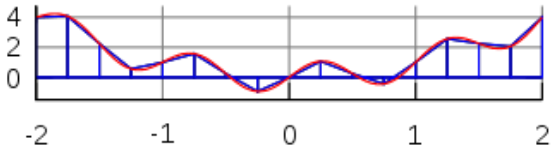


Figura 46 Exemple integració pel mètode Trapezoïdal

I per veure encara més l'error que pot donar això, he realitzat un petit experiment amb el Matlab per poder veure les diferències, he escollit la funció d'una rampa que al integrar-la queda com una corba parabòlica. En 10 segons creix 10 unitats, l'àrea final seria de 50 unitats.

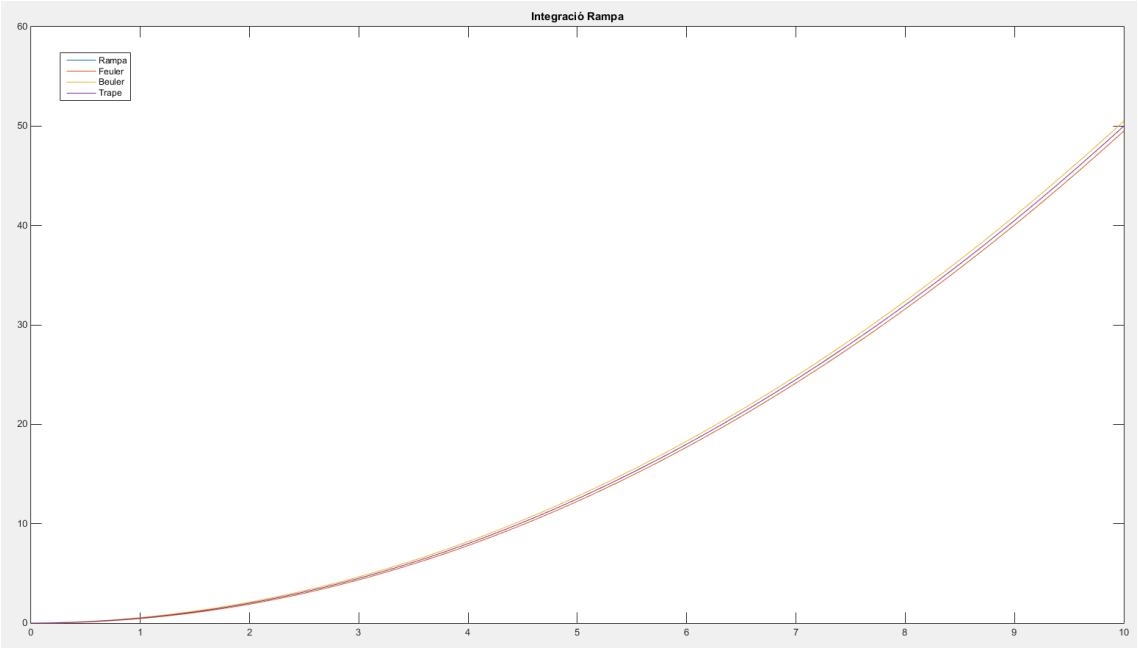


Figura 47 Resultats de la integració d'una rampa

Aquí es pot observar com les integracions per Euler van acumulant errors a causa de la seva manera de calcular. Com més gran sigui el temps de mostreig, més gran serà l'error, i com més complicada sigui la corba, més errors acumularan.

En la següent imatge podem veure com aquests errors creixen:

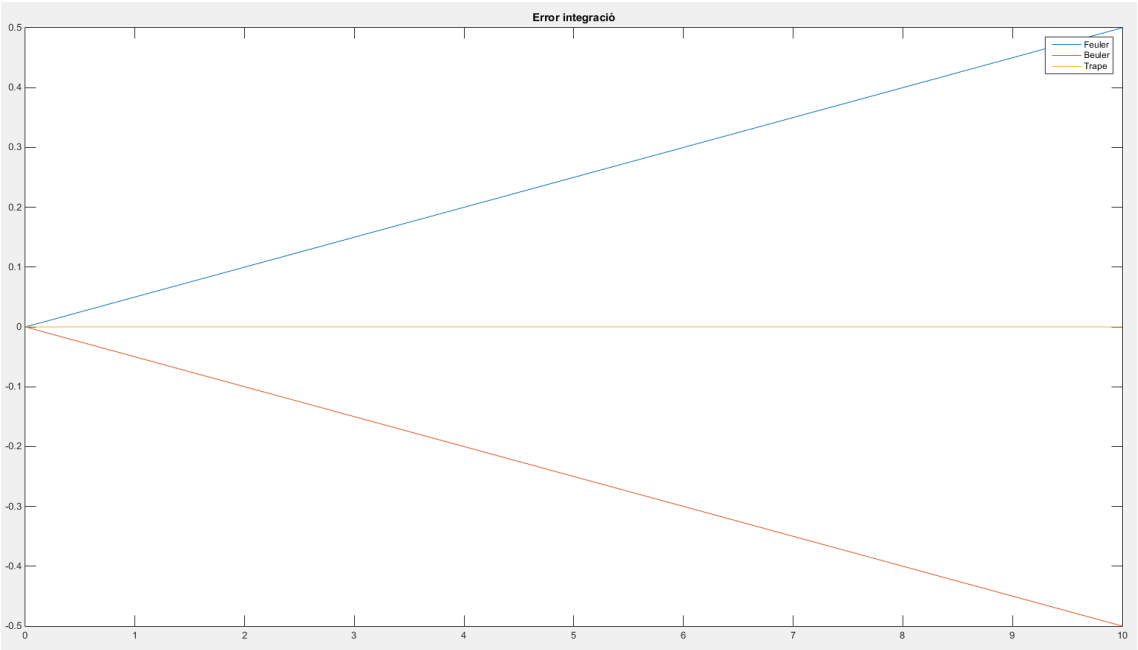
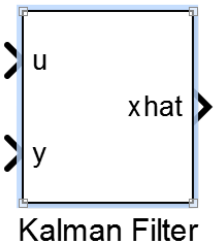


Figura 48 Errors en la integració

Per aquesta raó he escollit la integració trapezoïdal.

Kalman filtre:



Aquest bloc fa la funció del *Kalman filtre*, que anteriorment ja em explicat. Les entrades seran les dades necessàries per a cada cas, i també es pot activar la entrada de els matrius com a línies de dades i una entrada de reset. Per a les sortides tenim per defecte l'estimació de l'espai d'estats, però també es pot treure l'estimació de les mesures i l'error de la matriu P si volem més informació.

Figura 49 Bloc Kalman filter

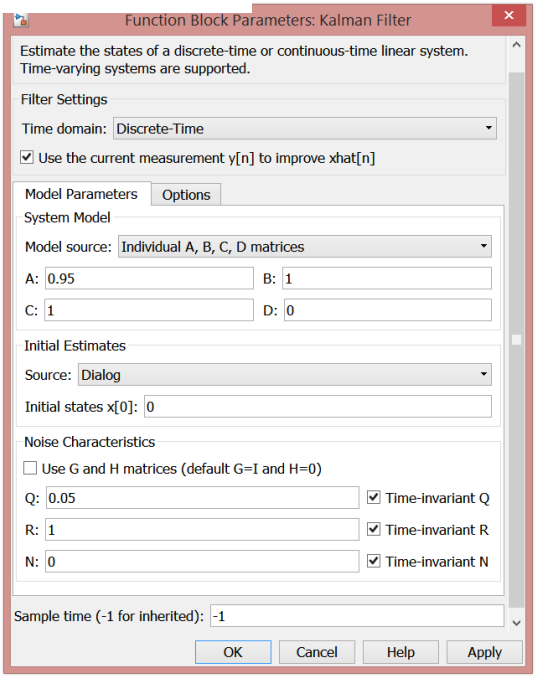


Figura 50 Captura de la finestra de configuració del bloc Kalman filter

Aquests paràmetres poden ser introduïts de diferent manera: al principi del programa manualment com esta per defecte, o bé per ports d'entrada i introduint el valors des del sistema de blocs del Simulink, això permet variar els valors de les matrius durant l'execució per fer models més complexos.

Com es pot veure tindrem les matrius A, B, C i D, que correspondran a l'espai d'estats, les Q i R que són les de covariància de l'error.

També es pot escollir el valor inicial de l'espai d'estat, i si introduíssim les matrius A,B,C i D de manera externa ens demanaria valor inicial de la matriu P.

Depenent de les dimensions de les matrius amb que es treballi, s'hauran d'introduir les entrades en forma de vector columna de diferents dimensions. Els resultats també sortiran d'aquesta manera, així que s'han d'usar mètodes per compactar les línies de connexió.

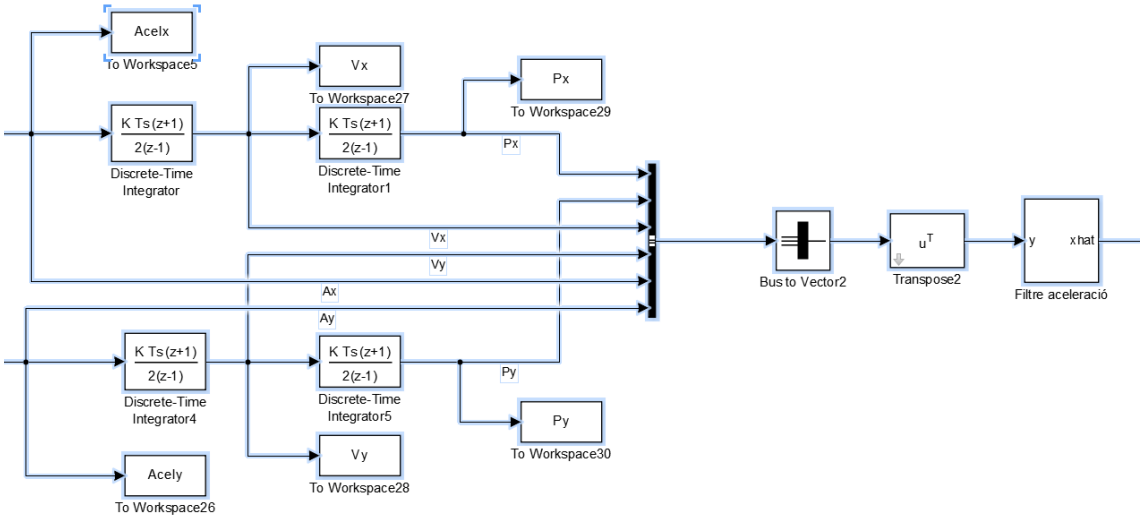


Figura 51 Captura d'un exemple de compactació de línies de dades

Aquí tenim un exemple d'això, per poder introduir totes les variables primer em de convertir-ho tot en un bus d'informació, això funciona com un *buffer* que el que farà és enviar paquets de totes les dades que li arribin, i convertirem aquests paquets a vectors i posteriorment fer la transposada per que ens quedi un vector columna per a poder introduir en el bloc del Kalman.

Per a poder descomprimir les dades utilitzarem un bloc anomenat *Demux*, que s'encarrega de separar un bus de comunicació en línies individuals per a poder ser tractades separadament.

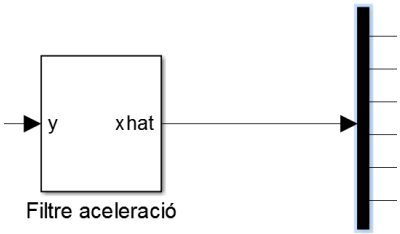


Figura 52 Captura de un Demux

Per poder establir la connexió entre els diferents models de Simulink faig us del bloc per poder guardar dades al *Workspace*, i guardar-les com una estructura, on hi haurà al mateix moment el temps i altres paràmetres per poder fer la sincronització.

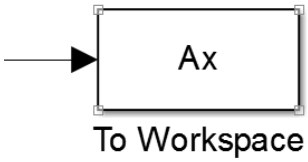


Figura 53 Bloc To Workspace

Posteriorment per poder processar les dades guardades al *Workspace*, utilitzem el bloc següent i el configurem per tal que rebi dades de tipus estructura, al haver guardat el temps ens permet simular realment com si estiguéssim rebent les dades en temps real.

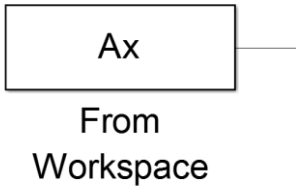


Figura 54 Bloc From Workspace

A part d'aquests també utilitzaré els blocs més comuns com els guanys, sumadors, productes, funcions trigonomètriques, etc. Si es vol més informació, es pot trobar en el lloc web d'ajuda del programa Matlab.

Evolució de les proves

Aquí faré un petit recull del procés que he fet fins arribar a la solució final i quins problemes he anat trobant, i també perquè he escollit aquestes solucions.

La idea principal era realitzar el seguiment dels robots a través de tres mètodes: sensors inercials, encoders i teòricament.

El dos mètodes, encoder i teòric, van resultar fàcils d'implementar i provar sobre el robot, ja que són mètodes simples i bastants usats; però el de sensors inercials va començar a donar diversos problemes durant la seva implementació. El primer i més important va ser el soroll, els sensors d'acceleració són bastant sorollosos, i l'ús d'ells sobre un sistema on les acceleracions en comparació amb el soroll, són bastant petites, encara va complicar més el problema. El segon, el procés d'integració, com ja he explicat anteriorment la integració és un procés on el petit error es va acumulant, com s'han de fer 2 integracions el problema encara s'amplia més.

Les primeres solucions van ser enfocades a solucionar el *bias* o *offset* del sensors, qualsevol sensor té un valor constant que hauria de ser el nostre 0 de refèrencia, però per diferents raons no és 0: el voltatge, l'alineació, la temperatura i molts altres factors que influeixen. Per solucionar aquesta diferència s'han de realitzar diversos ajustos.

En els acceleròmetres el més important pot ser l'alineació amb la gravetat de la Terra. Idealment el sensor hauria d'estar amb el eix Z totalment coincident amb el camp gravitatori de la terra, però com és gairebé impossible en valors tant petits, aquesta gravetat afecta els altres eixos del sensor i creen un *offset*.

Per solucionar això inicialment vaig usar el mètode de mostreig, que es basa en recollir dades dels sensors en repòs durant un temps i obtenir aquest valor de *bias* i restar-lo per obtenir el valor real que volem.

