Universitat
Autònoma
de Barcelona

Escola d'Enginyeria

ESTUDI: MICROCONTROLADOR LPC2119 – ARM

CAS PRÀCTIC: ANÀLISIS, DISSENY I CONSTRUCCIÓ D'UN ROBOT AMB PÈNDUL INVERTIT

Memòria del Projecte Fi de Carrera
d'Enginyeria en Informàtica
realitzat per
Josep Maria Sanz Xicola
i dirigit per
Joan Oliver Malagelada
Bellaterra, 16 de Setembre de 2009

Universitat Autònoma de Barcelona

Escola Tècnica Superior d'Enginyeria

El sotasignat, ...

Professor/a de l'Escola Tècnica Superior d'Enginyeria de la UAB,

CERTIFICA:

Que el treball a què correspon aquesta memòria ha estat realitzat sota la seva direcció per en

I per tal que consti firma la present.

Signat: .....

Bellaterra, ...de.....de 200.....

Agraïments

Voldria agrair el present projecte a,
el meu director de projecte per la seva atenció i predisposició.
Als meus pares per l'ajuda inestimable,
a l'atenció de les meves germanes per el seu parer,
i finalment, a tot el suport rebut pels qui m'envolten.

Índex

1. Introducció

El present projecte pretén ser una anàlisis de les prestacions i possibilitats del microcontrolador LPC2119 amb l'objectiu d'aprofundir en aspectes com ara la seva arquitectura i els seu corresponent mode de funcionament. Com a resultat de tals coneixement, i amb la finalitat d'integrar el dispositiu sobre un sistema amb requeriments de temps real, ha esdevingut la materialització d'un robot que incorpora un pèndul invertit d'un sol eix.

La implementació del pèndul invertit busca explotar les possibilitats que ens brinda el micrcontrolador en el camp dels sistemes robòtics dins un context on la restricció temporal és cap dalt. Amb la categoria de cas pràctic, la pretensió del pèndul invertit és mantenir en tot moment la seva estabilitat vertical, rectificant amb un moviment del robot, si és degut, el procés de davallada del pèndul, contrarestant la força de la caiguda per restablir la verticalitat.

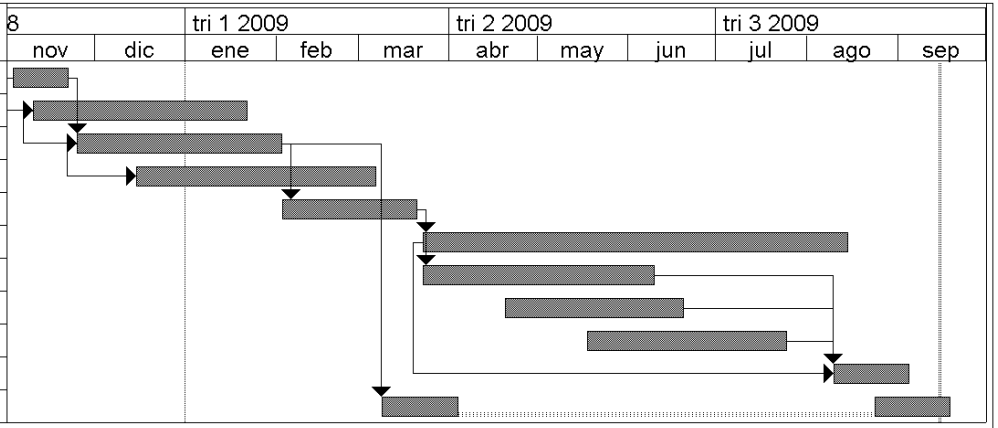
Durant la realització del pèndul, fruit de les proves i l'experimentació, ha esdevingut el desenvolupament de tota una llibreria de mètodes i funcions software que configuren les diferents parts del microcontrolador. Aquesta feina, més enllà de la utilitat en aquest projecte, pot tenir un abast més ampli, com ara una projecció cap a la docència o la reutilització en altres projectes.

Per tant, la present memòria es desglossa en dues parts diferents: una primera part teòrica, breu introducció on s'expliquen els conceptes més generals de l'arquitectura i propietats més rellevants de la plataforma d'estudi; i una segona part on es descriu abastament el pèndul invertit: els factors que s'han considerats, la metodologia seguida, experiments, resultats i viabilitat...

Finalment, la memòria conclou exposant les resolucions i experiències obtingudes durant tot aquest període: des de l'assignació del projecte fins la redacció d'aquesta.

2. Planificació i costos

Id	Nombre de tarea	Duración	Comienzo
1	Busqueda de documentació	15 días	lun 03/11/08
2	Estudi de l'arquitectura ARM	53 días	lun 10/11/08
3	Estudi del MCU: LPC2119	50 días	mar 25/11/08
4	Proves simples	60 días	lun 15/12/08
5	Dissenys avançats	34 días	mar 03/02/09
6	Cas pràctic: Pèndul invertit (estudi)	105 días	lun 23/03/09
7	Proves amb l'acceleròmetre	57 días	lun 23/03/09
8	Proves amb els servomotors	45 días	lun 20/04/09
9	Proves amb el PID	50 días	lun 18/05/09
10	Proves amb tot el sistema	20 días	lun 10/08/09
11	Redacció memòria final	40 días	lun 09/03/09



Proyecto: PFC

Tarea

División

Figura 2-1: Planificació del projecte

Element	Quantitat	Preu Unitat	Preu
Boe Bot Chassis (Parallax)	1 u	16.10 €	16.10 €
SumoBot Wheel & Tire (Parallax)	2 u	2.70 €	5.40 €
DC Rotation Servo (Parallax)	2 u	8.9 €	17.8 €
AA Battery Holder (Parallax)	1 u	4.76 €	4.76 €
Battery (Varta)	1 u	3.10 €	3.10 €
Accel. LIS302 DL	1 u	13.67 €	13.67 €
ET-ARM Stam Module (LPC2119)	1 u	19.15 €	19.15 €
ET-ARM Stamp Board	1 u	17.06	17.06 €
Altres	-	100 €	100 €
		Total:	**197.04 €**

Taula 2-1: Taula de costos per la fabricació del robot

3. El microcontrolador LPC2119

LPC2119 és un microcontrolador de la casa NXP Semiconductors, fundada per Philips. Com molts altres microcontroladors, posseeix una memòria, un processador i entorn aquest, es revesteix d'un conjunt de perifèrics que es comuniquen amb el processador a través d'un bus. Tot plegat s'integra dins un mateix encapsulat.

Aquesta vindria a ser una breu definició de microcontrolador però, en quins punts es diferenciaria LPC2119 respecte la resta de microcontroladors? De ben segur, l'atractiu més interessant del dispositiu integrat és, més enllà de les peculiaritats dels seus perifèrics, el processador ARM que incorpora.

Per tant, sembla obligatori haver de fer una revisió del que és i en què es basa l'arquitectura ARM així com del processador que incorpora l'LPC2119.

Finalment, més enllà del processador, explicarem el sistema d'interrupcions, memòria i enumerarem els diferents perifèrics.

3.1. ARM

ARM Ltd. és una companyia anglesa fundada a principis dels 90 com a fruit de l'associació empresarial entre Acorm i Appel. Si bé, l'arquitectura ARM va ser creada amb anterioritat per Acorn, a partir de llavors l'empresa responsable del manteniment i evolució de l'arquitectura va passar a ser ARM Ltd. Així doncs, vora els anys noranta, ARM Ltd. neix amb una arquitectura ja creada i amb unes directrius fixades: **simplicitat en el disseny i baix consum**. Aquests aspectes van xocar amb el corrent de disseny antagònic dels processadors CISC, creixement i consum desmesurat. No és estrany doncs que ARM Ltd., amb els seu compromís de rendiment i consum, es fes ràpidament amb el incipient mercat de dispositius embedded. Actualment, l'hegemonia dels processadors ARM en aquest sector

encara perdura, sent exemple el processador ARM7TDMI amb milions de llicències venudes en el camp de la telefonia mòbil.

El secret de l'èxit rau en l'arquitectura. Sophie Wilson i Steve Furber, enginyers d'Acorm, van ser els encarregats de desenvolupar una arquitectura conforme els requeriments insígnia.

Pocs anys abans, David A. Patterson establia les bases del model d'arquitectura RISC. Les característiques proposades suposaven un nou enfocament alhora de dissenyar processadors: conjunt reduït d'instruccions amb instruccions d'amplada fixe, ample banc de registres, arquitectura load-store i execució d'una instrucció per cicle. Patterson i Ditzel van argüir, no erradament, que l'arquitectura RISC oferia els següents avantatges [Fubert00]:

- Una àrea d'integració més petita
- Un temps de disseny del processador més curt
- Un rendiment més elevat com a resultes de tenir un processador simple (freqüències de treball més elevades)

L'arribada de tot aquest nou panorama en el camp de la computació va ser font d'inspiració inestimable per la gènesis de la nova arquitectura ARM, adoptant molts dels preceptes RISC. La dada històrica és produeix quan el 1986 es comercialitza el primer model ARM, el processador ARM2 (l'ARM1 va restar com a prototip), sent el primer processador comercial RISC.

No obstant això, l'arquitectura ARM no va seguir a la literalitat l'especificació RISC. Tot i estar abastament inspirada, hi van haver punts divergents com ara: absència de window register, no implementació de *delayed branches* i la no garantia d'una instrucció per cicle; així com també la inclusió d'instruccions de múlti-transferència no contemplades en l'arquitectura RISC original.

El pas del temps ha demostrat l'encert de l'arquitectura ARM, el qual proporciona tots els beneficis de l'enfocament RISC però al mateix temps, n'evita alguns petits aspectes que si cap, la fan encara més idònia per el desenvolupament de sistemes embedded.

A més a més, no només la simplicitat del disseny de l'arquitectura va comportar un èxit inicial, sinó que gracies aquesta, les successives revisions van conservar part de l'essència de la primera, fet que posa de relleu la robustesa i el caire modular

de l'arquitectura primigènia, en gran mesura, també perquè la funcionalitat extra es va i es continua procurant afegir a partir d'extensions.

Versió	Any	Característiques	Implementacions
v1	1985	26-bit RISC	ARM1
v2	1987	ARM1 + Coprocessador	ARM2, ARM3
v3	1992	32-bit, MMU, 64-bit MAC	ARM6, ARM7
v4	1996	Thumb	ARM7TDMI, ARM8, ARM6TDMI
v5	1999	Extensions DSP, Jazelle	ARM10
v6	2001	SIMD, Thumb-2, Trust-Zone,multiprocessament	ARM11, ARM11 MPCore
v7	-	VFP-3	Cortex

Taula 3-1 Evolució de l'arquitectura ARM [Ryz06]

CPU	#pipeline	Organització en mem.	Freq.	MIPS/MHz
ARM6	1985	Von Neumann	25 MHz	-
ARM7	1987	Von Neumann	66MHz	0.9
ARM8	1992	Von Neumann	72 MHz	1.2
ARM9	1996	Harvard	200 MHz	1.1
ARM10	1999	Harvard	400 MHz	1.25
Strong ARM	2001	Harvard	233 MHz	1.15
v7	-	Von Neumann	550 MHz	1.2

Taula 3-2 Evolució de l'arquitectura ARM [Anon00]

3.1.1. Arquitectura ARM

Enumerem algunes de les característiques de l'arquitectura ARM més importants:

- Arquitectura load-store
- Instruccions de longitud fixe, 32-bits
- Format d'instruccions amb tres camps d'adreça

El fet de ser una arquitectura load-store implica que totes les instruccions, a excepció de les instruccions de transferència a memòria, treballen amb valors dels registres o immediats i el resultat s'emmagatzema sempre en un segon registre. Les instruccions de transferència a memòria són dedicades i exclusives per a tal propòsit; no existeixen instruccions de memòria a memòria com podriem trobar en una arquitectura CISC.

L'arquitectura especifica una amplada de paraula de 32-bits, tant per instruccions com per a dades, i la memòria és vista com un pla de 2^{32} bytes direccionables amb un endianess no especificat (en funció de la versió i el fabricant, podem trobar processador ARM tant en Big-endian com en Little-endian). Per cada accés a memòria es recuperen paraules de quatre bytes —*4-byte boundaries align*—.

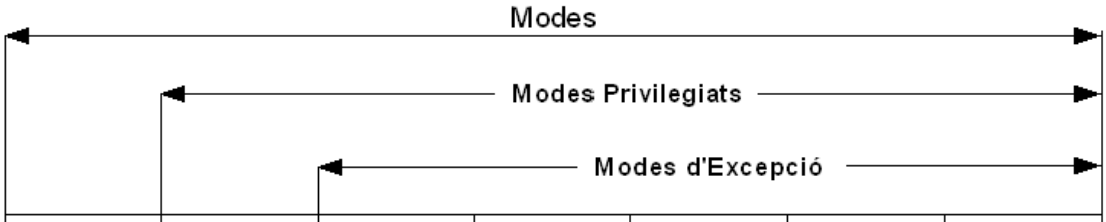
Totes les instruccions conserven una longitud de 32-bits, fet que arrossega dues conseqüències: descodificació simple de les instruccions i conservar un pipeline predictible, fàcil d'explotar. Així mateix, totes les instruccions tenen com a màxim tres camps d'adreces pels operands (direcció de registre o literal): dos operands d'origen i un tercer com a destí.

Des del pipeline de tres etapes, quan es va concebre l'arquitectura ARM, ha transcorregut molt de temps ampliant en gran mesura la família de processadors. No obstant, fins a la quarta revisió de l'arquitectura el pipeline va conservar les tres etapes —*Fetch - Decode - Execute*—, no així per revisions posteriors, podent trobar processadors amb 5 (ARM9TDMI, ARMv4), 6 (ARM1020E, ARMv5), 8 (ARM1136J, ARMv6), 9 (ARM1156T2, ARMv6) i fins a 13 (Cortex-A8, ARMv7) etapes de pipeline [Wiki01]. ¿La tendència dels nous ARM traeixen la simplicitat dels seus orígens en pro d'un cicle-per-instrucció més baix? Sens dubte, un pipeline més llarg representa un processament de l'instrucció més complex (un grau de segmentació major). No obstant, ARM continua sent lider consolidat en sistemes embedded de baix consum i, si bé són processadors més

complexos per un guany de capacitat computacional, tots ells continuen distingint-se per ser encara processadors RISC.

3.1.2. El model de programació

Durant l'execució són visibles setze registres de propòsit general en qualsevol mode de treball, podent ser tots ells operands font com destí d'una operació.



User	System	Supervisor	Abort	Undefined	Interrupt	Fast Interrupt
r0	r0	r0	r0	r0	r0	r0
r1	r1	r1	r1	r1	r1	r1
r2	r2	r2	r2	r3	r3	r3
r4	r4	r4	r4	r4	r4	r4
r5	r5	r5	r5	r5	r5	r5
r6	r6	r6	r6	r6	r6	r6
r7	r7	r7	r7	r7	r7	r7
r8	r8	r8	r8	r8	r8	r8_fiq
r9	r9	r9	r9	r9	r9	r9_fiq
r10	r10	r10	r10	r10	r10	r10_fiq
r11	r11	r11	r11	r11	r11	r11_fiq
r12	r12	r12	r12	r12	r12	r12_fiq
r13	r13	r13_svc (SP)	r13_abt (SP)	r13_und (SP)	r13_irq (SP)	r13_fiq (SP)
r14	r14	r14_svc (LR)	r14_abt (LR)	r14_und (LR)	r14_irq (LR)	r14_fiq (LR)
PC	PC	PC	PC	PC	PC	PC
CPSR	CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
		SPSR_svc	SPSR_abt	SPSR_und	SPSR_irq	SPSR_fiq

Figura 3-1: Conjunt de registres ARM [ARM05]

Prenen rellevant importància els registres `r13`, `r14` i `r15` que, tot i ser registres de propòsit general que contenen dades operables amb instruccions comunes, tenen un tractament especial durant l'execució. Les seves funcions són:

- **`r13`:** StackPointer (SP)
- **`r14`:** Link Register (LR, direcció de retorn)
- **`r15`:** Program Counter (PC)

A tots els efectes, encara que tenen una funció específica, són vistos com a registres normals. Per tant, no és aconsellable operar directament sobre aquest registres i en tal cas, s'ha de ser molt curós a l'hora de modificar-los.

Existeixen únicament dos registre de propòsit específic. Els dos estan relacionats amb el comportament del processador. Un és la paraula d'estat (CPSR, Current Program State Register) i l'altre és el registre de paraula d'estat guardada (SPSR, Saved Program State Register).

CPSR reflexa l'estat actual del processador:

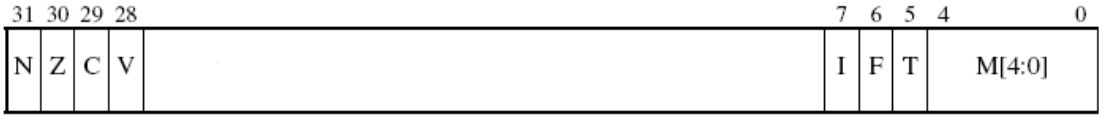


Figura 3-2: Paraula d'estat, CPSR [ARM05]

Els camps són els següents:

- **N, Z, C, V:** Flags condicionals. S'alteren en funció del resultat d'una operació de processament.
- **I,F:** Flags d'activació/desactivació d'interrupcions. Interrupt i Fast interrupt.
- **T:** Bit d'execució Thumb o ARM.
- **Mode:** Indiquen en quin mode ens trobem.

SPSR és propi dels modes privilegiats (a excepció del mode supervisor) els quals s'entren a través d'una excepció, havent de guardar en SPSR el valor de CPSR del mode anterior per recuper-lo un cop finalitzada l'excepció.

Sent registres especials, aliens al banc de registres, és necessari utilitzar instruccions especials per a manipular els valors. Bàsicament hi ha dues instruccions:

- Recuperar el valor d'un dels registre, `MRS`
- Emmagatzemar un valor a un dels dos registres, `MSR`.

3.1.3. Els modes d'execució

L'arquitectura ARM defineix una sèrie de modes d'execució que determinen uns privilegis. Aquests privilegis són fonamentalment dos: accés a l'espai adreçable de la memòria (existència de zones de memòria amb mapeig de perifèrics, registres del sistema...) i execució a instruccions restringides (`MRS` i `MSR`).

Existeixen set modes d'execució:

- **User mode:** És el mode per defecte, on majoritàriament s'executarà tot el software. És l'únic mode que té restringits els privilegis.
- **System mode:** Mode per el SO. Dins d'aquest mode el SO gestiona i manté les diferents tasques.
- **Software Interrupt mode** (`svc_mode`)**:** Similar a l'anterior però aquest s'accedeix a partir d'una excepció software (trap). Facilita la implementació d'un mecanisme a crides a sistema (syscalls).
- **Interrupt mode** (`int_mode`)**:** Mode que s'accedeix a partir d'una excepció generada per una petició d'interrupció.
- **Fast Interrupt mode** (`fint_mode`)**:** Igual que l'anterior però l'excepció és causada per una petició d'interrupció del tipus Fast Interrupt. Les Fast Interrupt són interrupcions amb més prioritat que les interrupcions i amb temps de resposta més ràpid.
- **Undefined Instruction mode:** L'accés es produeix al produir-se una excepció d'una instrucció impossible de descodificar per el processador ni per els coprocessadors que pugui integrar. La causada pot ser per una instrucció que encara no assignada a l'ARM ISE o bé, per inicià una emulació software de la instrucció.
- **Abort mode:** Mode per tractaments de memory faults. L'excepció pot ser font d'un prefetch abort (fallida a recuperació d'una instrucció en memòria) o d'un data fault (fallida en la transferència d'una dada cap o des de a memòria).

És important torna a fer un esment al banc de registres un cop havent llistat els diferents modes. Com sabem, el model de programació es basarà entorn als setze registres que formen el banc més els dos registres especials CPSR i SPSR. Ara bé, això no significa que el nombre total de registres siguin aquests divuit.

Cada mode veu setze registres de propòsit general que conformen el banc de registres més CPSR. No obstant això, existeixen registres exclusius per a cada mode que sol·lapen les posicions (veure **Figura 3-1**).

Els modes d'excepció —tots els modes excepte user i system mode—, just per la seva condició d'eventualitat, tenen cadascun un registre SPSR propi per emmagatzemar la paraula d'estat del mode que reemplace. A més, tots els modes d'excepció posseeixen un SP (r13) i LR (r14) propis, sent inaccessibles des d'altres modes i viceversa.

Excepcionalment, fint_mode posseeix, a banda de SP i LR, cinc registres propis en el banc de registres que van de r8 fins r12.

Finalment, user mode i system mode tenen tots els registres en comú, diferenciant-se únicament l'un de l'altre només perquè el segon té privilegis d'execució i d'accés complet al mapa de memòria.

Així doncs, per exemple: el registre r0 serà comú a tots els modes, mentre que r8 també ho serà a excepció de fint_mode que tindrà el seu propi registre r8, fint_r8. fint_mode no podrà veure r8 i els altres modes no veuran fint_r8. PC (r15) serà comú a tots igual que CPSR, no així SPSR: spsr_svc, spsr_int, spsr_fint, spsr_undf, spsr_abort (veure figura).

3.1.4. Instruccions

Les instruccions ARM són de longitud fixe de 32-bits, podent ser agrupades en tres grans grups:

- Data processing instructions
- Data transfer instructions
- Control flow instructions

3.1.4.1. Data processing instructions

Les instruccions de processament tenen com a màxim tres camps d'operands (immediats o registres): dos operands origen i un operand resultant de l'operació. Sempre, el primer operand origen i el destí són registres, possibilitant al segon operand origen ser un literal o un registre. A més, és norma en l'arquitectura ARM que el segon operand origen pugui ser pre-operat abans de l'etapa execution del pipeline —abans de l'entrada a l'ALU—. Així doncs, s'aconsegueix que tots els operands que entren a l'ALU tinguin amplada 32-bits.

Els resultats de les operacions són sempre de 32-bits[1]. Addicionalment, totes les instruccions de processament tenen un flag per especificar si volem o no alterar la paraula d'estat amb el resultat de l'operació.

3.1.4.2. Data transfer instructions

Les instruccions de transferència a memòria es produeixen de registre a memòria i viceversa, existint la possibilitat d'instruccions multi-transferència.

Les operacions de transferència simple ocupen un cicle de més en el pipeline, per tant, l'accés a memòria suposa un coll d'ampolla en l'execució ja que retarda en un cicle el flux d'instruccions que transcorre pel pipeline.

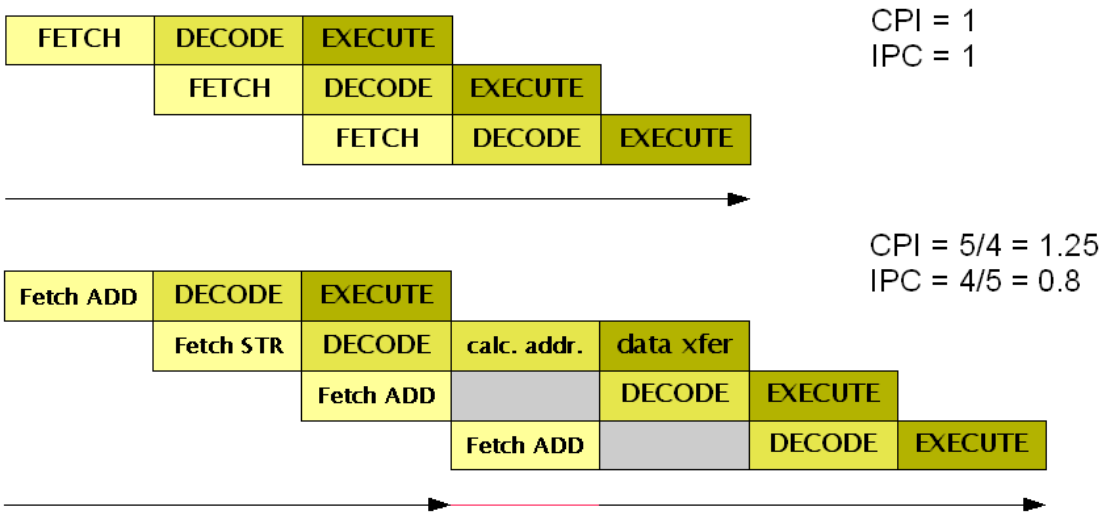


Figura 3-3: Pipeline, Process Instruction vs Transfer Instruction [Fubert00]

[1] Existeix una excepció si el processador incorpora el multiplicador ARM, el qual genera resultats de 64-bits

Les instruccions de multitransferència permeten, a partir de tècniques de pre i post-increment, obtenir una operació més òptima respecte a efectuar múltiples operacions de transferència simples (es guanyen cicles utilitzant calculs a priori).

3.1.4.3. Control flow instructions

Les instruccions de control de flux permeten salts condicionals i incondicionals a diferents posicions d'un programa.

Un salt trenca la dinàmica seqüencial del programa tornant inservible tot procés per avançat del pipeline, obligant un *refill* del mateix pipeline. Per no haver de patir aquesta severa penalització, l'arquitectura ARM fa ús extensiu d'instruccions condicionals al llarg de tot el seu repertori d'instruccions, a excepció de les pròpies instruccions de control de flux.

L'execució d'instruccions condicionals permeten executar una instrucció en base a un dels flags de condició de la paraula d'estat. Si la condició no es compleix, en comptes de fer un salt i trencar la seqüencialitat del programa, es produeix un NOP en l'etapa d'execució de la corresponent instrucció, salvaguardant el pipeline.