Seguidament em vaig centrar en el problema del soroll, vaig realitzar proves amb diferents tipus de filtratge, FIR, passabaix, passabanda, Butterworth, però cap em donava els resultats adequats. I estudiant com ho solucionaven la resta d'investigadors vaig trobar el filtre Kalman, vaig poder observar com aquest mètode era utilitzat per l'estimació de models i filtratge de senyals. Llavors vaig decidir intentar implantar-lo al meu treball.

Després de fer la deguda recerca i l'estudi del seu funcionament, vaig comprovar que era la solució més adient pel meu problema, i els resultats obtinguts demostren aquesta conclusió.

Em vaig decidir a implementar aquest mètode sobre el robot, però ràpidament van començar a sorgir problemes en la comunicació amb el robot. El meu objectiu inicial era muntar sobre el robot tot el programa, i deixar al Matlab només la recollida dels resultats i la seva posterior presentació pel treball. Però el robot no tenia la suficient potència de càlcul o la compilació del codi no era la suficientment optimitzada per tal que el robot pogués executar el programa a temps real i al mateix moment, mantenir la connexió amb l'ordinador i el Matlab.

A partir d'aquí vaig haver de pensar una altra solució, i vaig arribar a la conclusió de separar el procés. La part de recollir dades només la realitzaria el robot, i la part del processament de les dades les realitzaria el meu ordinador a posteriori. Això va suposar un millora del procés ja que anteriorment cada vegada que modificava algun paràmetre o volia fer canvis, havia de tornar a carregar tot el programa al robot, un procés lent i tediós, i ara podia fer múltiples canvis a les mateixes dades i veure els diferents resultats de manera ràpida.

I encara més, ara podia ampliar i recarregar el processat de les dades amb diferents mètodes i més complexes, sense patir que el robot no pogués seguir el temps de mostreig.

Per aquesta raó vaig decidir incloure primer la estimació del bias a través d'un Kalman, i així poder fer ajustaments dinàmics durant tot l'experiment. I també incloure diferents fusions i mètodes per tenir més opcions de comparació.

Després de tot aquest procés he arribat al model final on puc realitzar una gran varietat de mètodes per a poder-los comparar, però ara no puc aplicar una idea original que era trobar un mètode que funcionés per poder utilitzar-lo per a realimentar l'entrada de control i que aquest fos més precís, per fer això s'hauria de muntar una connexió amb el robot més complicada, així que aquesta idea ha hagut de ser descartada tot i ser una de les primeres que van sorgir.

Expilació conjunt de blocs

Com ja he explicat els experiments estan dividits en dos parts, primerament l'adquisició de dades i després el seu processament.

Primerament faré l'explicació del model de simulink d'obtenció de dades, aquest està format per la part que controla la potència dels motors i el sensors. La primera part està composada d'un bloc rampa o esglaó que pot obtenir valors de 0 a 100 i va directe als blocs de control dels motors, aquesta part és la que variarà a cada experiment, ja que és la que controla les ordres de moviment, i la utilitzarem per poder realitzar les diferents proves.

El procediment més comú és mantenir el robot sense moviment durant uns segons per que els sistemes dinàmics ajustin els paràmetres i després començar el moviment designat per aquella prova. Per aquesta raó, he escollit que no se li doni potència al motor fins que passi un temps d'espera designat.

Els sensors van directe al bloc de guardat de dades en el Workspace, per posteriorment poder ser importades a l'altre model.

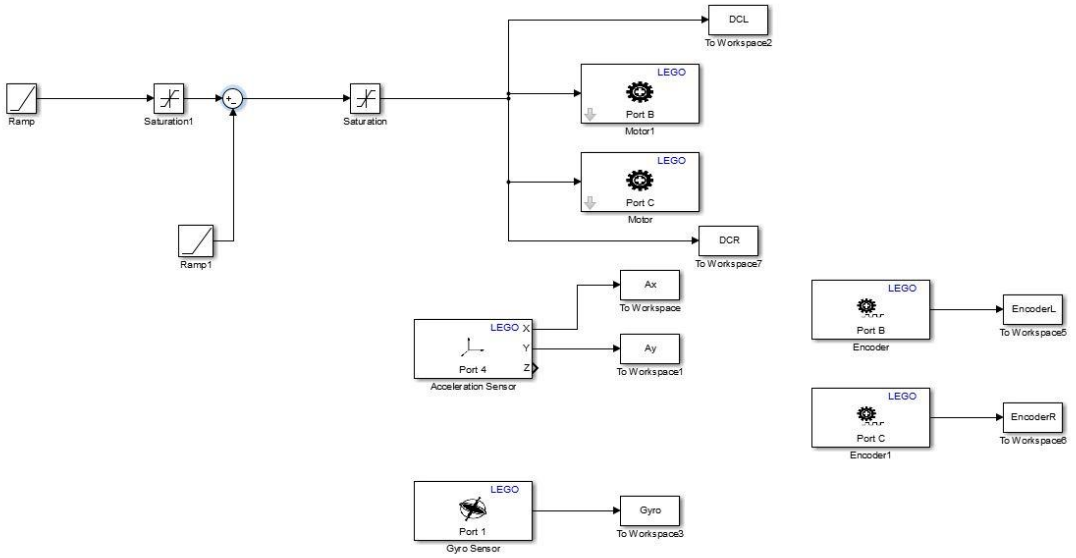


Figura 55 Captura del model de recollida de dades

En el segon model de Simulink, que és el processament de dades tenim moltes parts, intentaré fer una expilació ordenada seguint l'ordre que segueixen les dades.

Primerament tenim els blocs que introdueixen les dades al model, que provenen del Workspace. Els sensors inercials, acceleròmetre i giroscopi tenen el seu sistema de correcció de bias, primerament un valor base que ha estat trobat per observació i una altra part dinàmica que es calculada cada vegada que es posa en funcionament el model.

Per al cas dels acceleròmetres tenim el Kalman que s'encarrega d'això; el seu model és una petita integració de primer nivell per poder compensar el drift del sensor a mesura que avança el temps, controlat per un coeficient, aquest coeficient m'ha donat grans problemes ja que a causa de diferents condicions cada execució varia, finalment he aconseguit establir que els millors resultats es troben al voltant de 0.2 i 0.45.

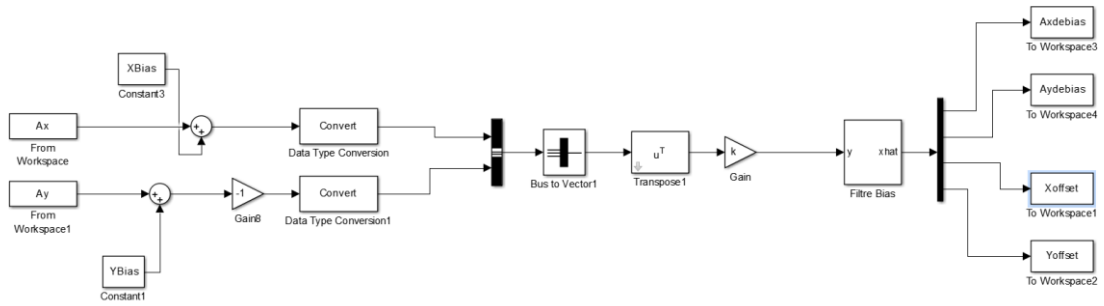


Figura 56 Captura del model simulink

Com podem veure tenim les entrades del Workspace que els hi sumem un *bias* obtingut a través dels experiments anteriors. Fem un canvi de signe l'eix de les y, ja que l'eix del sensors és del sentit contrari del nostre eix base. Ho composem en un vector columna i ho

introduïm al kalman i d'ell obtenim l'acceleració sense el bias, trobat a través de l'estimació. Els paràmetres són els següents:

$$y = \begin{bmatrix} A_x \\ A_y \end{bmatrix}$$

Equació 55

$$x = \begin{bmatrix} a_x \\ a_y \\ Xoffset \\ Yoffset \end{bmatrix}$$

Equació 56

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ ts & 0 & 1 & 0 \\ 0 & ts & 0 & 1 \end{bmatrix}$$

Equació 57

$$B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Equació 58

$$C = \begin{bmatrix} 1 & 0 & coef & 0 \\ 0 & 1 & 0 & coef \end{bmatrix}$$

Equació 59

$$Q = A * A'$$

Equació 60

$$R = 0.001 * C * C'$$

Equació 61

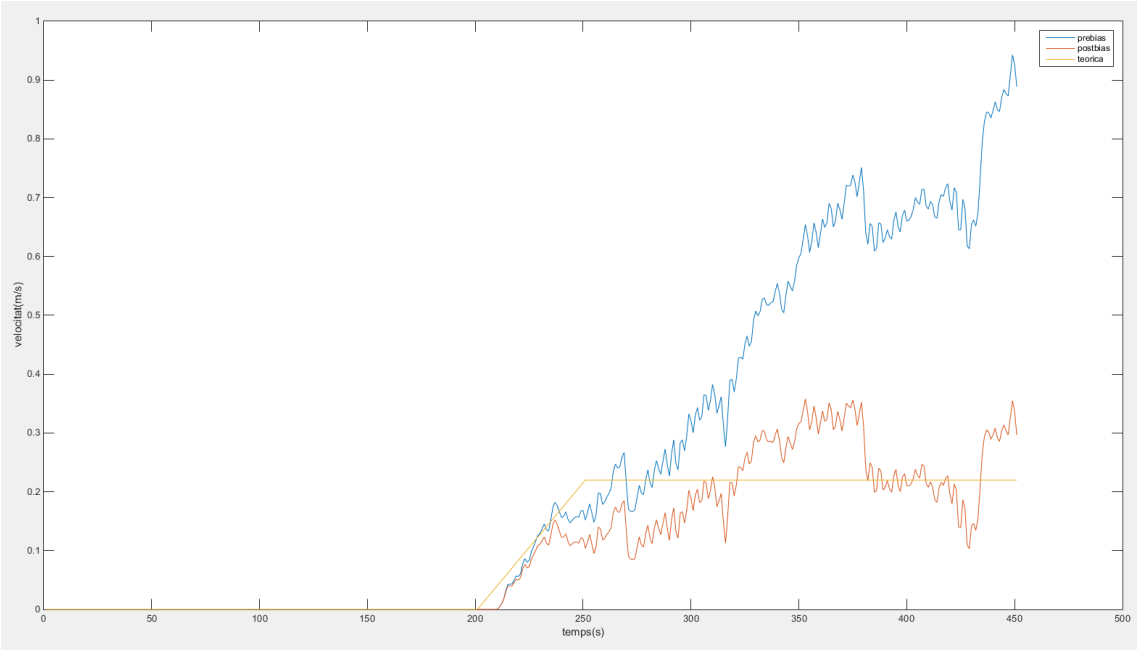


Figura 57 Exemple del bias dinamic

Aquí es pot observar l'efecte del procediment. La groga és la teòrica, la taronja és ls valors que em processat i la blava són els valors que no han passat per el filtre. Podem observar fàcilment la tendència creixent si no apliquéssim aquest mètode.

Pel giroscopi he realitzat una simple integració dels valors durant un temps, per després fer la divisió pel nombre de mostres amb la finalitat d'obtenir una mitja i poder restar al valor que obtenim del sensor i així calibrar.

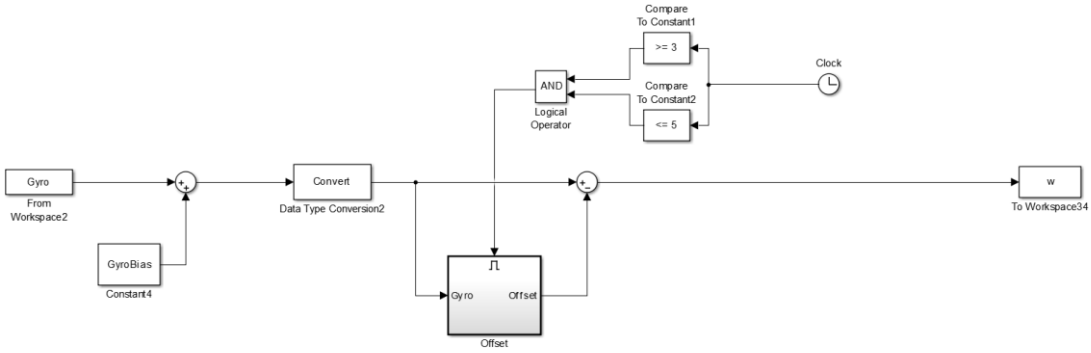


Figura 58 Captura del model simulink

El bloc de càlcul del offset està controlat per temps, només funcionarà durant 2 segons, entre el 3 i 5. L'espera de 3 segons és degut a que el robot tarda uns segons en establir la connexió bluetooth i a vegades les a dades abans dels 3 segons no són registrades. I 2 segons d'espera per poder recollir suficients dades per tal de tenir una aproximació lo suficientment acurada.

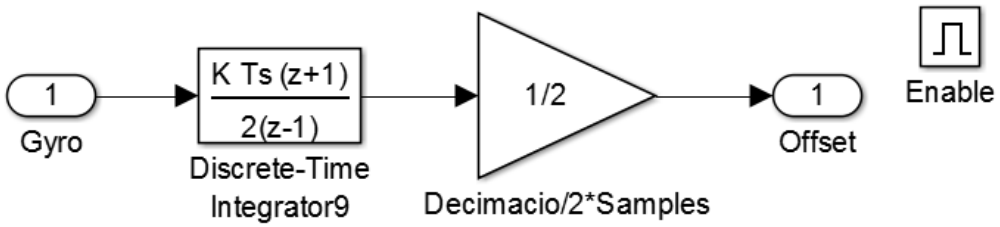


Figura 59 Captura del model simulink

Dins del bloc per fer la mitja tenim un integrador i després fem:

$$Mitjana = \frac{\sum_0^n x}{n}$$

Equació 62

Com el integrador aplica una decimació que és 1/ts i el nombre de mostres es calcula multiplicant el temps que es prenen les mostres per la freqüència de mostreig:

$$n = t * \frac{1}{ts}$$

Equació 63

$$Mitjana = \frac{\sum_0^n x * \frac{1}{ts}}{t * \frac{1}{ts}} = \frac{\sum_0^n x}{2}$$

Equació 64

D'aquí obtenim el guany de 1/2, un valor que no dependrà del temps de mostreig.