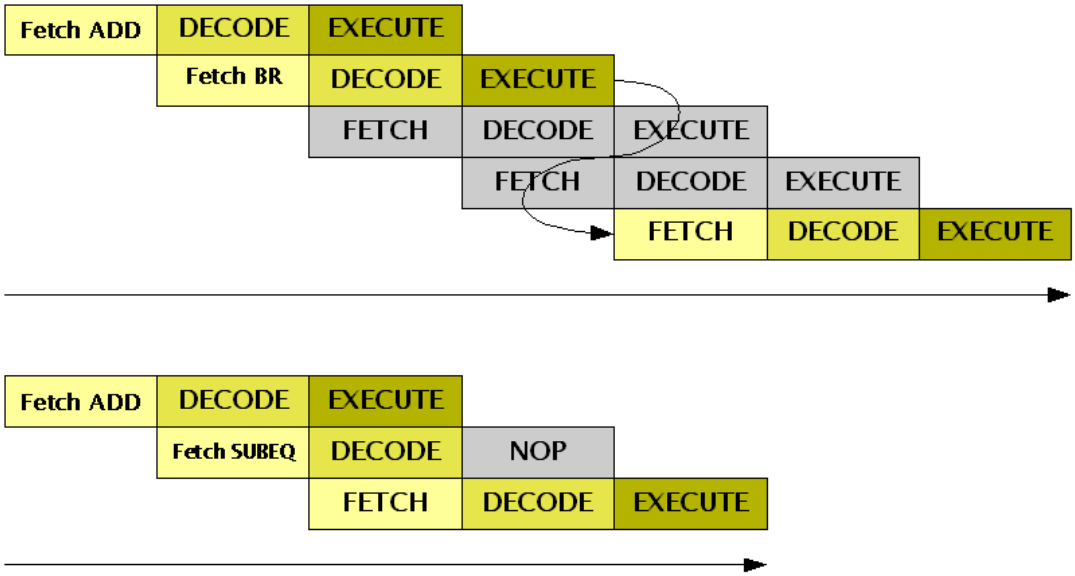


Figura 3-4: Pipeline, Control flow Instruction vs execució condicional d'una instrucció

3.1.5. Instruccions Thumb

L'extensió Thumb va ser introduïda en la quarta versió de l'arquitectura ARM, incorporant amb ella tot un nou repertori d'instruccions.

La finalitat de l'extensió és la reducció de l'espai de memòria ocupat pel codi del programa utilizant instruccions d'amplada 16-bits. Això però, no significa que es faci una emulació 16-bits sobre un processador de 32-bits.

Si el processador treballa en mode Thumb, després de l'etapa Fetch i abans de l'etapa Decode, l'extensió Thumb descomprimirà la instrucció 16-bits, expandint-la a 32-bits. Per tant, tan els operands font i destí com el resultat de l'operació seran valors de 32-bits.

Es pot copsar ràpidament que si la traducció d'instrucció ARM per la instrucció Thumb és directa, l'àrea de codi ocupada en memòria es reduirà a la meitat. Encara que, no sempre serà així ja que el repertori d'instruccions Thumb és més limitat, sent impossible fer una traducció u a u —per una instrucció ARM caldrà més d'una instrucció Thumb—. Precisament per aquest aspecte, tot i que una codificació Thumb ocupa menys memòria, té una execució més lenta per haver d'executar més instruccions. A part, el model de programació en *Thumb mode* només veu la meitat del banc de registres: només són visibles de `r0` fins `r7`.

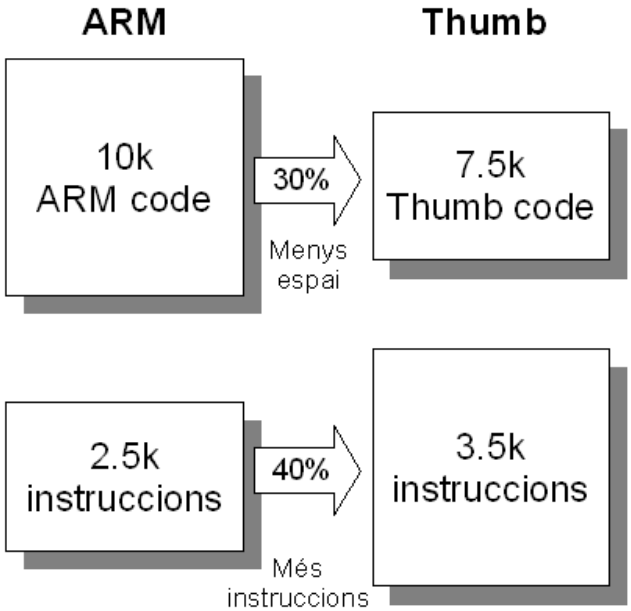


Figura 3-5: Comparativa entre ARM i Thumb code (mesura i nombre d'instruccions) [Olaf07]

Estudis independents han conclòs que el Thumb code ocupa un 30% menys que l'ARM code però conté un 40% d'instruccions.

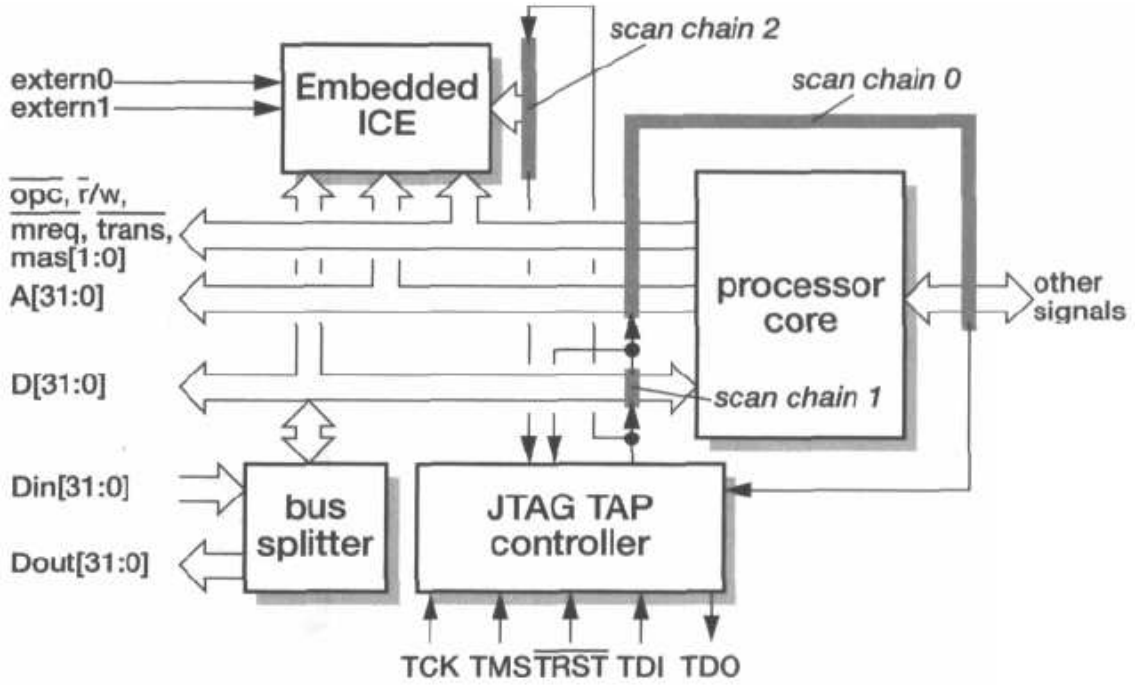
Tot i la pèrdua de rendiment, continua sent rendible, més encara per dispositius embedded on l'espai de memòria continua sent encara un factor a tenir en compte a l'hora de desenvolupar aplicacions software.

No obstant aquest handicap, codificar en instruccions Thumb no vol dir comprometre's a codificar sempre en 16-bits. L'arquitectura ARM contempla l'*interworking* que és la convivència d'instruccions Thumb i ARM en un mateix codi (canvi de Thumb a ARM i d'ARM a Thumb en temps d'execució). D'aquesta forma, les rutines crítiques que requereixen un major rendiment es poden codificar en ARM code; la resta de codi no crític pot fer servir el repertori Thumb per reduir espai en memòria.

3.1.6. ARM7TDMI

L'ARM7TDMI és el processador que incorpora el microcontrolador d'estudi LPC2119 (versió sintetizable), però també ho és de molts altres dispositius, sent un dels processadors més exitosos de la casa ARMLtd. Aquest processador conté un únic core ARM7 amb disseny basat en la quarta versió de l'arquitectura ARM (ARMv4). Per tant, tindrà un pipeline de tres etapes —*Fetch* – *Decode* – *Execution*— amb una taxa una mica inferior a la d'una instrucció per cicle (0.9 IPC – 1.1 CPI).

Junt amb el processador, l'acompanya l'extensió Thumb (T) per descomprimir i interpretar instruccions 16-bits, una interfície JTAG per la depuració on-chip (D) amb un mòdul de suport anomenat EmbeddedICE que permet inserir hw-breakpoints i hw-watchpoint que aturen l'execució i passen a un estat debug; un multiplicador de 32-bits (M) amb resultat de 64-bits (dos registres d'emmagatzematge) i finalment, dues fonts d'interrupcions (I), Fast Interrupt reQuest (FIQ) i Interrupt ReQuest (IRQ).



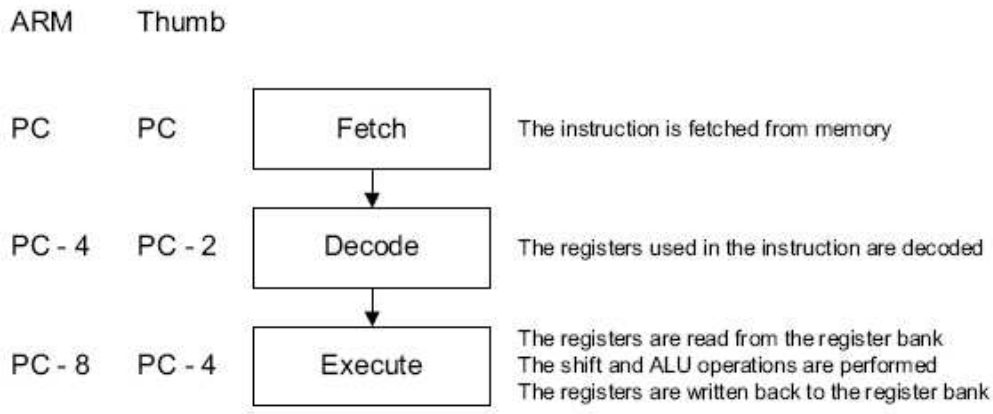


Figura 3-6: ARM7TDMI, diagrama de blocs i pipeline [Fubert00], [ARM05]

ARM Ltd. en comercialitza dues versions:

- **ARM7TDMI:** Macrocel·la optimitzada. El procés tecnològic de fabricació ve definit per la tecnologia de disseny de la macrocel·la.
- **ARM7TDMI-S:** Macrocel·la sintetitzable on es poden afegir modificacions o eliminar extensions.

Un dels avantatges de la versió sintetitzable és la de no comprometre la tecnologia de fabricació del disseny final (disseny del fabricant que compra la macrocel·la). No obstant, aquest grau de llibertat que comporta poder configurar certs paràmetres

del processador i optar a un procés tecnològic concret repercuteix negativament tant en l'àrea final ocupada (50% major) com en el consum (una pèrdua d'un 50% d'eficiència energètica). Tot i així, per empreses com NXP, surt més a compte la llicència de la versió sintetitzable que no haver de supeditar la resta d'elements a la tecnologia de la macrocel·la (probablement, una tecnologia més cara).

3.2. Sistema d'interrupcions

Sabem de l'existència dels dos tipus d'interrupcions. Cada interrupció té la seva corresponent línia de petició —FIQ i IRQ—, on una Fast Interrup reQuest té un tractament més prioritari sobre una Interrupt ReQuest.

A cada una de les dues línies hi estan connectats tots els perifèrics que requereixen mecanisme d'interrupció per avisar al processador .L'assignació de quin perifèric activa una o altra línia es fa a través de software, modificant una serie de registres relacionats amb el sistema d'interrupcions.

També és possible assignar múltiples fonts a una mateixa línia de petició, el tractament però serà diferent per a les FIQ que a per a les IRQ.

3.2.1. Mecanisme Fast Interrupt

Les Fast Interrupt són, com bé indiquen el seu nom, interrupcions ràpides. El temps de canvi de context d'un mode a `fint_mode` té una durada més curta i prioritat major sobre les interrupcions normals.

Per altra banda, podem connectar diferents perifèrics i dispositius per què activin la línia `FIQ` —programant els registre `VICIntEnable` i `VICIntSelect`—. Un cop es detecta l'excepció, es guarda el mode de treball actual i es passa al mode d'excepció `fint_mode`, llançant-se l'única rutina que tracta la interrupció `FIQ`. Si tenim diferents dispositius associats a la línia `FIQ`, dins el cos de la rutina caldrà avaluar, via software amb una estructura `switch-case` o similar, un registre —`VICFIQStatus`— que ens indicarà quins perifèrics han activat `FIQ`. Segons l'ordre d'avaluació estarem marcant la prioritat d'atenció.