Seguidament faig la integració del valor del giroscopi per obtenir el valor de l'angle del robot. Així poder realitzar la rotació dels eixos de les acceleracions

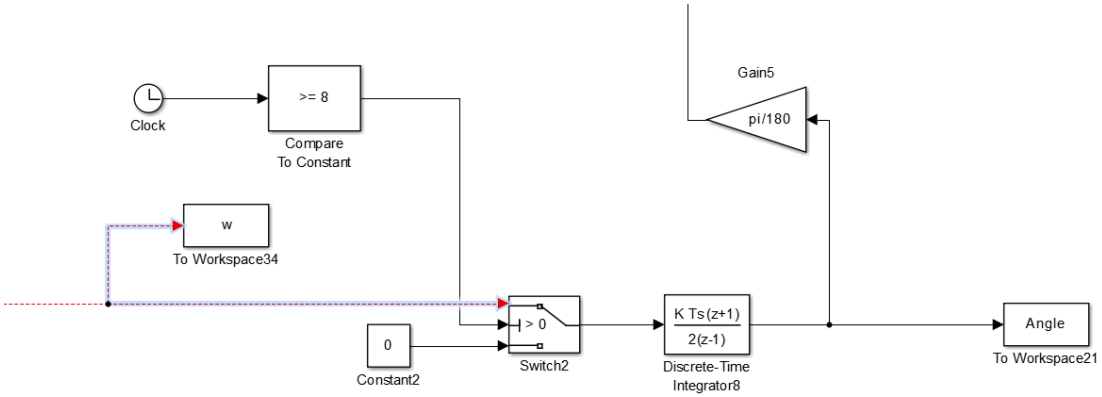


Figura 60 Captura del model simulink

Tenim el *switch*, que controla que quan el temps sigui igual a 8 segons, comenci a processar les dades, i fem la integració. Hi ha un guany per passar de graus a radiants, ja que els càlculs dels models es fan en radiants.

I per la part de l'acceleròmetre, com ja he dit anteriorment jo només utilitzaré els eixos x i y de l'acceleròmetre, per poder fer la conversió als eixos cartesians bases he de fer el mètode conegut com rotacions de Euler, que són unes equacions que defineixen les projeccions dels eixos rotats sobre els eixos bases segons la rotació d'aquests.

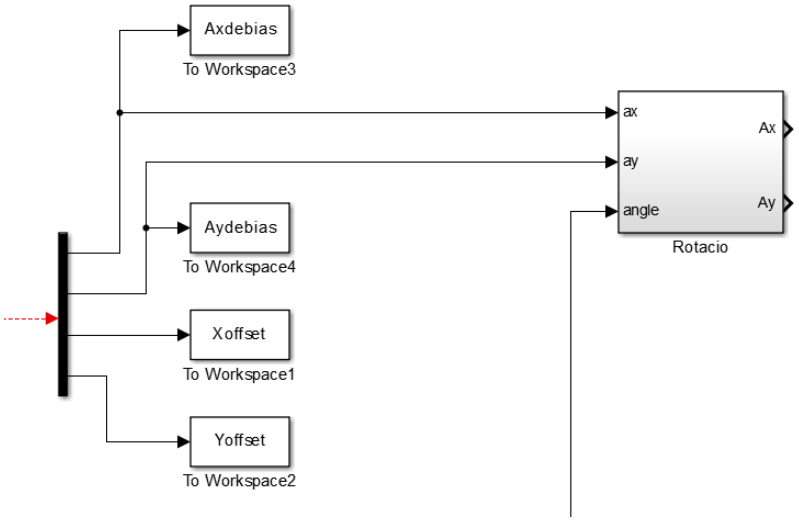


Figura 61 Captura del model Simulink

Dins del bloc Rotació tenim:

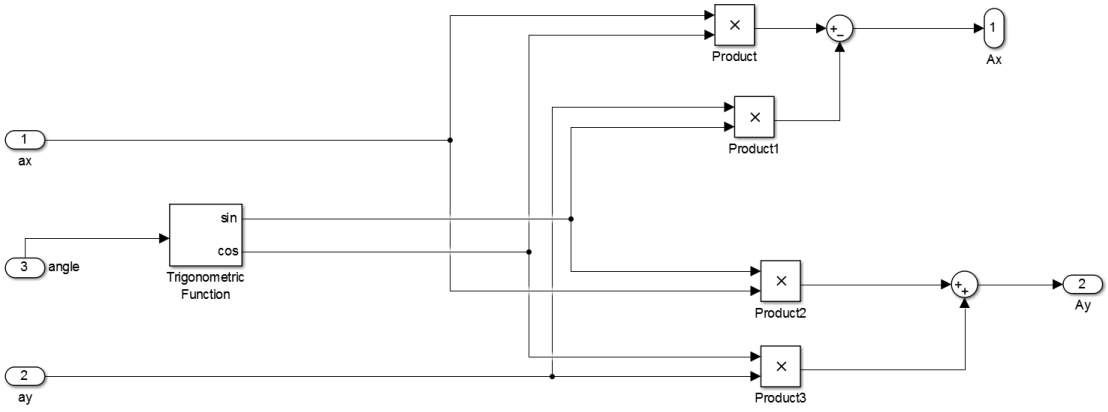


Figura 62 Captura del model simulink

Que no és més que l'aplicació de les formules de rotació que són:

$$A_x = a_x * cos\alpha - a_y * sen\alpha$$

Equació 65

$$A_y = a_x * sin\alpha + a_y * cos\alpha$$

Equació 66

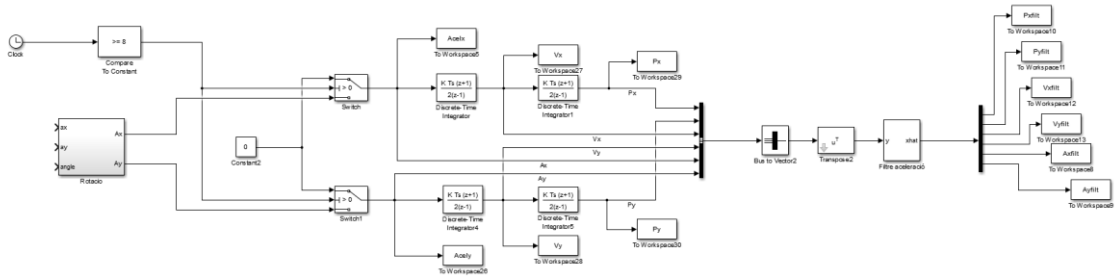


Figura 63 Captura del model Simulink

Quan ja he obtingut l'acceleració en cada eix, faig la doble integració i la compactació en un bus, per tal de introduir-los al filtre kalman. Aquest realitzarà el filtratge de l'acceleració a través del mètode anteriorment explicat. Els paràmetres resultants són aquests:

$$y = \begin{bmatrix} p_x \\ p_y \\ v_x \\ v_y \\ a_x \\ a_y \end{bmatrix}$$

Equació 67

$$x = \begin{bmatrix} p_{xfilt} \\ p_{yfilt} \\ v_{xfilt} \\ v_{yfilt} \\ a_{xfilt} \\ a_{yfilt} \end{bmatrix}$$

Equació 68

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 & \frac{ts^2}{2} & 0 \\ 0 & 1 & 0 & ts & 0 & \frac{ts^2}{2} \\ 0 & 0 & 1 & 0 & ts & 0 \\ 0 & 0 & 0 & 1 & 0 & ts \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Equació 69

$$B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Equació 70

$$C = \mathbb{I}(6x6)$$

Equació 71

Llavors els valors de Q i R s'han trobat a través d'experiments i tunning per aproximacions segons quins obtenien millors resultats.

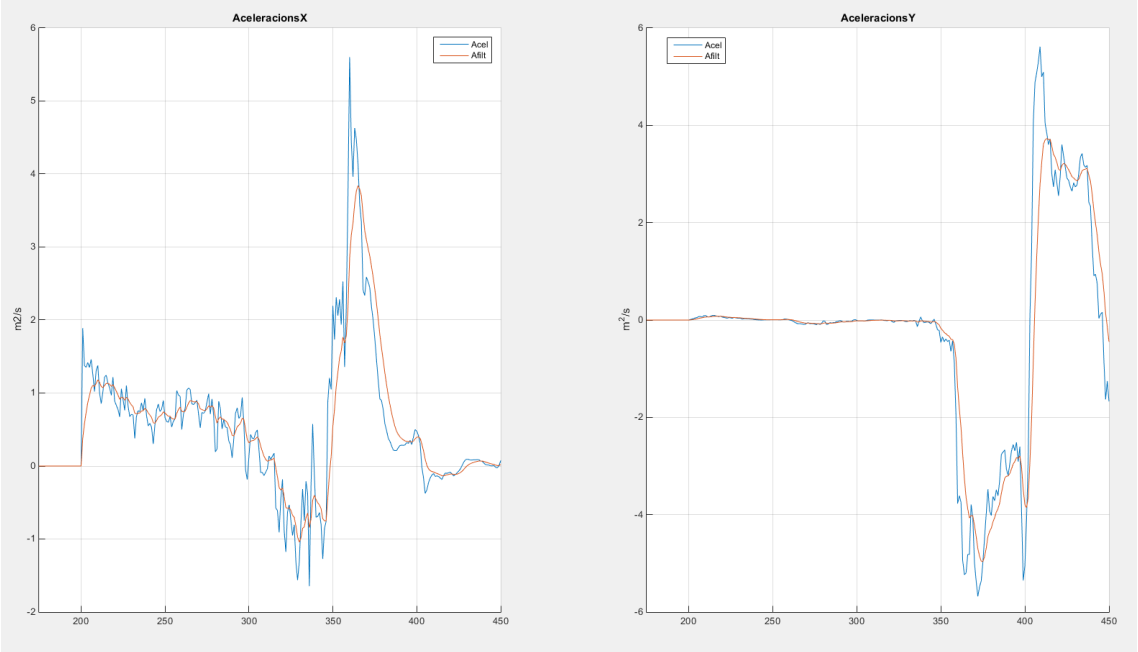
$$Q = 0.001 * A * A'$$

Equació 72

$$R = 0.1 * C * C'$$

Equació 73

Aquí podem observar un exemple del funcionament amb un moviment aleatori del sensor per poder comprovar els resultats d'aquest filtratge:



Per la part dels encoders, el que fem és la derivació, per obtenir la velocitat angular de cada roda les introduïm al bloc del model diferencial per obtenir la posició i la velocitat del robot respecte els eixos.

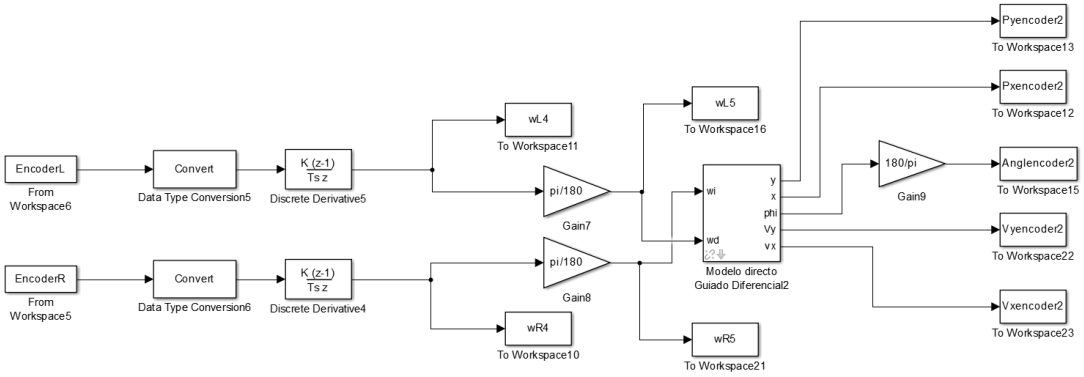


Figura 64 Captura del model simulink

Dins del bloc s'apliquen aquestes equacions obtingudes de les equacions de restricció del model diferencial. Que relacionen el gir de les rodes amb el moviment del robot.

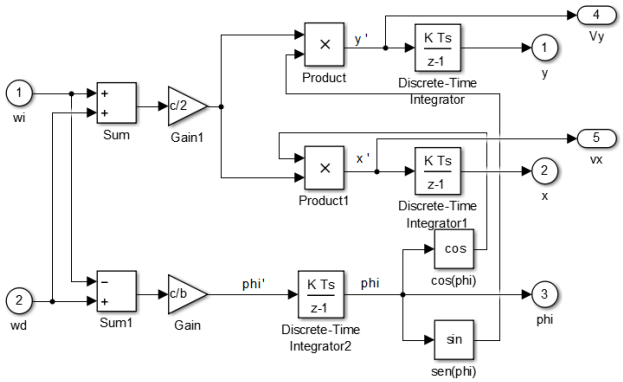


Figura 65 Captura del model simulink

Les equacions que corresponen a aquest diagrama:

$$Vx = \frac{r(wl + wd)}{2} * \cos\varphi$$

Equació 74

$$Vy = \frac{r(wl + wd)}{2} * \sin\varphi$$

Equació 75

$$\omega = \frac{r(wd - wl)}{b}$$

Equació 76

On la r és el radi de la roda i la b és la distancia que hi ha entre les rodes. Són les equacions estretes de l'apartat anterior.

També tenim el model teòric, amb els valors de potència introduïts als motors i el valor de conversió obtingut experimentalment, obtenim la velocitat lineal de cada roda, i les introduïm al model diferencial, però a diferencia de l'anterior aquí tenim la velocitat lineal de la roda no la angular, per aquesta raó les equacions no són les mateixes.

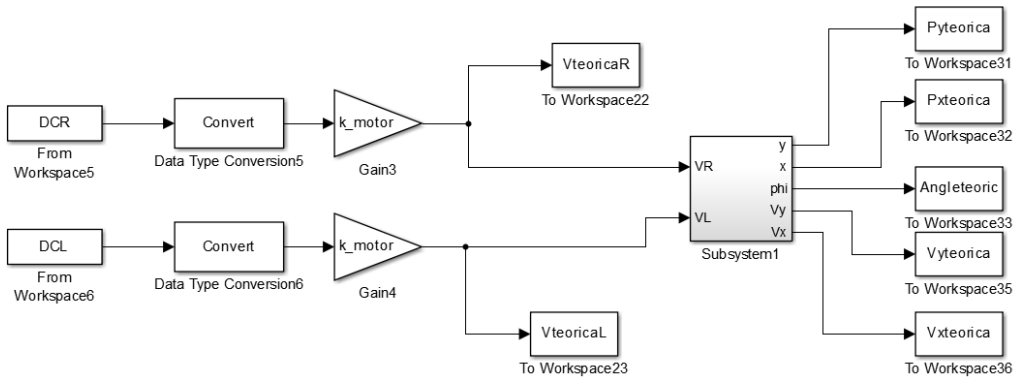


Figura 66 Captura del model simulink

Quan ja tenim les dades processades de cada opció, procedim a realitzar les fusions dels diferents mètodes, amb el filtre Kalman, seguidament tenim les equacions usades a cada cas:

Inercial + encoder

En aquest fusionem les acceleracions filtrades dels sensors inercials amb la velocitat i posició obtinguda del model del encoder.

Les connexions quedaran d'aquest manera:

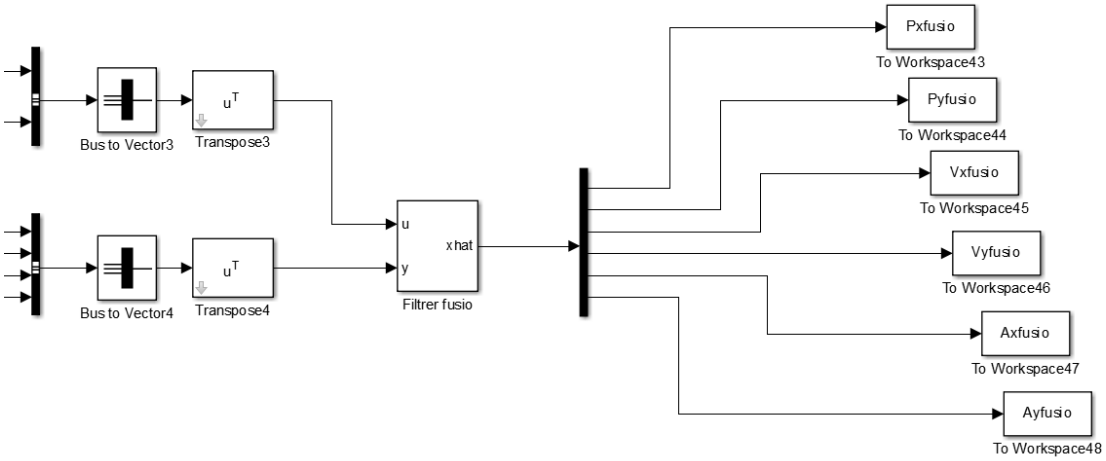


Figura 67 Captura del model Simulink

I els paràmetres són aquests:

$$y = \begin{bmatrix} Vencox \\ Vencoy \\ Pencox \\ Vencoy \end{bmatrix}$$

Equació 77

$$x = \begin{bmatrix} p_x \\ p_y \\ v_x \\ v_y \\ a_x \\ a_y \end{bmatrix}$$

Equació 78

$$A = \begin{bmatrix} 1 & 0 & ts & 0 & 0 & 0 \\ 0 & 1 & 0 & ts & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Equació 79

$$B = \begin{bmatrix} \frac{ts^2}{2} & 0 \\ 0 & \frac{ts^2}{2} \\ ts & 0 \\ 0 & ts \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Equació 80

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Equació 81

Per a la Q i la R ho he fet de la mateixa manera que en el filtratge de l'acceleració per experimentació. Els resultats més prometedors ha estat amb valors de:

$$Q = 0{,}001 * \mathrm{A} * \mathrm{A}'$$

Equació 82

$$R = 125 * C * C'$$

Equació 83

Inercial + teòric

Té la mateixa estructura que l'anterior i els mateixos paràmetres, però les entrades de les mesures en aquest cas, seran els valors obtinguts del model teòric.

Metodologia seguida per fer les proves

Per a realitzar les proves, primer de tot he fet l'adquisició de dades i he utilitzat el codi d'un sistema de guardat de les proves experimentals que està explicat a l'annex.

Posteriorment, després de cada tanda de 5 proves he fet el processament d'aquestes dades conservant els mateixos paràmetres, coeficients i valors de les matrius dels Kalman.

Cada tanda de proves tindrà una referència al nom de la carpeta per si es vol consultar els resultats als arxius annexos o realitzar noves proves amb altres models o altres paràmetres.

Resultats de les proves

Aquí faré una petita explicació de cada prova i ficaré els resultats més significatius i quines conclusions es poden extreure de cada una.

Tots els eixos estan en unitats internacionals, la posició en metres, la velocitat en m/s i l'acceleració en m/s^2. El eix inferior està en número de mostres.

Proves realitzades

Com ja he explicat, les proves es faran en tandes de 5, i per a cada experiment hi ha diferents tandes canviant certs paràmetres i veient com afecten els canvis als resultats.

Aquí nomes mostraré les proves més interesants o amb millor resultats sinó ocuparia massa pàgines amb gràfiques gairebé iguals. Els resultats poden ser revisats ja que estan tots guardats en arxius de dades del Matlab per a que tothom pugui comprovar.

Correspondència de les variables:

- Gràfic acceleracions:
    - Acel: Acceleració sense filtrar
    - Afilt: Acceleració filtrada
- Gràfic velocitats:
    - Vteorica: velocitat calculada amb el model teòric
    - Vencoder: velocitat calculada amb el model dels encoders
    - Vfilt: velocitat filtrada dels sensors inercials
    - Vfusio: velocitat del mètode de fusió inercials+encoder
- Gràfic posició:
    - Pfilt: posició filtrada dels sensors inercials

- Pteorica: posició calculada amb el model teòric
- Pencoder: posició calculada amb el model dels encoders
- Pfusio: posició del mètode de fusió inercials+encoder

- Grafic pla:
  - Pfilt: representació del moviment en el pla dels sensors inercials
  - Pteorica: representació del moviment en el pla del model teòric
  - Pencoder: representació del moviment en el pla del model dels encoders
  - Pfusio: representació del moviemnt en el pla del mètode de fusió inercials+encoder

Experiments:

Paràmetres:

Qbias:1

Rbias:0.001

Coef:0.45

Qfilt:0.001

Rfilt:0.1

Qfusio:0.001

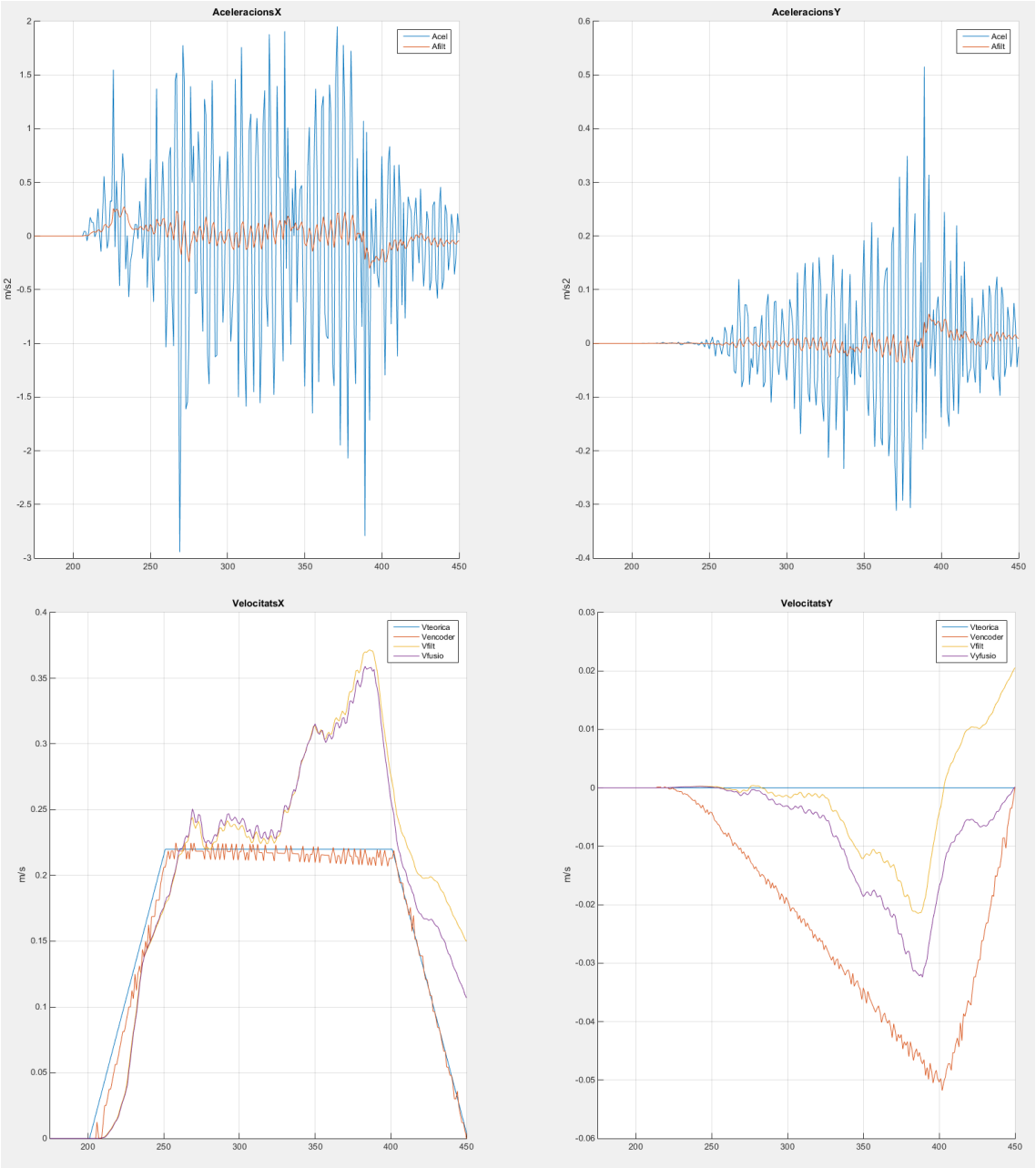
Rfusio:125

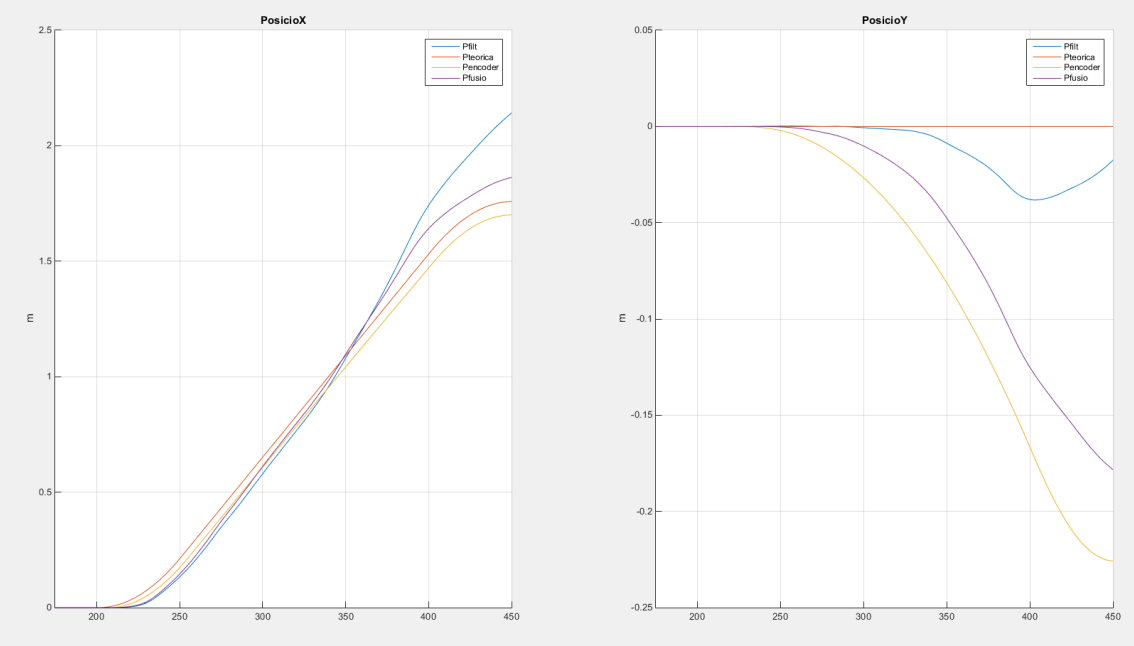
- Rampa doble

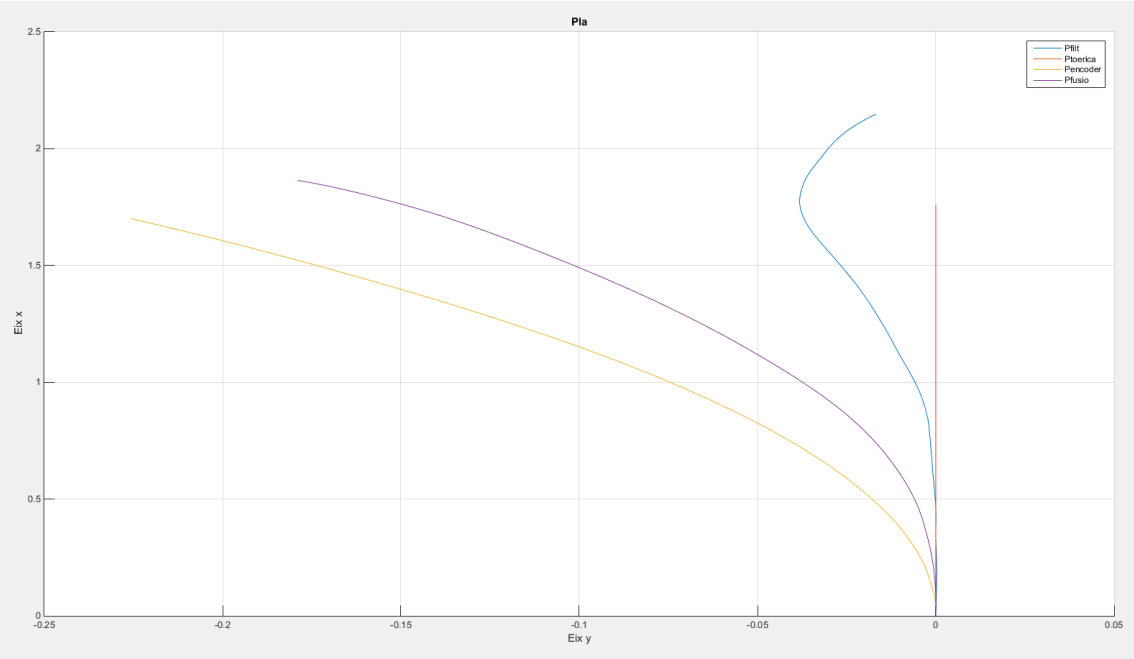
Aquest experiment consisteix en accelerar el robot de manera constant fins a una velocitat definida, mantenir-la durant un cert temps i després desaccelerar el robot de la mateixa manera amb desacceleració constant. El mòbil tarda 2 segons en accelerar i desaccelerar i està 6 segons en velocitat constant.