Observem que tota la rapidesa que es guanya via hardware amb el mecanisme de Fast Interrupt el podem perdre si assignem masses fonts d'interrupció a la línia `FIQ` pel simple fet d'haver de trobar l'origen de l'excepció.

No és obligatori, però si norma de bon ús associar poques fonts de petició d'interrupció a la línia `FIQ`. D'aquesta manera, no perdem el significat de Fast Interrupt: una excepció ràpida que cal ser tractada amb la màxima prestesa.

3.2.2. Mecanisme d'interrupció

Igual que les Fast Interrupt, les interrupcions són excepcions i com a tals, canvien el mode al seu corresponent, `int_mode`. També, com en el cas anterior, podem tenir diferents fonts d'interrupció associades a la línia `IRQ`; no obstant, si abans no era recomanada aquesta praxis, en les interrupcions normals si que ho és. NXP ha enriquit el mecanisme d'interrupció ARM mitjançant un esquema d'interrupcions vectoritzades.

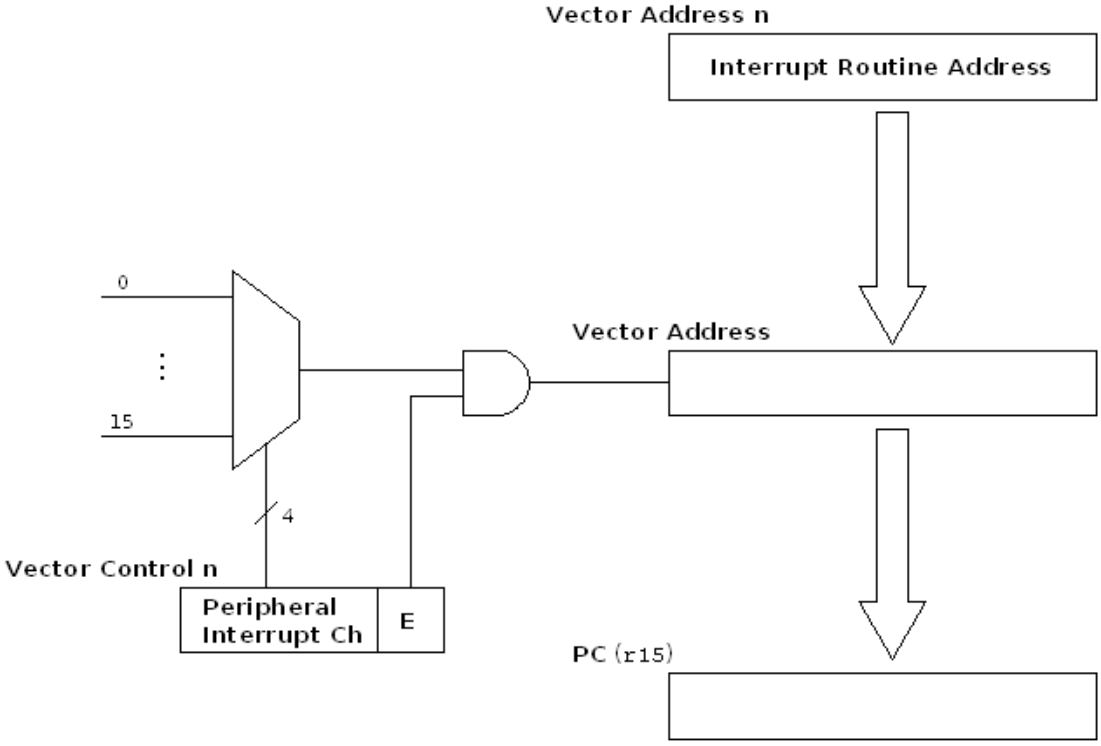


Figura 3-7: Mecanisme d'interrupcions vectoritzades (VIC)

Existeixen setze vectors d'interrupció amb prioritat descendent (`slot 0`, el més prioritari; `slot 15`, el menys prioritari) per resoldre, **via hardware**, l'origen de la petició d'interrupció. A cada vector només se'l pot associar amb una única font d'interrupció —un perifèric— i se l'ha d'habilitar per a que sigui examinat[2]. El procés d'associació d'un perifèric a un vector es produeix configurant una sèrie de registres via software. Cada vector conté una adreça de la rutina de servei d'interrupció a la que ha de saltar en cas d'activar-se una petició del perifèric associat al vector.

A més a més de la taula de vectors d'interrupció, existeix una última opció, *non-vectored interruption*, per associar tots aquells perifèrics que requereixen d'un tractament d'interrupció un cop la taula de vectors té totes les posicions associades —o no—. La rutina de tractament d'interrupcions no vectoritzades només es desperta si existeix una petició d'interrupció i cap dels dispositius associats als vectors d'interrupció l'ha generada; dit d'una altra manera, la rutina de servei de les interrupcions no vectoritzades és l'última que es crida, sent aleshores el servei amb menys prioritat en cas d'existir una interrupció.

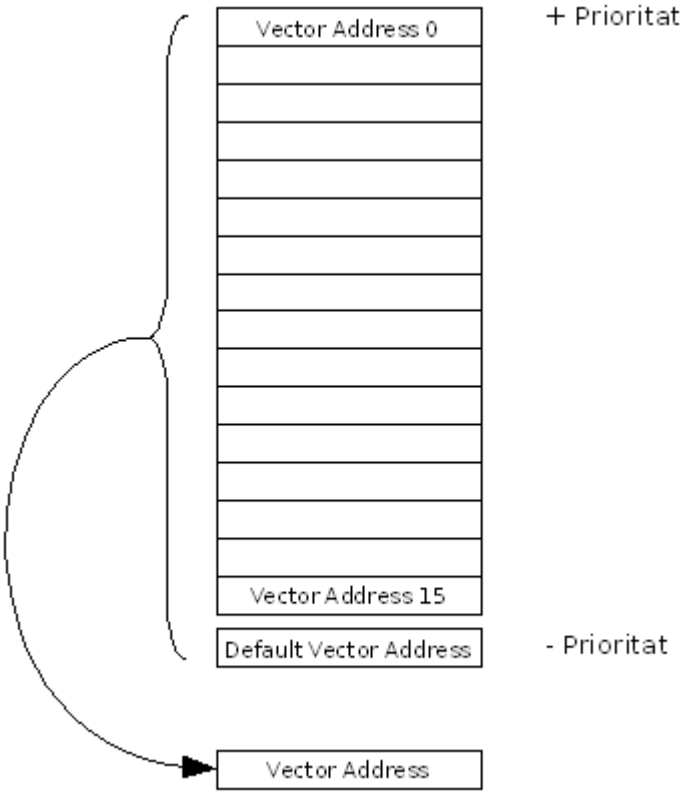


Figura 3-8: Vector d'interrupcions [HIT05]

[2] És possible, però és de ser programador mediocre, associar dos vectors a un mateix perifèric. En aquest cas, el vector d'ordre menor serà qui actuarà

Per la naturalesa de ser una rutina que possiblement hagi de tractar amb varies interrupcions, cal identificar qui ha generat la interrupció, destriant via software (estructura `switch-case`) la font d'interrupció (ídem que les Fast Interrupt).

3.3. Memory Accelerator Module

L'accés a memòria Flash, per part del processador, es fa a través d'una petita caché intel·ligent implementada per NXP, connectant-se al bus local del que disposa l'ARM7, especial i per acoblar memòria. La integració d'aquesta caché té un motiu ben justificat: el microcontrolador pot treballar a una freqüència de 66 MHz. Amb tot, la Flash incorporada només permet una latència de lectura de 20 MHz. El fet de tenir una connexió directe entre Flash i processador provocaria una degradació en el ritme d'execució (per cada fetch d'instrucció es provocaria un *stall* fins a disposar de la dada, trencant tot l'avantatge del pipelining).

Òbviament podríem ubicar el codi del programa en la regió de memòria SRAM del microcontrolador, però hem de pensar que l'SRAM és un recurs escàs, preferiblement utilitzable per altres usos. Així doncs, la funció d'aquesta caché és la de servir una paraula (instrucció) per cada fetch que es sol·liciti.

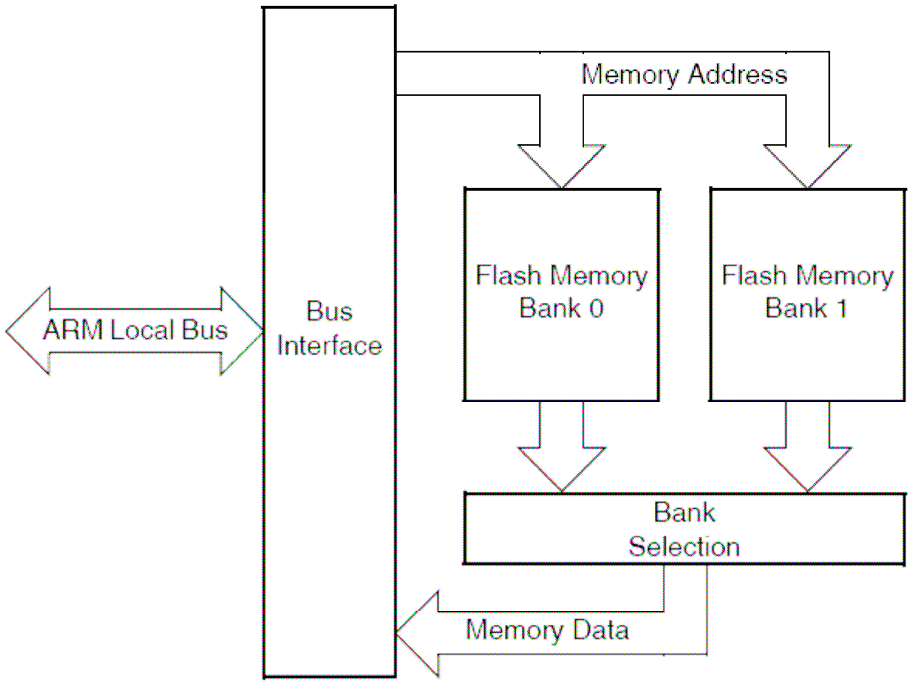


Figura 3-9: Memory Accelerator Module [HIT05]

La seva estructura és la següent: conté dos bancs accessibles de forma independent. Per cada banc, un registre de 128 bits (quatre instruccions ARM o vuit instruccions Thumb) per recuperar línies de la Flash. Resumidament, quan un banc està servint instruccions, l'altre està recuperant la següent línia. Si l'accés és seqüencial (absència de salts en l'execució) els *memory hits* estan garantits. D'altra banda, si es produeix algun tipus de salt, és molt possible que es produeixi un *memory miss (*no es pot garantir ja que el salt es pot produir dins d'una de les direccions que conté la caché).

Existeix també, un registre comú als dos bancs que fa la mateixa tasca, però per recuperació de dades del programa emmagatzemades en la Flash.

Tot i la seva complicació, com és habitual en els sistemes de computació, el mecanisme de caché és totalment transparent al programador/usuari.

3.4. Bus de perifèrics i perifèrics

El processador ARM7TDMI incorpora la interfície del bus AMBA AHB per a la comunicació amb elements aliens al processador.

Tant l'especificació (AMBA) com el protocol (AHB) han estat definits per ARM Ltd., convertint-se ràpidament en un bus molt popular sent de fet, un standard de facto en en sistemes SoC.

Aquesta memòria dista molt d'enunciar i definir les característiques del bus, és més, no és necessari comprendre el seu funcionament per posar en marxa el sistema; només cal saber que en forma part.

Sí és important conèixer que connectat al bus AHB no pengen directament els perifèrics del microcontrolador. NXP introdueix un bridge que desacobla el bus AHB del propi bus de perifèrics. Dues són les raons cap dalts per fer-ho:

- Els perifèrics no poden interferir sobre d'altres elements més rellevants connectats al bus AHB.
- Poder ajustar una freqüència de treball menor (dividir la freqüència) sense que afecti als elements connectats al bus AHB.

Així doncs, del bus anomenat VPB actua d'espina dorsal, d'on hi pengen, ara sí, tots els perifèrics dels que disposa el microcontrolador:

- Dos timers de 32-bits
- Unitat de PWM (6 sortides PWM simples, 3 sortides PWM dobles)
- Quatre conversor A/D 10-bits
- Interfícies I^2C (Fast I^2C)
- Dues interfícies SPI
- Dues interfícies CAN
- Dues UARTs
- Fins a quaranta sis pins d'entrada i sortida (General Purpose Input Output pins). Dotze pins amb funcions d'interrupció externa i/o captura.

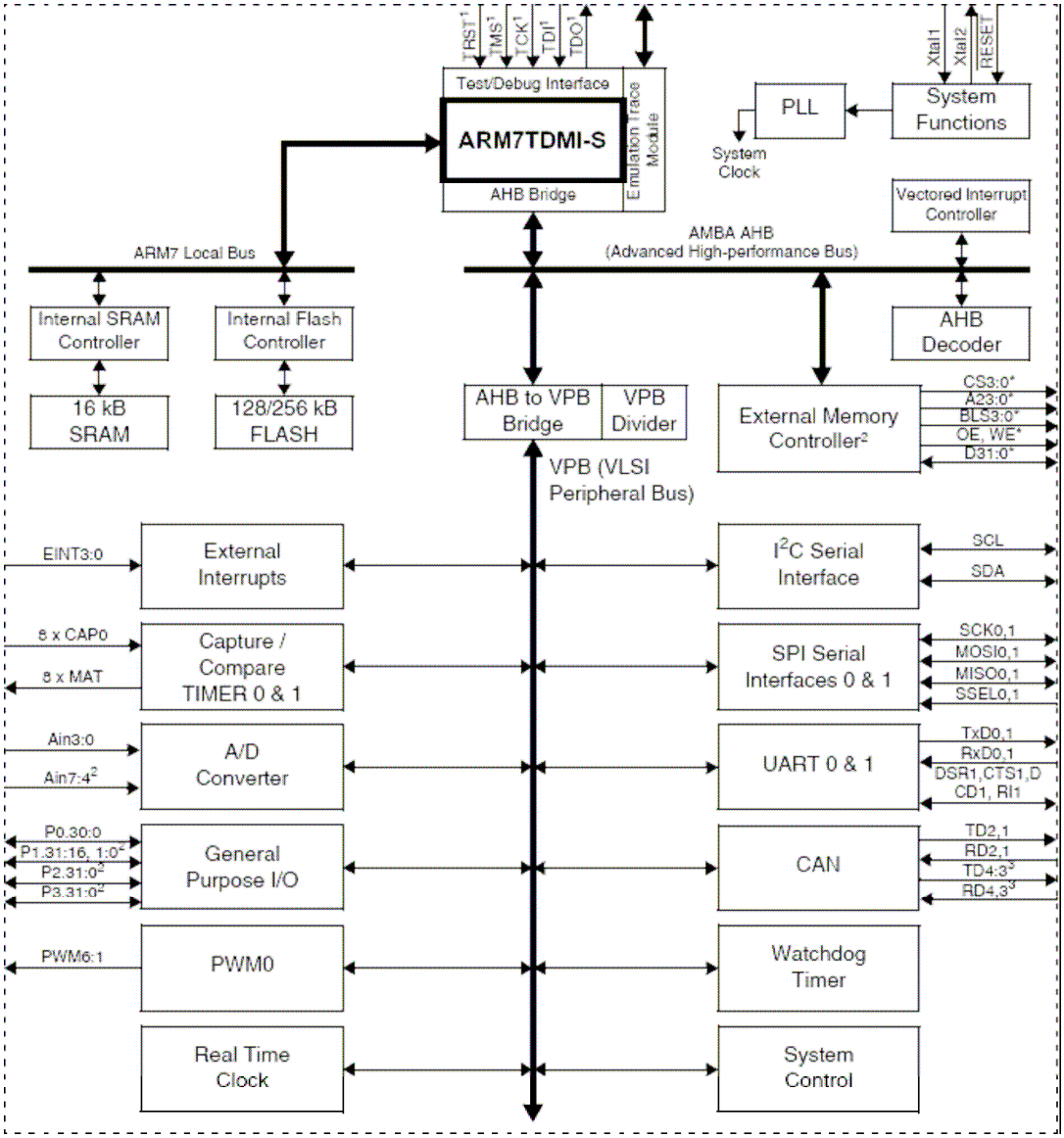


Figura 3-10: Diagrama de blocs del LPC2119 [ARM05]

3.5. Eines de desenvolupament

ARM Ltd. ha procurat disposar d'eines suficients per tal que els desenvolupadors independents puguin confeccionar solucions en base a la seva oferta. Prova d'això n'és l'empresa Keil, germana petita d'ARM Ltd., dedicant-se especialment a proporcionar solucions software de desenvolupament per a dispositius amb marca ARM: compiladors, assemblers, depuradors, RT kernels, llibreries de recursos, etc... No obstant la gran quantitat d'eines, és just fer esment que Keil és una empresa que viu de vendre els seus productes software, independentment d'ARM Ltd., i per tant, molta de la seva oferta està restringida només a aquells qui paguen el preu de les llicències o sota un regim d'avaluació amb opcions limitades.

Existeix un port ARM de GNU (gnuarm) que aglutina tot un conjunt d'eines lliures (toolchain) per a poder desenvolupar amb les típiques eines GNU (gcc, gdb...). És especialment interessant per aquells qui opten només per treballar amb eines lliures o per qui les limitacions de les versions d'avaluació restringeixen el desenvolupament.

Comparatives efectuades determinen que la mida del codi generat entre una solució Keil i una genuïnament GNU varia substancialment en funció del que vulguem generar, observant que per norma, el codi resultant és major en solucions GNU. És comprensible si tenim present que tant les llibreries precompilades com el mateix compilador de Keil són eines ad-hoc d'ARM, enfocades específicament per explotar i optimitzar l'arquitectura.

En el cas particular d'aquest projecte hem optat per utilitzar l'entorn d'avaluació proporcionat per Keil, uVision, que tot i ser una versió d'avaluació, és prou flexible com per dur a terme petits i mitjans projectes sense cap restricció (es limita la generació i depuració d'aquells projectes que el seu .hex sobrepassa un espai màxim fixat en la condició d'avaluació).

L'entorn incorpora editor, compilador, eines de profiling i depurador, permeten durant la depuració, la simulació de perifèrics (Timers, PWM, UARTs, I^2C...).

La posada en marxa d'un nou projecte és molt simple i tant la generació com la depuració no comporta cap complicació afegida a excepció d'aprendre les típiques combinacions de tecles. Conté una ajuda molt completa relacionada tant amb

l'entorn com amb d'altres conceptes (model de programació, repertori d'instruccions, visió general de l'arquitectura...).

Una apreciació personal: uVision de Keil, és ideal per fer-ne un ús docent ja que les limitacions són prou laxes com per mostrar les possibilitats del dispositiu microcontrolador, podent desenvolupar petits exemples pedagògics. Per una projecte de desenvolupament, és més complicat; cal sospesar de quina magnitud és el projecte i fins a quin punt necessitem eines de suport. Per petits i mitjans projectes, tant es bo un entorn comercial com un entorn GNU. A mesura que creix el projecte és possible la necessitat d'adoptar una solució purament comercial, ja sigui per les qüestions de suport en el desenvolupament (pe: simulador) com per la necessitat d'un component software comercial (RTOS, drivers, llibreries...).

4. El pèndul invertit

Un cop apreses les funcionalitats del microcontrolador, passem a un segon terme la base teòrica i juguem amb l'experimentació. L'objectiu és doncs, establir un cas pràctic que posi de manifest les qualitats del dispositiu conjuntament amb una série d'instruments. Ens interessa crear un sistema actiu/reactiu amb una comunicació bilateral sensors-actuadors i sobre tot, amb l'acompliment del temps de resposta dins unes restriccions temporals —temps real—.

L'elecció del pèndul invertit, per sobre d'altres opcions, aglutina totes les consideracions esmentades amb anterioritat. Hi son presents la necessita de sensors —detecció de caiguda— com d'actuadors —moviment de reacció— i és fonamental, ja per la detecció com per la reacció, una resposta dins uns paràmetres temporals[3].

El present capítol enuncia el problema del pèndul invertit, narrant tot el procés de desenvolupament seguit, la solució adoptada i si és o no viables. Com afegit, s'explica el funcionament de les diferents parts acompanyades amb les dades obtingudes de la subseqüent instrumentalització, així com la integració del tot en un sistema.

Finalment voldria fer esment del fet que durant el procés de realització del pèndul invertit, s'han efectuat diversos experiments per extreure conclusions i/o paràmetres. Molts d'aquests experiments, per ser simples proves, no hi són representats en la present memòria, quedant com una feina obscura. Tanmateix, voldria tenir l'oportunitat de deixar-ne constància en aquestes línies.

4.1. El problema del pèndul invertit

Un pèndul invertit d'un sol eix és aquell que té un únic eix lliure on la tija del pèndul oscil·la sobre aquest, traçant una trajectòria semicircular i en posició invertida —l'eix d'ancoratge es troba a sota, produint-se el moviment pendular a la part

[3] Un sistema en temps real no significa necessàriament un sistema d'alt processament de dades. El terme temps real fa referència als sistemes que dins els seus requeriments, garanteixen unes restriccions temporals.

superior—. El problema radica en mantenir el pèndul en la seva verticalitat. Aleshores, per tal de corregir la tendència del pèndul a decantar-se cap un sentit o altre, és necessari col·locar aquest sobre una plataforma mòbil, fent desplaçar el conjunt en el mateix senti de la caiguda.

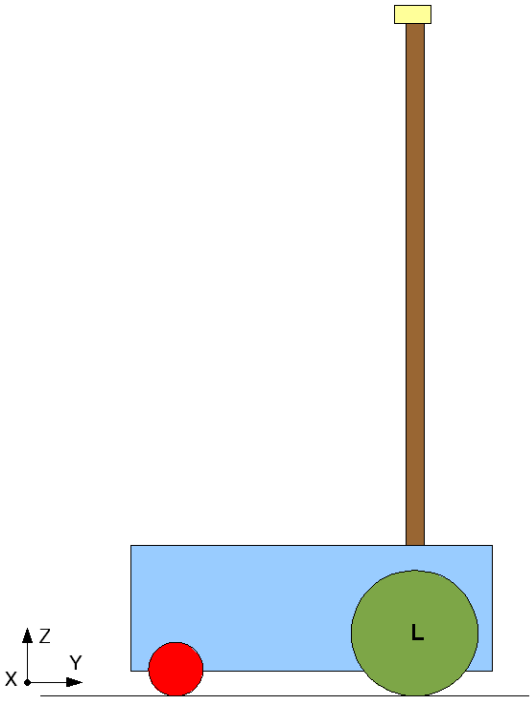
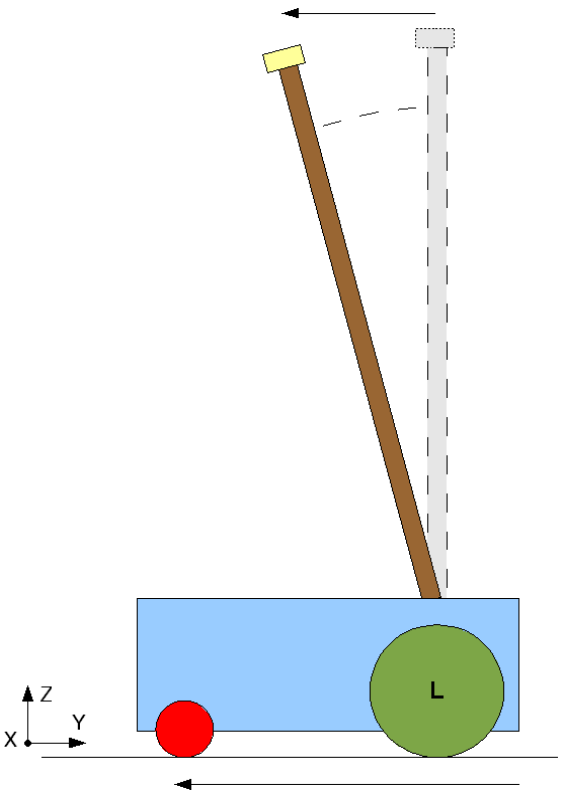


Figura 4-1: Pèndul invertit en equilibri



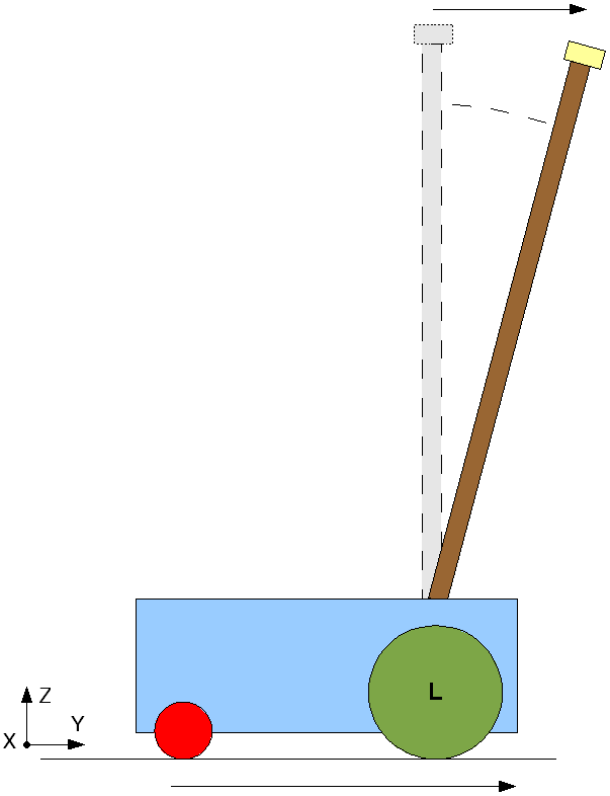


Figura 4-3: Pèndul invertit, caiguda a la dreta

El pèndul és mou sobre un pla que anomenarem *YZ* per ser els dos components que variaran (*x* romandrà sempre constant).