Tanda d'experiments: 1 Ref. Carpeta: Doble rampa final(2015_8_14)

Núm. d'experiment: 1 Ref: processat2015_8_14;19,17,16


AceleracionsX
Acel
Afilt
m/s2
AceleracionsY
Acel
Afilt
m/s2
VelocitatsX
Vteorica
Vencoder
Vfilt
Vfusio
m/s
VelocitatsY
Vteorica
Vencoder
Vfilt
Vyfusio
m/s





Conclusions de la tanda:

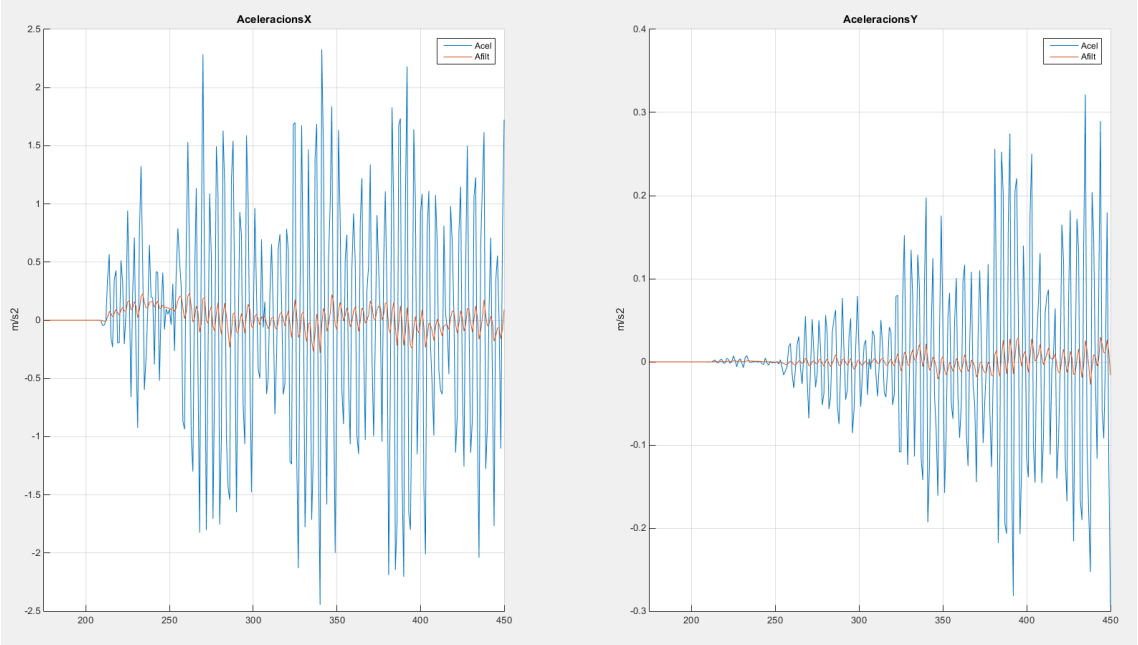
Podem comprovar la aleatorietat del experiments, tot i utilitzar els mateixos paràmetres podem veure com en alguns casos la velocitat no torna a 0 com hauria de ser, veiem com el model del encoder segueix gairebé perfectament el moviment en aquest cas, i el acceleròmetre es veu afectat per la mínima deformitat del terra, podem observar com el moviment no es recte realment, ja que es possible que una roda hagi derrapat una mica o s'hagi trobat amb una inclinació que ha variat el moviment.

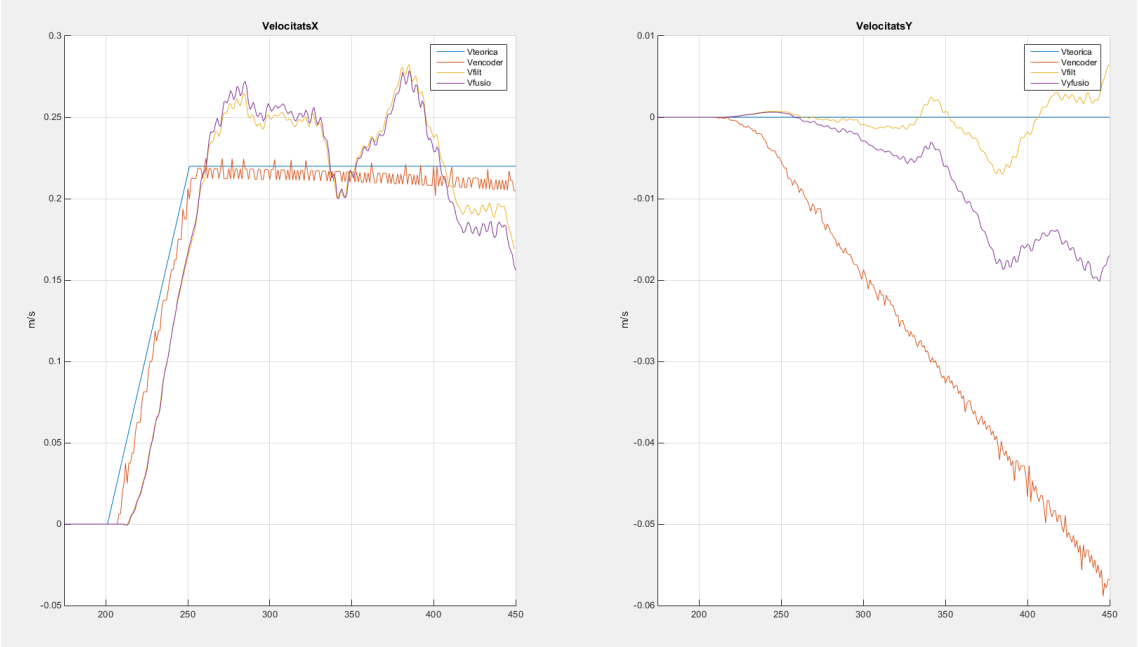
- Rampa simple

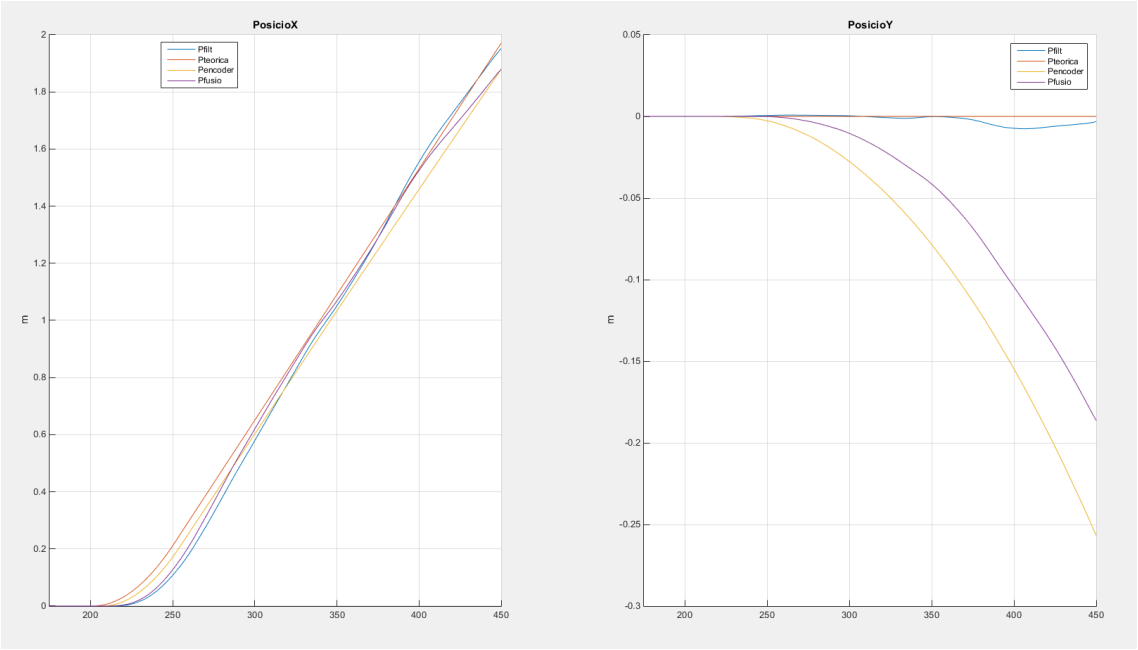
Aquest, a diferència de l'anterior, es manté el robot més temps en moviment fins al final de l'experiment. No hi ha desacceleració final.

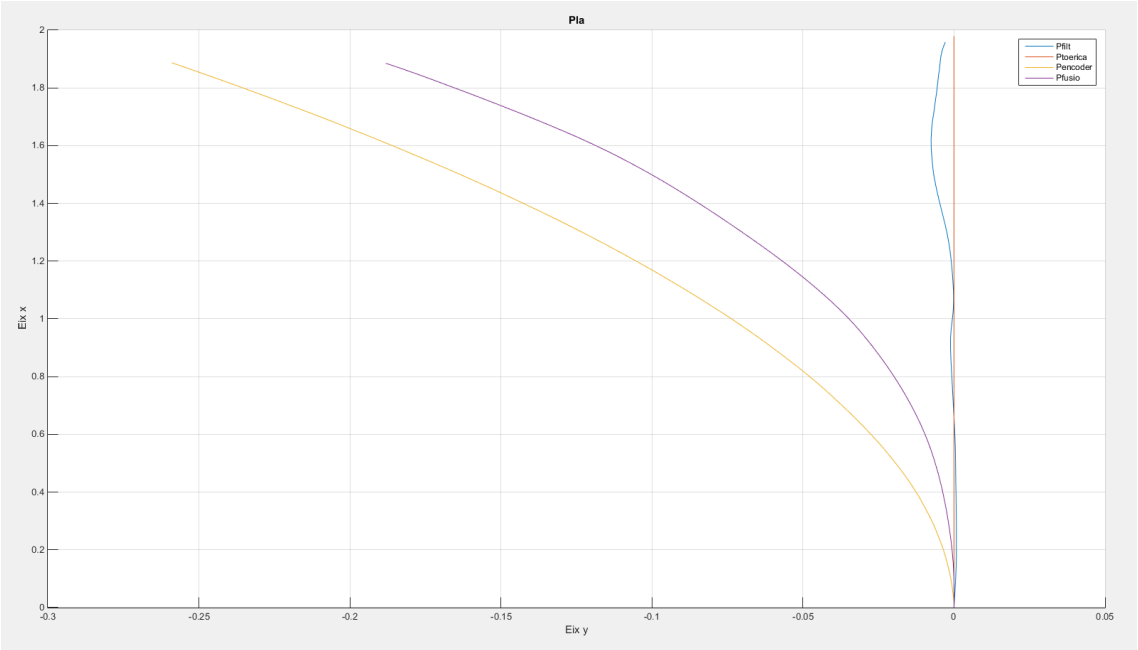
Tanda d'experiments: 2 Ref. Carpeta: Rampa Simple final(2015_8_14)

Núm. d'experiment: 4 Ref: processat2015_8_14;19,57,52









Conclusions de la tanda:

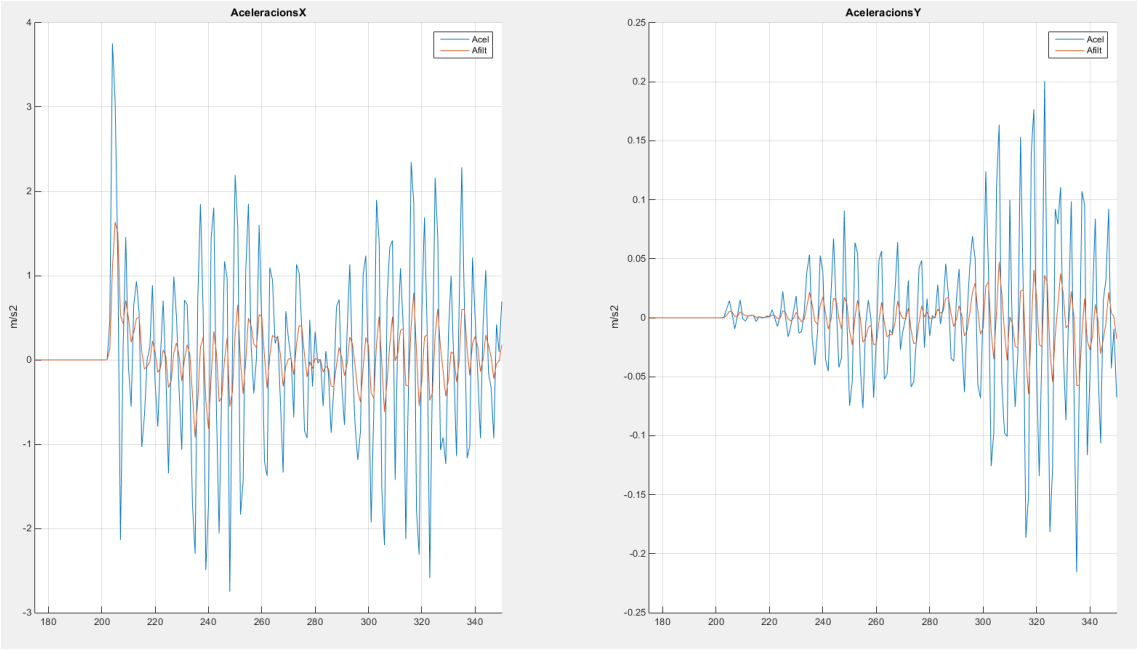
Podem observar com la velocitat en aquest cas es manté estable, i segueix la trajectòria.