En condicions ideals, el pèndul s'ha de trobar equilibrat sobre la seva vertical, immòbil, incidint tota la força de la gravetat sobre l'eix *z* mentre que l'eix *y*, no n'hi actua cap.

Una petita pertorbació que apliqui un força sobre l'eix *y* durant uns instant, pot desequilibrar el pèndul, fent que es produeixi un trasllat de forces. A mesura que el pèndul cau, la força de la gravetat que incidia al complet sobre l'eix *z*, es traslladarà gradualment a l'eix *y*.

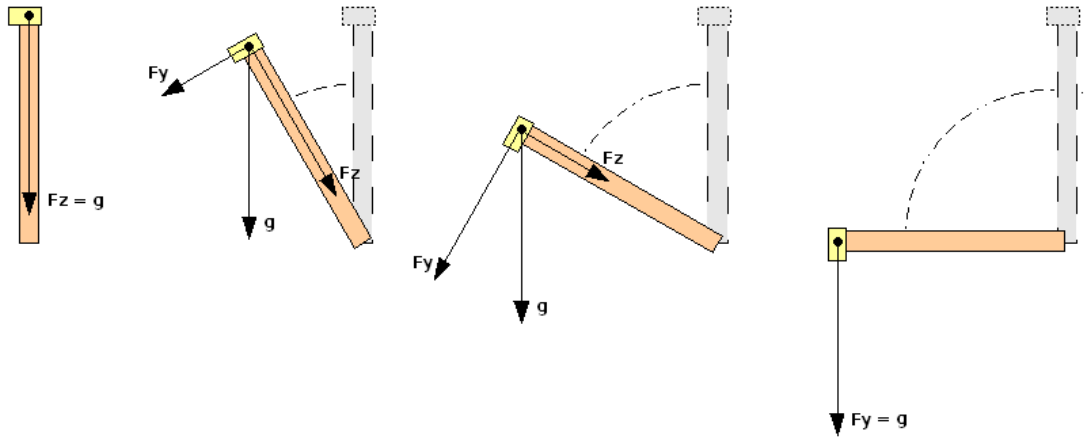


Figura 4-4: Trasllat de forçes de l'eix *z* a l'eix *y*

Just quan el pèndul arribi a la seva horitzontal, el trasllat de la gravetat de l'eix *z* al y serà complet. Podem dir que el pèndul es trobarà absolutament desequilibrar, en un estat insalvable.[4]

4.2. Objectiu

Com és de suposar, l'objectiu d'un pèndul invertit és el d'evitar el seu desequilibri procurant buscar la seva vertical per mantenir-se dret.

[4] Durant una davallada, intervenen altres forces a banda de la gravetat, no obstant, la gravetat és la força més notòria.

Típicament, el problema del pèndul invertit involucra una sèrie de càlculs i equacions que disten molt d'aquesta memòria i dels estudis cursats (és un problema purament físics). Tot i així, la cerca d'una solució a partir de l'observació empírica ens pot dur a una bona aproximació final.

Per tant, amb els coneixements físics elementals i a través de l'observació, provar anar afinant una solució que rectifiqui els moviments de caiguda del pèndul.

Sembla clar que la detecció prematura de la caiguda és una de les claus de l'èxit, ja sigui per tenir un marge d'acció més ample com pel simple fet d'evidenciar que el pèndul es troba en un estat de deriva salvable.

4.3. Realització

La realització del pèndul invertit ha requerit, a banda del microcontrolador LPC2119 com element processador, un dispositiu sensor per a la captura de dades respecte la inclinació del pèndul, més un actuador que, en funció de la inclinació, generi un moviment rectificador.

La captura de les dades s'obté a través d'un acceleròmetre de dos eixos. Amb ell es capturen pertorbacions de forces que desequilibren el pèndul així com la detecció d'una possible caiguda.

Les dades són enviades al microcontrolador, qui les processa i pren una decisió en base a uns resultats.

La resolució afecta directament al comportament del tercer component: l'actuador. Es fàcil suposar que l'actuador és quelcom similar a un motor pel fet d'haver de generar la contra-força per mantenir la vertical de la tija del pèndul. En el nostre cas, utilitzem un parell de servo-motors de corrent continua controlats per modulació d'amplada de pols i muntats sobre una plataforma metàl·lica.

Aquests són els components dels quals vam partir des d'uns inicis per a la realització del pèndul invertit. A continuació exposarem una breu explicació individual de cada component (sensor i actuador) així com els seus respectius modes d'operar.

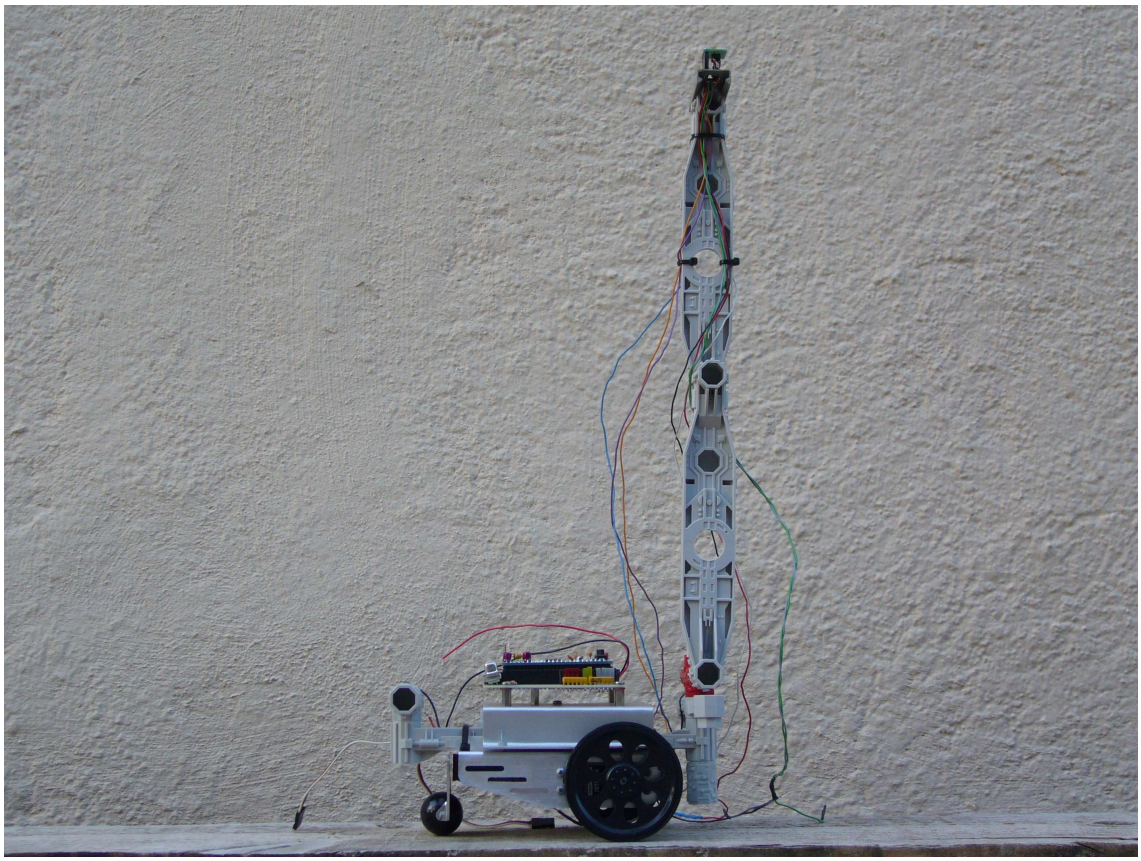


Figura 4-5: *Perfil mestre*. Les rodes motrius queden a la dreta de la bola eix

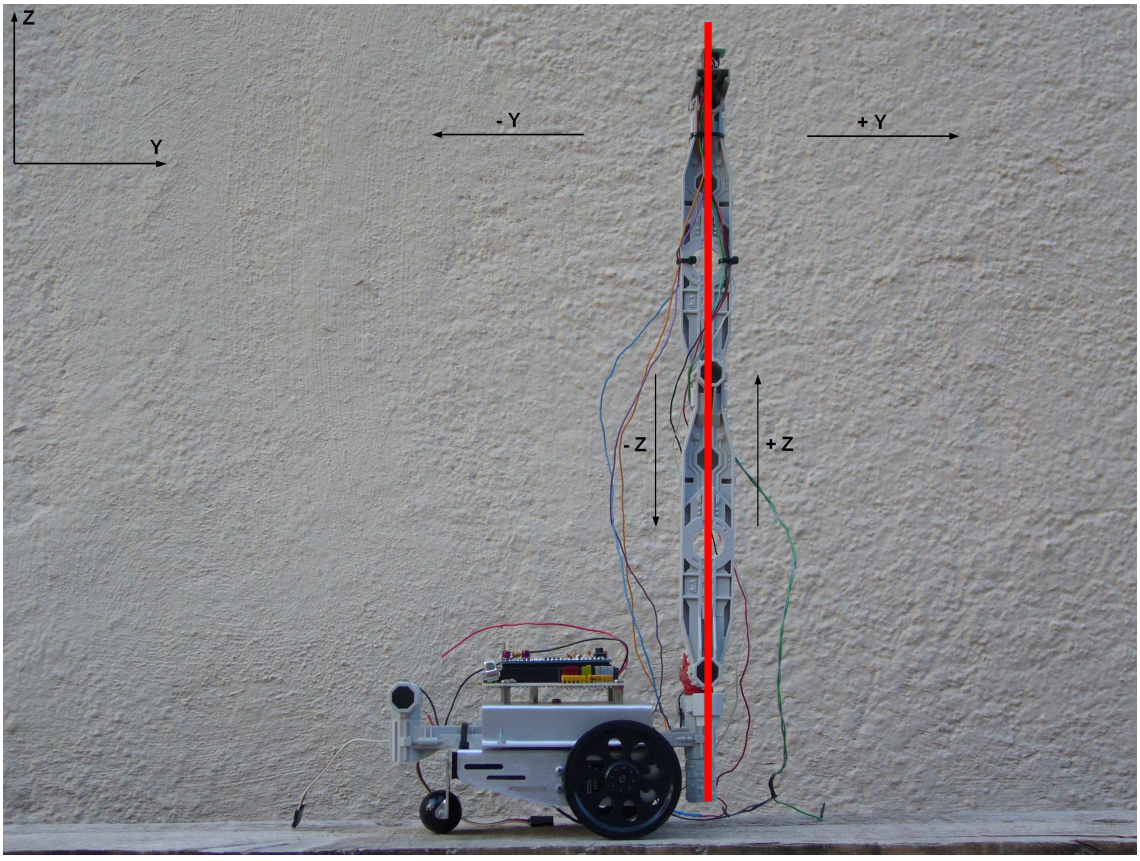


Figura 4-6: *Perfil mestre.* En vermell, la vertical. Dibuixades el sentit de les acceleracions

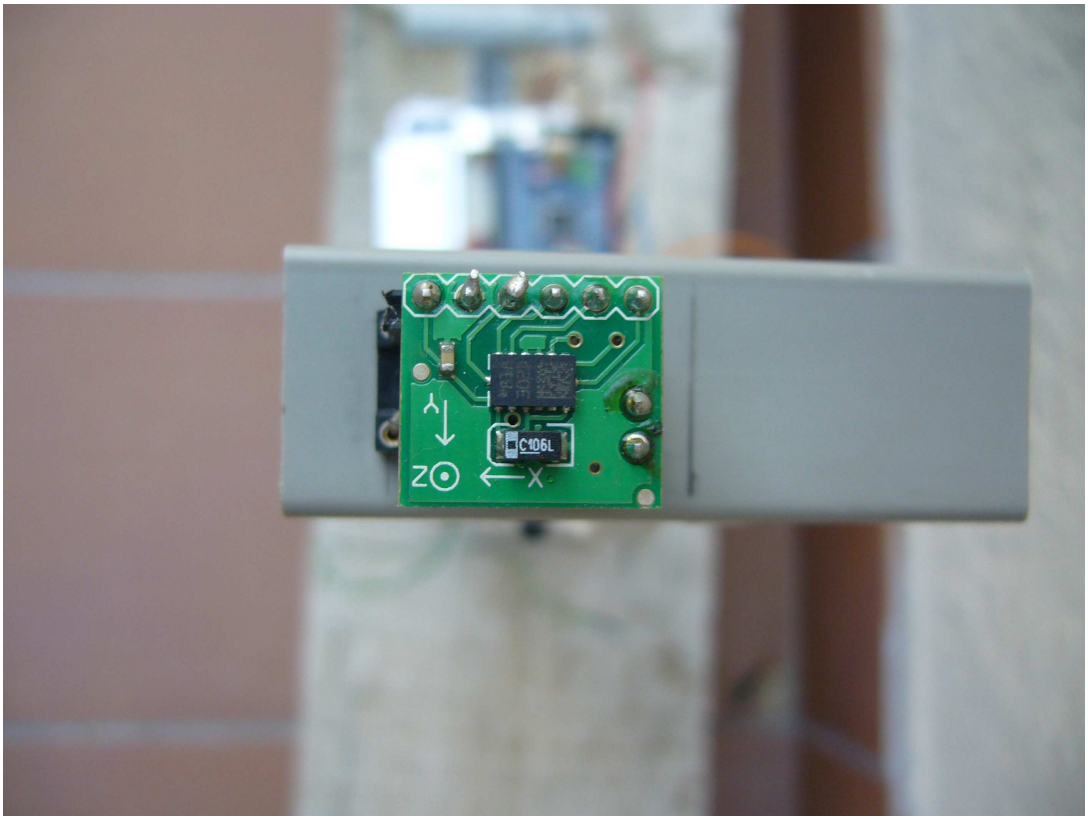


Figura 3-7: Acceleròmetre, vist des de la part superior

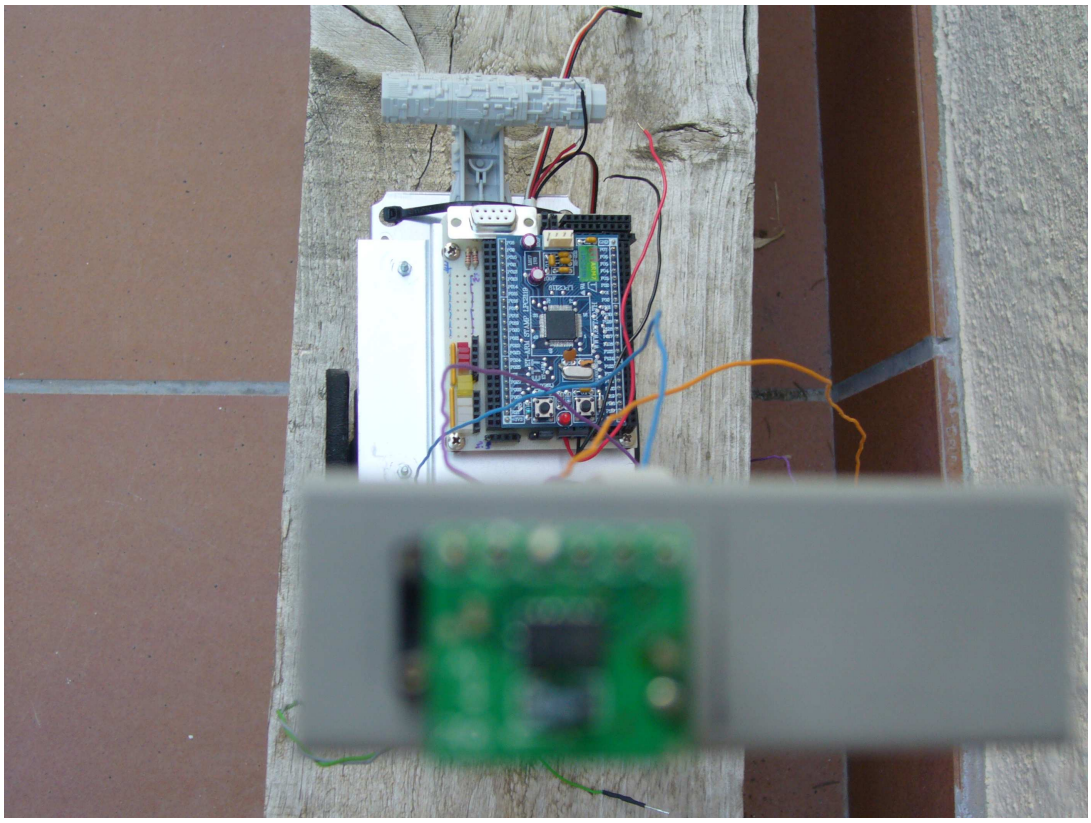


Figura 4-8: Placa amb el MCU LPC2119 muntada sobre la plataforma. Vista des de l'laçada superior del pèndul

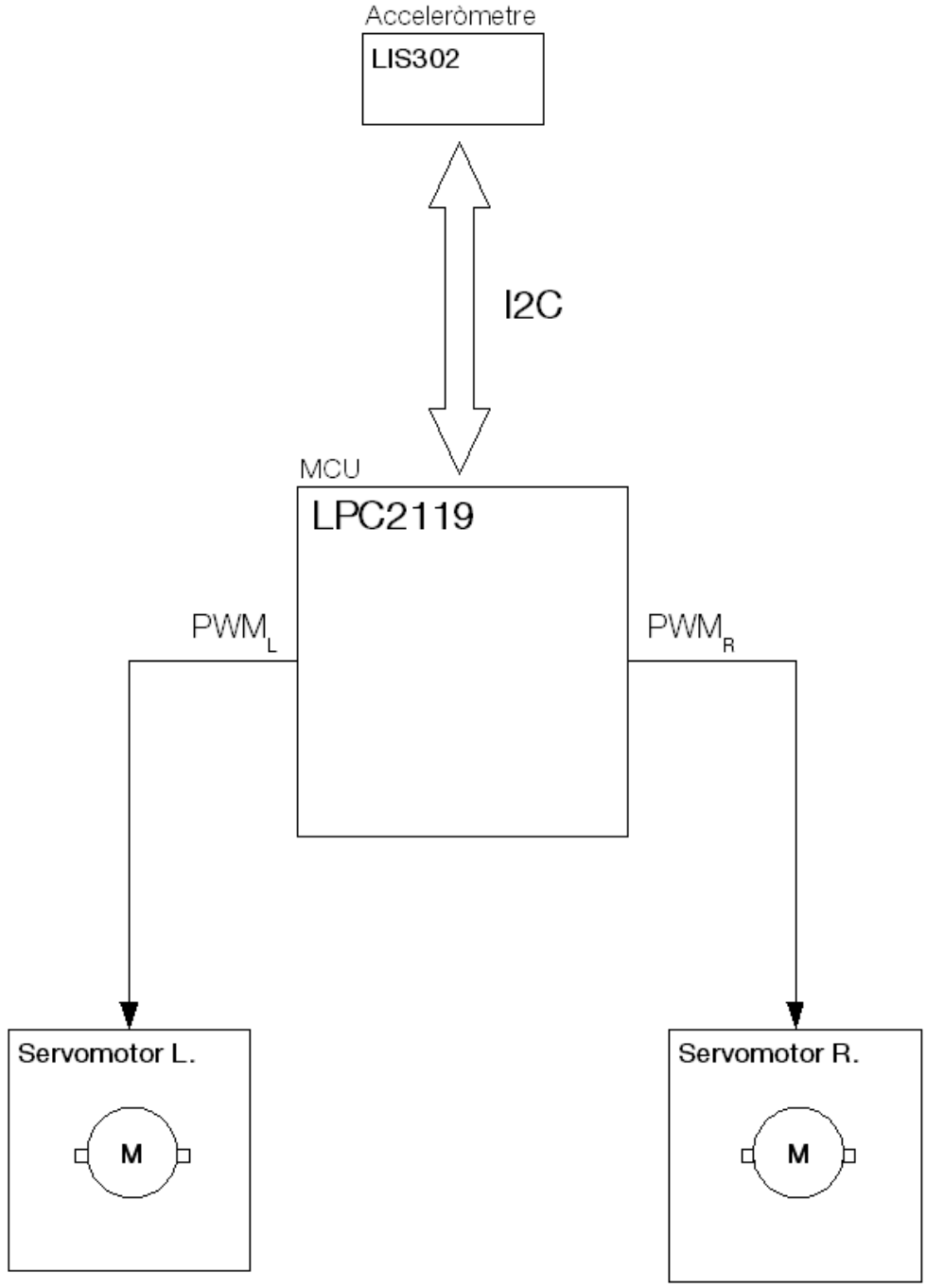
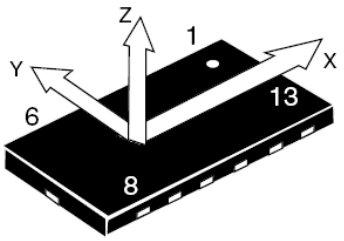


Figura 4-9: Estructura general per la construcció del pèndul invertit. L'acceleròmetre envia les dades a través de la comunicació I2C. El microcontrolador processa les mostres tot duent un procés de control per modular la velocitat d'ambdos servomotors

4.3.1. Acceleròmetre LIS302



L'integrat LIS302 és un acceleròmetre manufacturat per la casa ST. Disposa de tres eixos amb sensibilitat seleccionable de ±2g i ±8g. La transferència d'informació, ja sigui informació de control/configuració com dades, és integrament digital, podent optar entre comunicació I^2C o SPI.

Treballa en un rang de tensions d'alimentació de 2.16V fins a 3.6V i el seu consum típic és menor a 1mW, posseint un mode d'estalvi energètic que permet desactivar de la tensió d'alimentació aquells eixos no utilitzats.

Cada mostra d'un eix ocupa un byte (-128 — 127), proporcionant una nova mostra cada 2.5 ms (*output data rate* 400 Hz en mode Fast I^2C).

Donades les nostres condicions, requerim de dos dels tres eixos disponibles (*y* i *z*, sent constant a 0 l'eix *x*) interessant-nos també per la configuració de l'acceleròmetre amb la màxima sensibilitat possible, per poder detectar com abans millor la caiguda del pèndul. Optem per per seleccionar la mesura en ±2g —típicament 2.3g— que ens proporciona una sensibilitat de 18 mg/digit.

Per la part de comunicació, utilitzem I^2C, concretament Fast I^2C ja que treballa amb una taxa de transferència de 400 KHz.

A més a més, el LIS302 té una línia de sortida a mode de `irq` que avisa de la disponibilitat d'una nova mostra. No obstant això, no utilitzem aquesta funcionalitat ja que el mecanisme d'interrupcions externes de l'LPC2119 és molt sensible, activant el sistema d'interrupcions quan de fet, la senyal rebuda és una senyal paràsita —soroll per el connexionat indegut—. Com a alternativa i sabent que cada 2.5 ms tenim una nova mostra, programem un dels dos timers per a què produeixi un nou cicle de consulta.

4.3.1.1. Experimentació amb l'acceleròmetre

Un dels primeres experiments, un cop configurat l'acceleròmetre i feta la connexió adequada amb l'LPC2119, va ser obtenir les mostres i enviar-les a través del port RS-232 per ser capturades. Guardant els diferents valors en un arxiu, podien ser recuperats i interpretats en posterioritat. D'aquesta manera, el *modus operandi* va ser captura les mostres dels moviments mestres del pèndul:

- Pèndul equilibrat
- Pèndul en caiguda lliure cap a l'esquerra
- Pèndul en caiguda lliure cap a la dreta

per més tard, graficar-los i extreure'n conclusions.

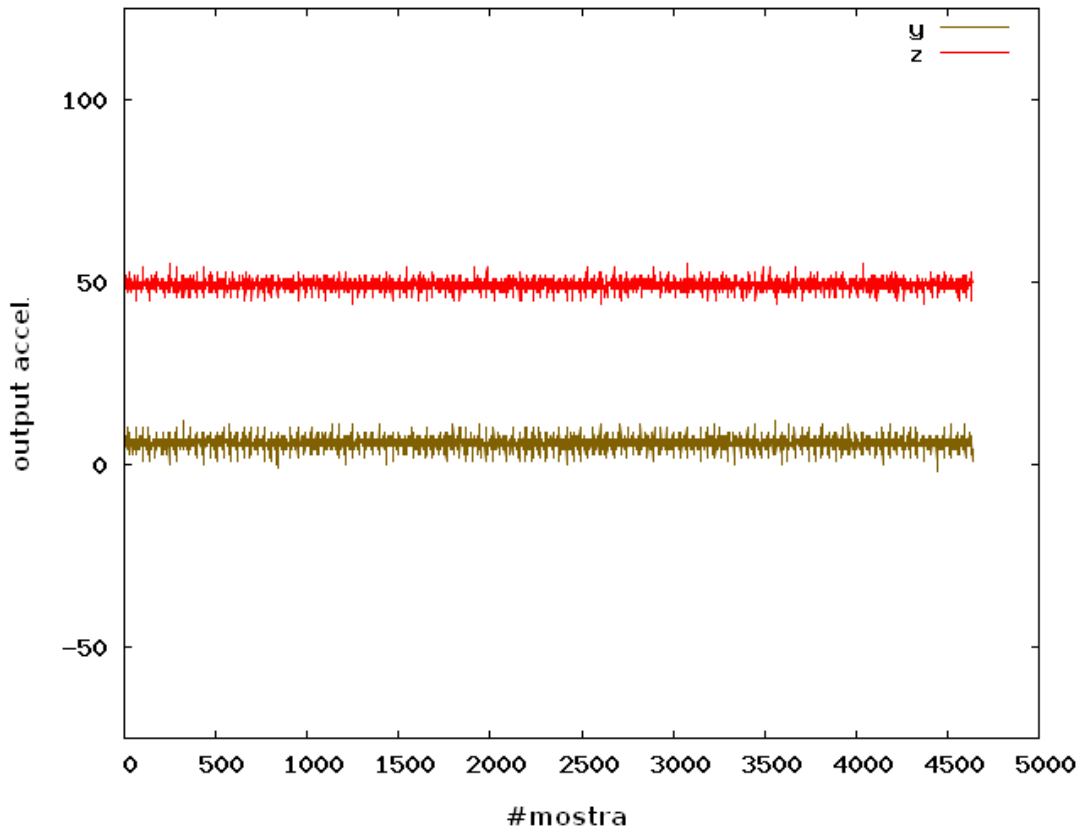


Figura 4-10: Pèndul invertit (PI) en equilibri amb la seva vertical, *PI estàtic.*

La **Figura 4-10** representa gràficament les mostres enviades per l'acceleròmetre quan el pèndul es troba en condició estàtica (**Figura 4-1**).