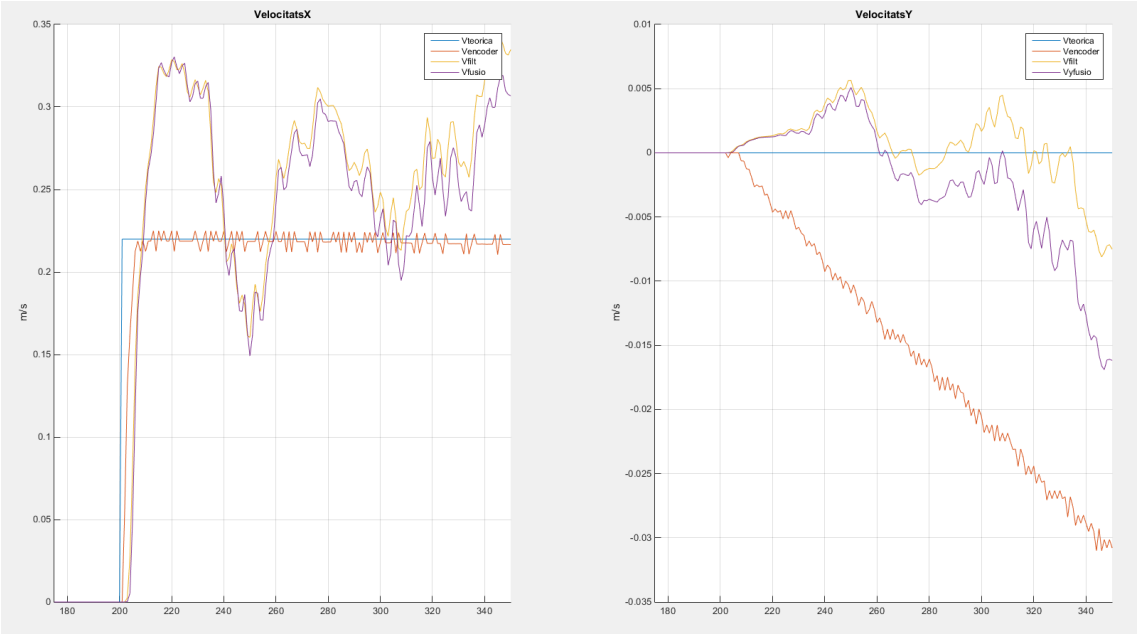
- Esglaó

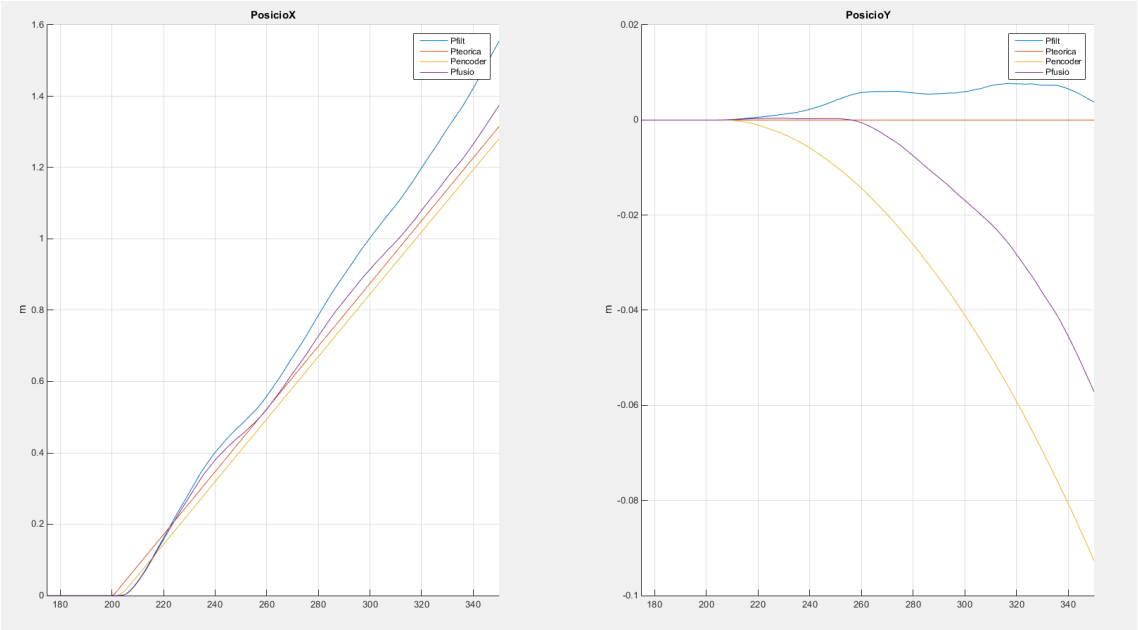
En aquest la acceleració es instantània i el robot passa de velocitat 0 a la desitjada, en l'experiment podrem comprovar l'efecte del lliscament de les rodes a causa de moviments bruscos.

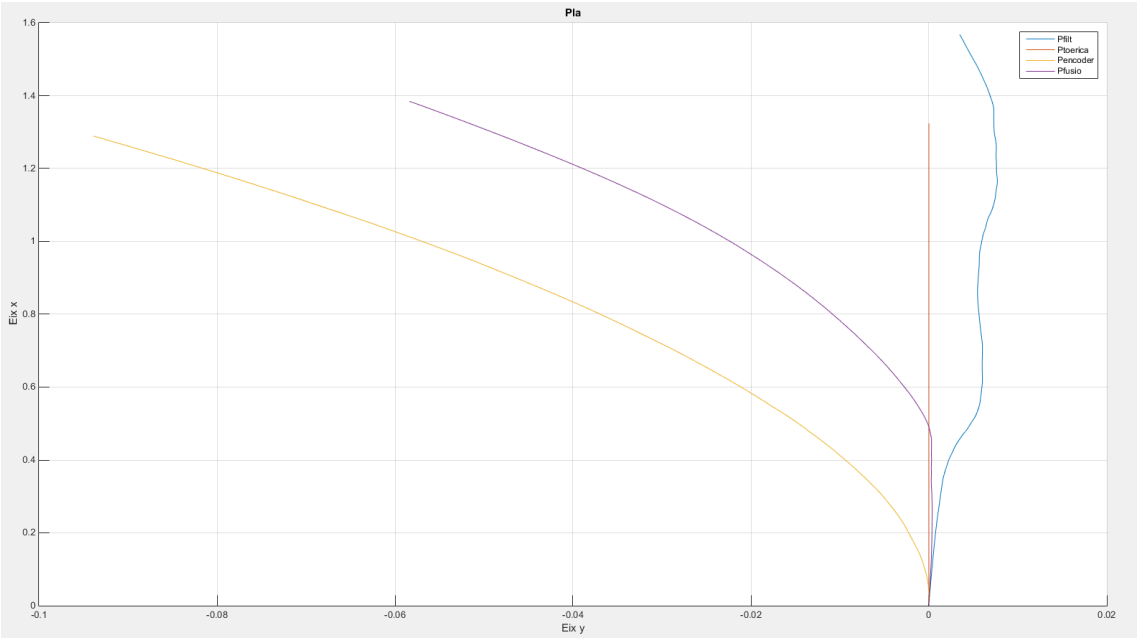
Tanda d'experiments: 3 Ref. Carpeta: Esglao final(2015_8_18)

Núm. d'experiment: 4 Ref: processat2015_8_18;15,30,11


AceleracionsX
Acel
Afilt
m/s2
AceleracionsY
Acel
Afilt
m/s2


VelocitatsX
Vteorica
Vencoder
Vfilt
Vfusio
m/s
VelocitatsY
Vteorica
Vencoder
Vfilt
Vyfusio
m/s





Conclusions de la tanda:

En aquest podem apreciar molts més errors, al tenir una acceleració molt brusca al principi provoca errors en les mesures. Durant aquesta brusca acceleració les rodes derrapen i es perd tracció.

- Pertorbació de trajectòries

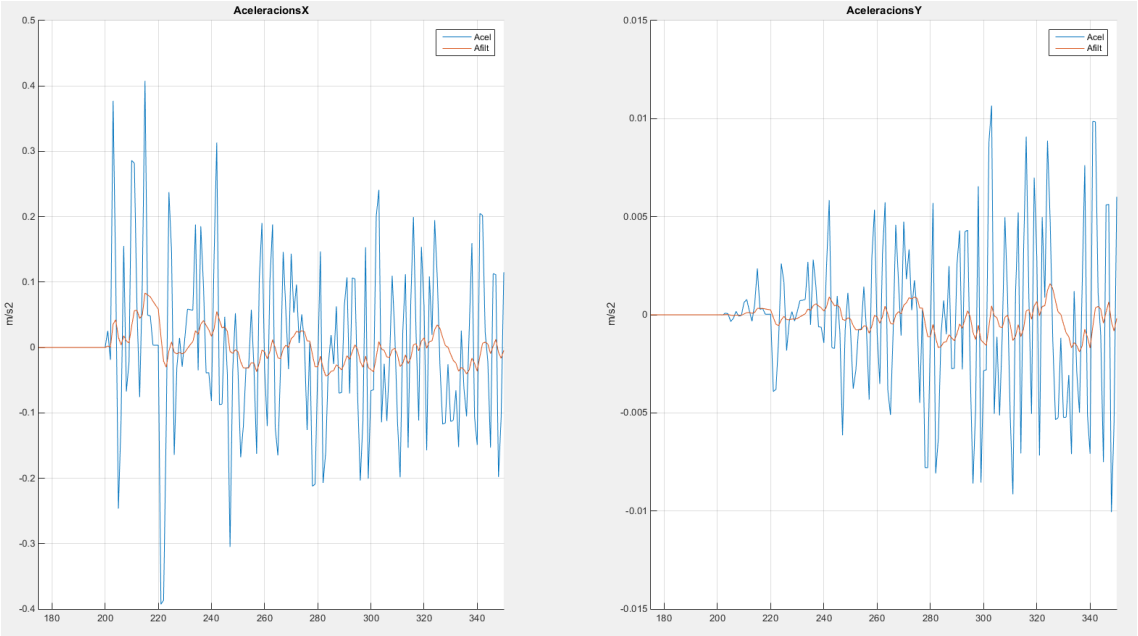
Aquests són un grup d'experiments on provocaré pertorbacions, com evitant el moviment del robot o aplicant una variació externa, i veure com el model es comporta a aquestes pertorbacions.

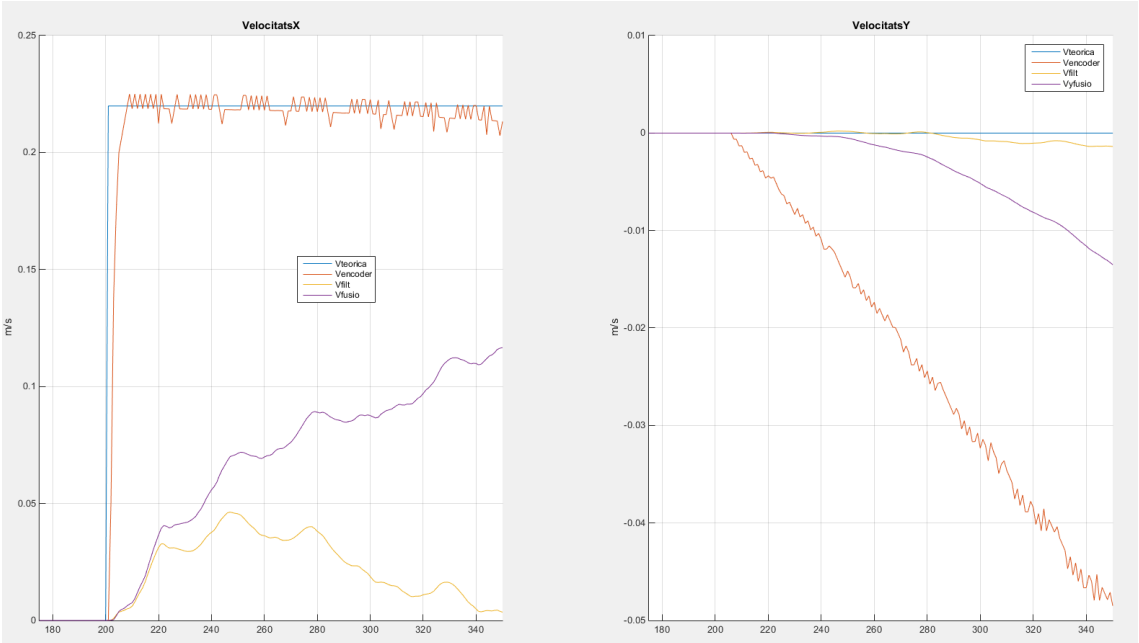
- Rodes no toquen a terra

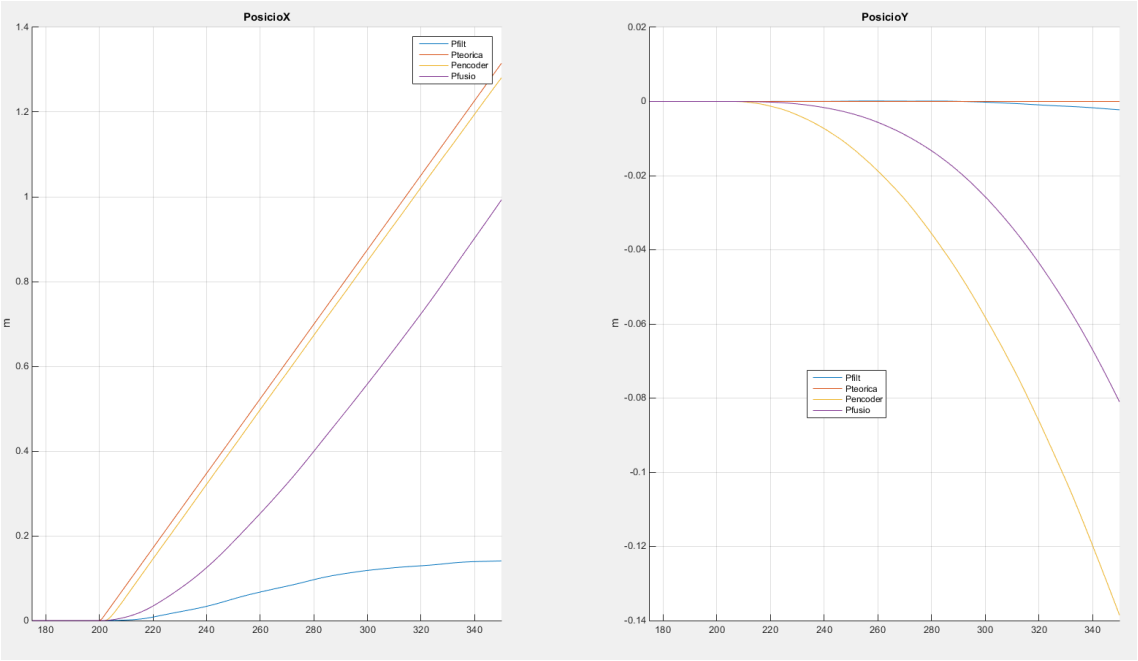
Es col·loca el robot elevat sobre algun objecte i es fa moure les rodes del robot com si s'estigués movent, els encoders de les rodes seguiran el moviment però els sensors inercials no detectaran cap moviment apart de les vibracions.

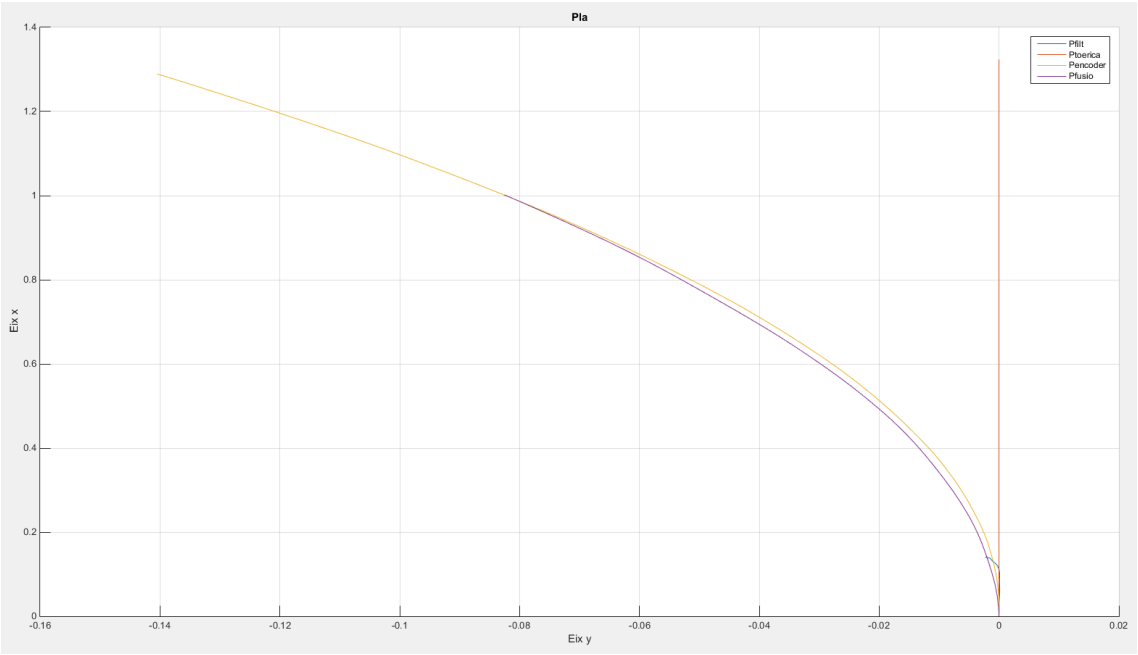
Tanda d'experiments:4 Ref. Carpeta: Floorless final(2015_8_18)

Núm. d'experiment: 5 Ref: processat2015_8_18;15,47,26









Conclusions de la tanda:

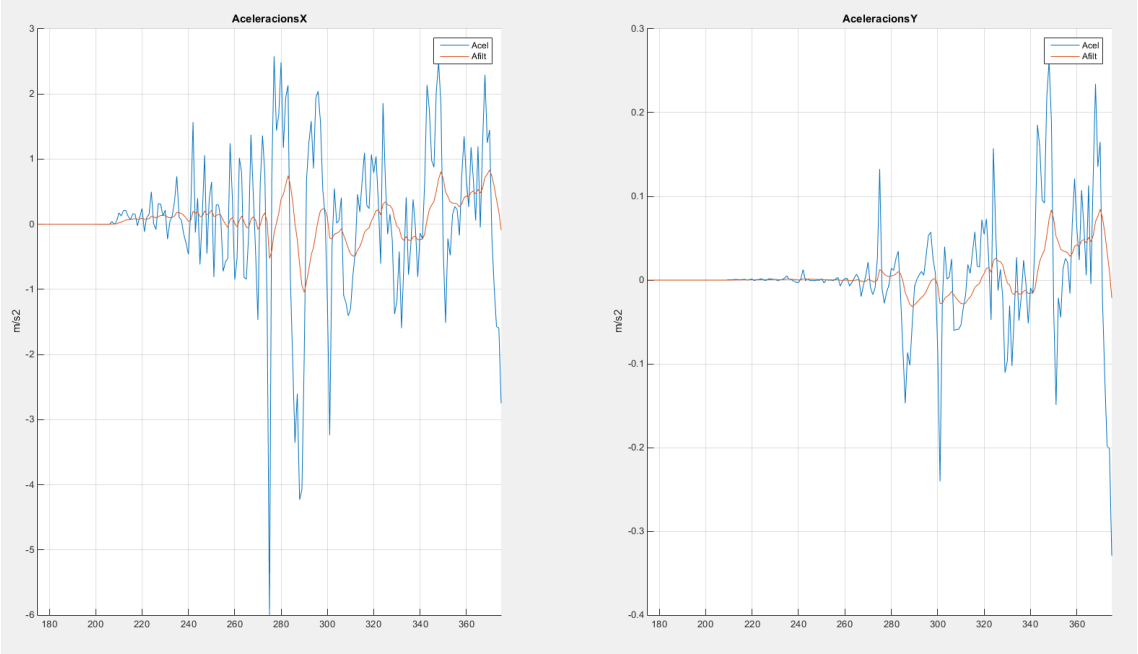
Tot i que aquí els sensors inercials funcionen de manera correcta, el model del encoder és totalment erroni, ja que com les rodes no toquen al terra no provoquen cap moviment al mòbil, i això és un problema molt gran que s'ha de detectar.

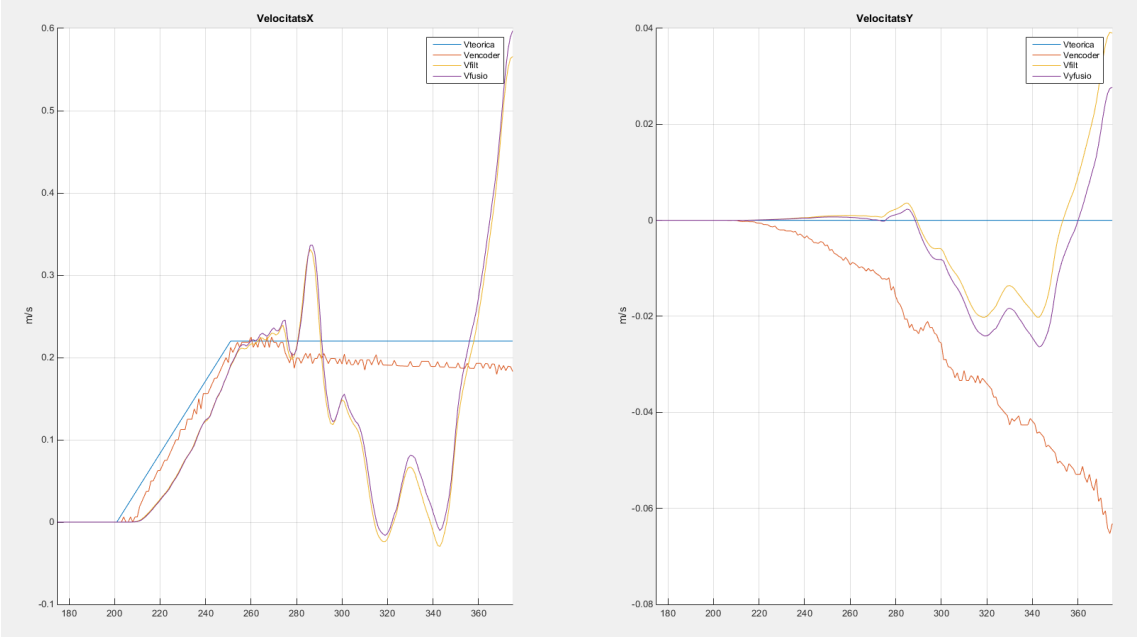
- Robot xoca contra la paret

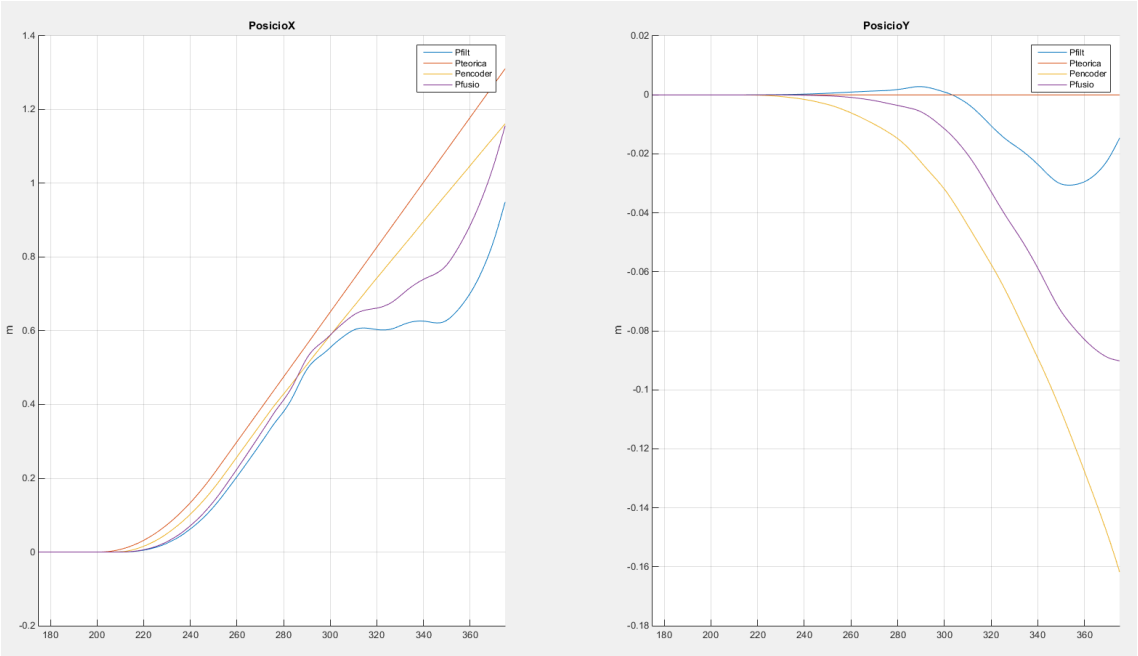
  Aquí el que farem és fer que el robot xoqui amb una paret de manera intencionada, com a l'anterior experiment els encoders seguiran girant però els sensors inercials detectaran el xoc.

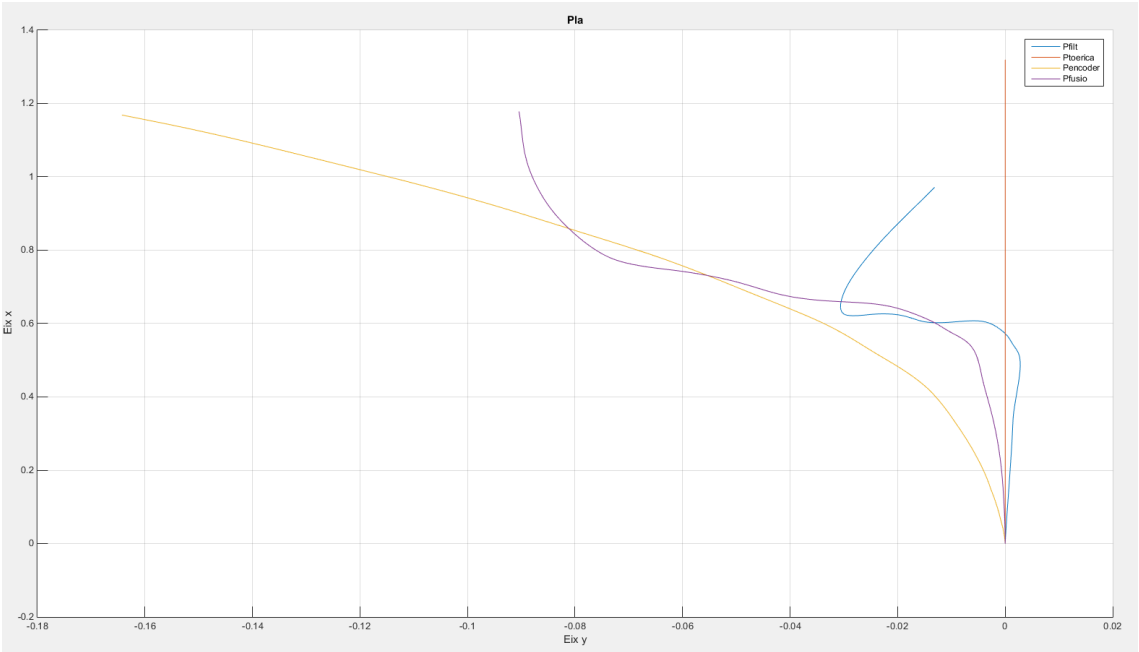
Tanda d'experiments: 5 Ref. Carpeta: Paret final(2015_08_19)

Núm. d'experiment: 2 Ref: processat2015_8_19;19,26,42









Conclusions de la tanda:

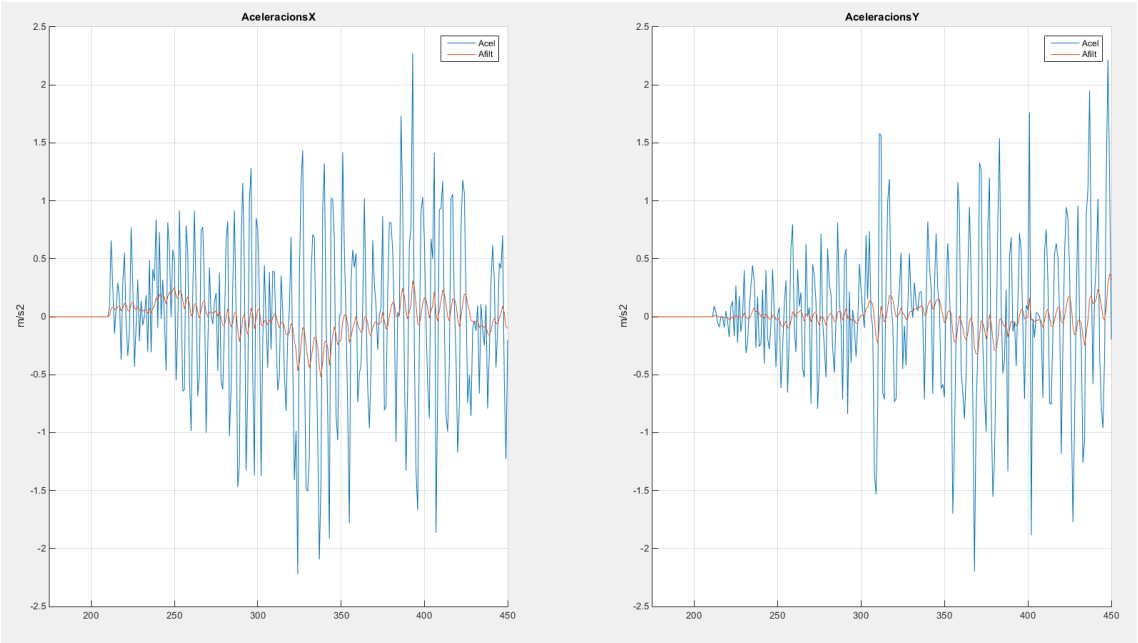
Com podem observar en aquest experiment, la resposta dels sensors inercials és correcta fins a un cert punt, aquest és un experiment complicat ja que el xoc contra la paret tot i ser a baixa velocitat, provoca moltes acceleracions ens diferents direccions.

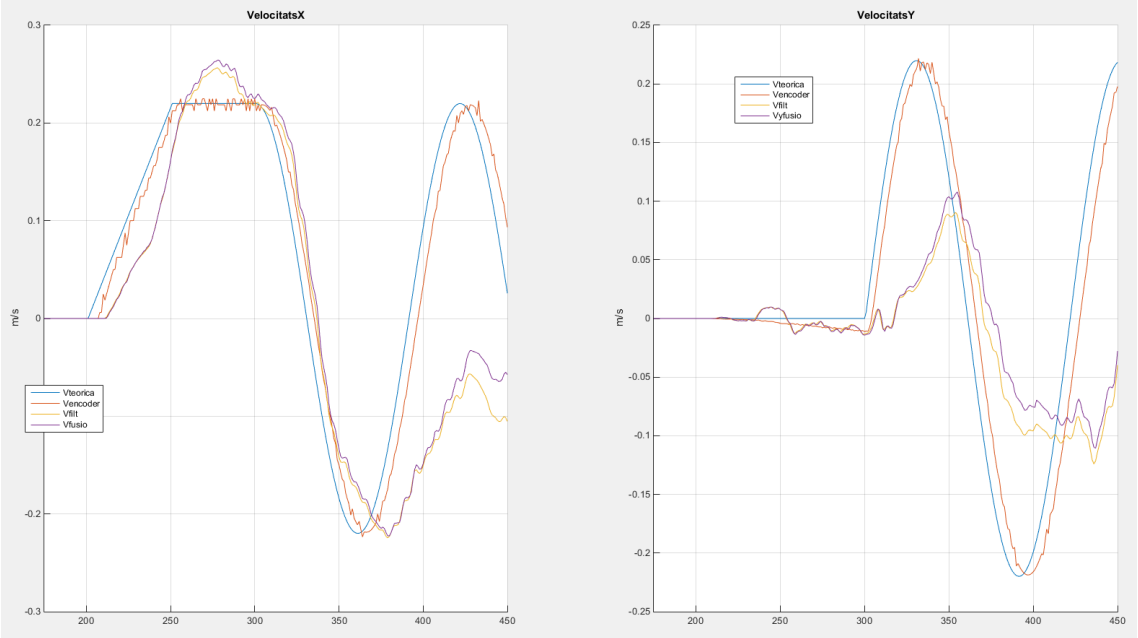
- Prova final (Trajectòries amb corbes)

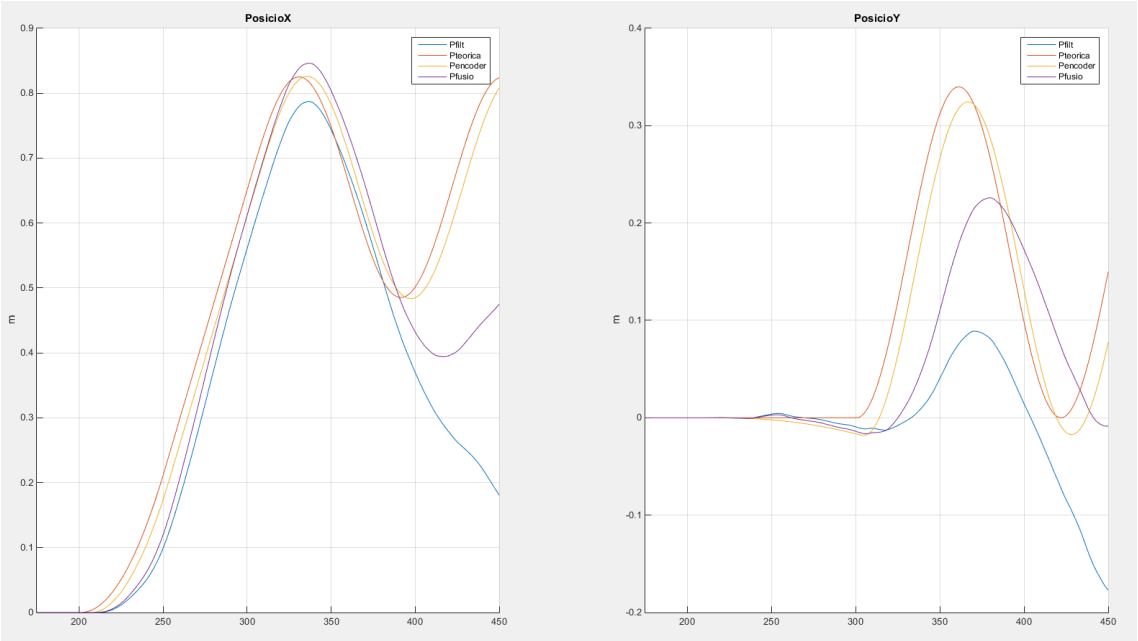
I finalment farem una trajectòria amb una corba; per acabar de comprovar si després de totes les proves i ajustar els paràmetres segons totes les dades recollides anteriorment, obtenim una conclusió sobre quins és/són els mètodes més fiables. Exactament farem desplaçar el mòbil en línia recta mentre accelera i després fer-li realitzar un cercle complet.

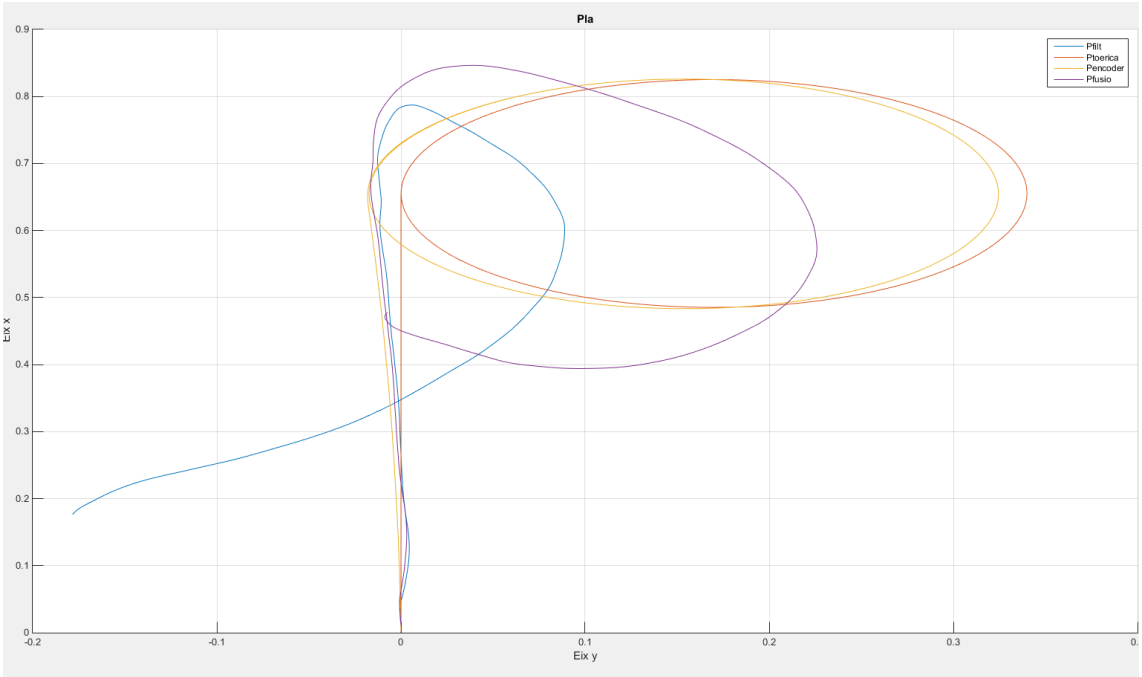
Tanda d’experiments: 2 Ref. Carpeta:

Núm. d’experiment: 2









Conclusions de la tanda:

En aquest experiment podem observar com amb els sensors inercials no podem obtenir una trajectòria fiable, tot i els mètodes aplicats no s'han obtingut uns resultats satisfactoris.

12.Conclusió

Durant la realització d'aquest treball he hagut de buscar molt a fons en un tema molt complicat, amb moltes variables externes, molts errors i interferències. He aconseguit trobar mètodes que poden resoldre aquests problemes, però com en tenen d'altres la millor solució és fusionar aquests mètodes i fer un combinat, on durant certs moments se li dóna més importància a un que a l'altre.

Tot i que els resultats dels meus experiments no han estat completament satisfactoris, crec que amb el material disposat ja són els suficientment correctes. Amb un material més precís i estable, podríem aconseguir resultats el suficientment fiables.

Els objectius inicials no han estat tots complerts, però amb el temps i la dedicació completa es podrien realitzar tots, i fins i tot expandir-los i implementar el mètode compacte per a altres projectes més complicats.

El seguiment de posició de mòbils continua sent un camp molt complicat, ja que com hem vist cada mètode proposat té molts contres que s'han d'estudiar abans d'aplicar-los.

Al ser un procés molt complex requereix molta capacitat de processar, això provoca que li resti capacitat al robot per a realitzar altres tasques, i s'hagin de buscar altres mètodes per a realitzar els càlculs. D'una manera semblant al que he realitzat jo, que he hagut de separar l'adquisició de dades i el processament al meu ordinador. S'hauria d'idear un sistema de comunicació perquè se li pogués enviar ordres al mateix moment que es processen les dades en temps real.

13.Bibliografia

- Fernando Reyes Cortés. *Robótica Control de robots manipuladores.* Marcombo
- Vedran Kordic. *Kalman filter.* Intech
- Roland Siegwart,Illah Reza Nourbakhsh. *Introduction to Autonomous Mobile Robots*
- R. Siegwart, M. Mason. 7630 – Autonomous Robotics Mobile Robot Kinematics.

  Adreça electrònica:

  http://dream.georgiatech-metz.fr/sites/default/files/3%20-%20Mobile%20Robot%20Kinematics.pdf
- Informació sobre els sensors: https://www.hitechnic.com/
- Informació sobre els motors: http://www.philohome.com/motors/motorcomp.htm

ANNEX

Codi pel guardat dels resultats de les proves, crea carpetes nombrant-les segons el dia, hora i segons, per facilitar la organització. A més a més, cada experiment de dades crea una carpeta on es guardaran totes les execucions amb les variacions que es facin als paràmetres:

```
clear Vfinal Tacel Tconst Inici Pendent var1;
c = clock;
dir = 'C:\Users\marc\Desktop\Projecte lego';
foldername=[num2str(c(1)),'_',num2str(c(2)),'_',num2str(c(3))];
if(exist('a'))
else
filename=['valors',num2str(c(1)),'_',num2str(c(2)),'_',num2str(c(3)),';',
num2str(c(4)),',',num2str(c(5)),',',num2str(round(c(6)))];
direct=[dir,'\',foldername,'\',filename];
mkdir(foldername)
save(direct,'-regexp','^(?!(c|dir|direct|filename|foldername).*$).')
hora = c;
end
Variablesunio
sim('C:\Users\marc\Desktop\Projecte lego\15,05,08\uniofinal.slx')
disp('Simulat')
foldername=[foldername,'\','Processat(',num2str(hora(4)),',',num2str(hora
(5)),',',num2str(round(hora(6))),')'];
if(exist('a'))
else
mkdir(foldername)
end
filename=['processat',num2str(c(1)),'_',num2str(c(2)),'_',num2str(c(3)),'
;',num2str(c(4)),',',num2str(c(5)),',',num2str(round(c(6)))];
direct=[dir,'\',foldername,'\',filename];
save(direct,'-regexp','^(?!(c|dir|direct|filename|foldername).*$).')
disp('Guardat')
a = 0;
Grafiques
```

Arxiu ".m", on es guarden els paràmetres del model i permet el ràpid canvi d'ells i automàticament a la següent execució s'actualitzaran.

```
k=9.81/200;
k_motor=0.004398;
Radi= 0.18/(2*pi);
Dist=0.17;
coef = 0.3;
XBias = -Ax.signals.values(1);
YBias = -Ay.signals.values(1);
GyroBias = -Gyro.signals.values(1);

Abias=[1 0 0 0;0 1 0 0;ts 0 1 0;0 ts 0 1];
Cbias=[1 0 coef 0;0 1 0 coef];


Qbias=Abias*(Abias.');
Rbias=0.001*Cbias*(Cbias.');
```

```
Afiltre=[1 0 ts 0 (ts.^2)/2 0;0 1 0 ts 0 (ts.^2)/2;0 0 1 0 ts 0;0 0 0 1 0
ts;0 0 0 0 1 0;0 0 0 0 0 1];
Cfiltre=eye(6);


Qfiltre=0.005*Afiltre*(Afiltre.');
Rfiltre=0.1*eye(6);


Afusio=[1 0 ts 0 0 0;0 1 0 ts 0 0;0 0 1 0 0 0;0 0 0 1 0 0;0 0 0 0 0 0;0 0
0 0 0 0];
Bfusio=[(ts.^2)/2 0;0 (ts.^2)/2;ts 0;0 ts;1 0;0 1];
Cfusio=[1 0 0 0 0 0;0 1 0 0 0 0;0 0 1 0 0 0;0 0 0 1 0 0];


Qfusio=0.001*Afusio*(Afusio.');
Rfusio=125*Cfusio*(Cfusio.');
```