Paràmetres importants d'aquesta gràfica:

```
Mitjana de l'eix Y:          5.880
Mitjana de l'eix Z:          49.228


Valor màxim de l'eix Y:      12
Valor mínim de l'eix Y:      -2

Amplada del rang Y:          15


Valor màxim de l'eix Z:      55
Valor mínim de l'eix Z:      44

Amplada del rang Z:          12
```

Observem primer que tant l'eix *y* com el *z* tenen un grau d'error, existint una tolerància que ronda els ±7 per mostra-valor d'eix.

L'eix *y*, proper a zero, es troba sobre el pla *XY* on, amb el pèndul en la seva vertical, cap força hi és aplicada. Contràriament, sobre l'eix *z* incideix completament la força de la gravetat en tot moment. Amb una sensibilitat 2g, típicament amb màxim mesurable d'uns 2.3g, l'acceleròmetre retorna el valor 49 per 1g. Aquest resultat concorda amb la comprovació del calibratge de l'acceleròmetre:

$$127 \times (1g / 2.3g) = 55$$

$$(55 - 7 = 48) < 49 < (62 = 55 + 7)$$

En tot cas, el valor 49 és molt proper a l'error. Això evidencia el fet que el pèndul, tot i mantenir-se estàtic, pot no trobar-se en la seva vertical, estant uns graus desviat —inclinat— cap un dels costats. Aquesta desviació contribueix a desdibuixar un mica el valor esperat, però —tenint present aquest factor— no ha de suposar cap problema.

Mostrem ara com són les gràfiques d'una caiguda lliure[5]; com, el pèndul passa d'un estat estàtic a perdre la verticalitat degut a un petit impuls sofert.

[5] Prenent com a referència el perfil mestre, **Figura 4-5**

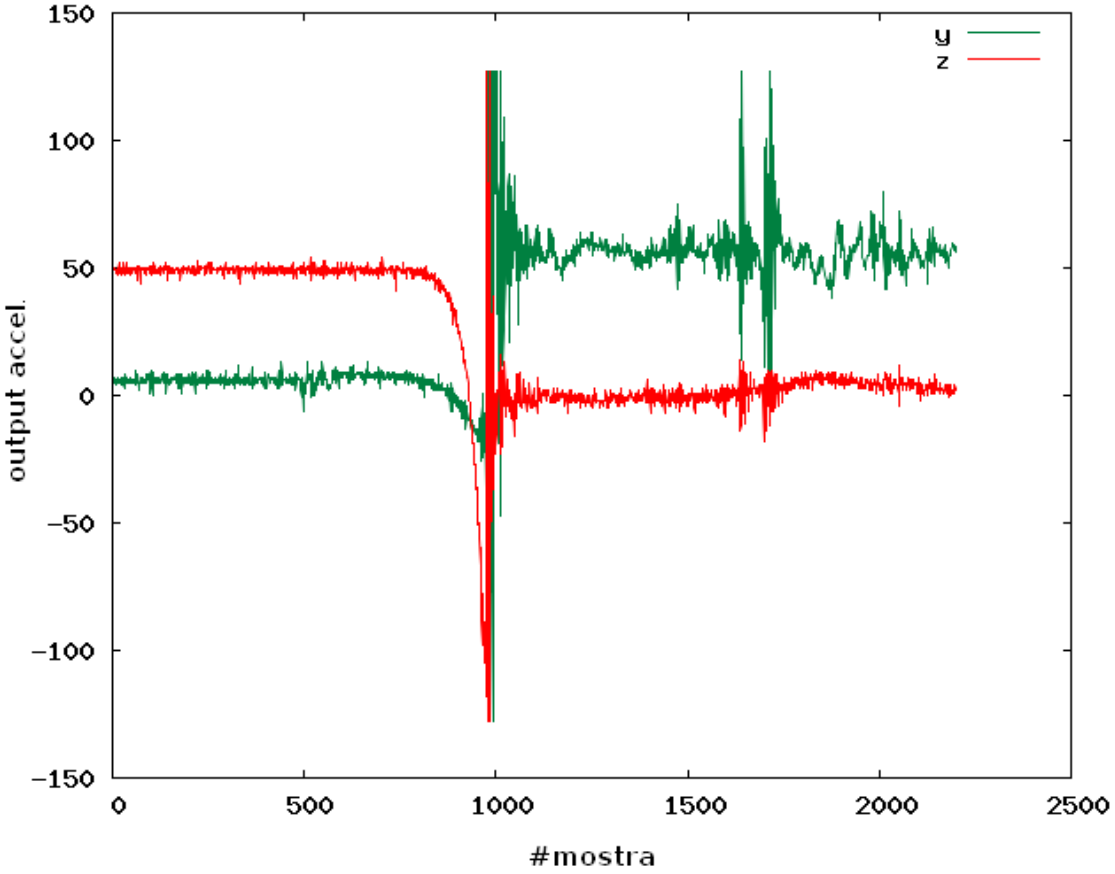


Figura 4-11: Pas d'estàtic a deriva. Caiguda lliure a **l'esquerra**.

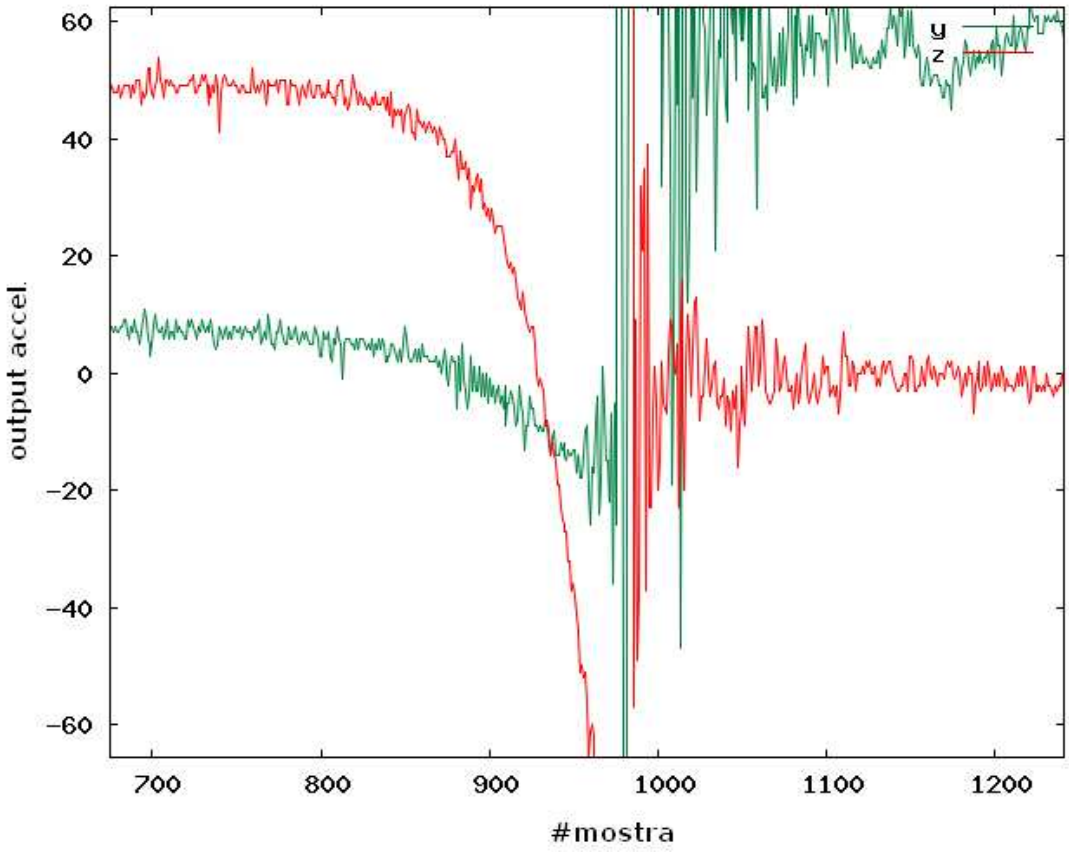


Figura 4-12: Ampliació del moment de la caiguda (**Figura 4-11**)

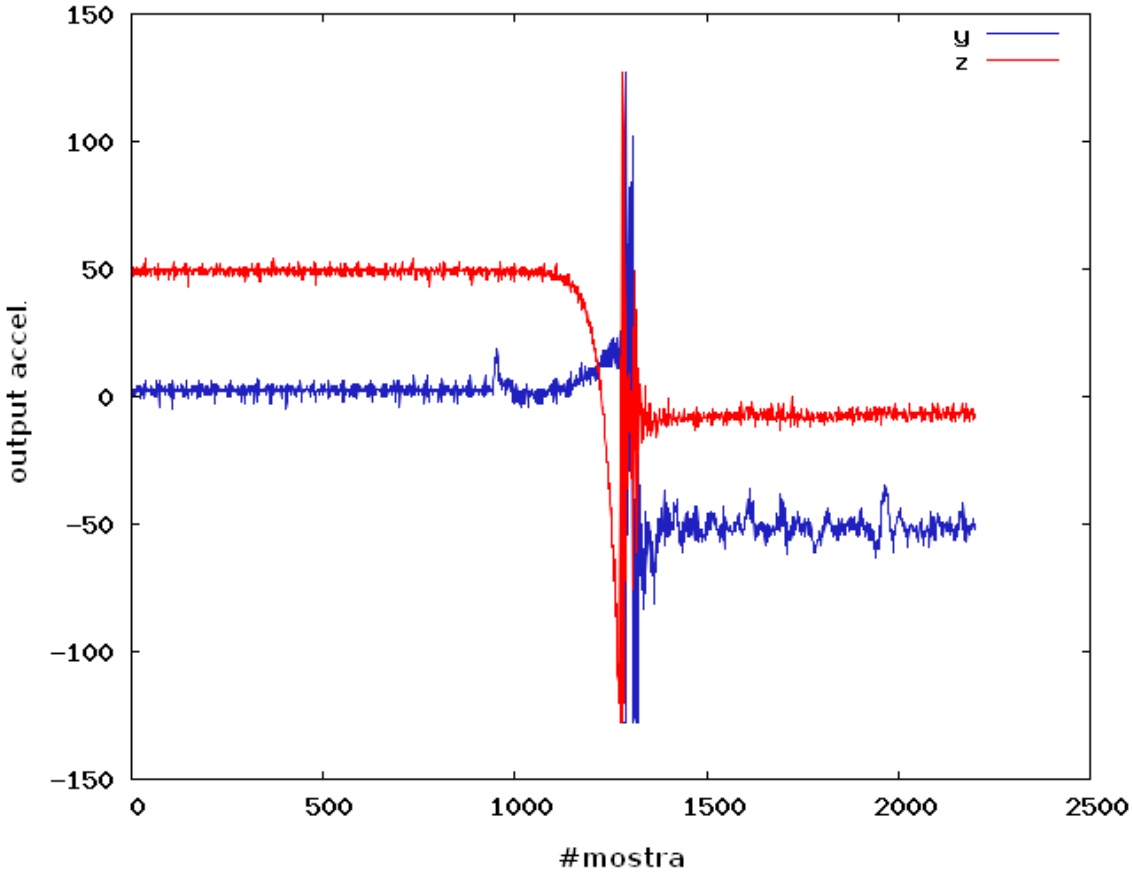


Figura 4-13: Pas d'estàtic a deriva. Caiguda lliure a **la dreta**.

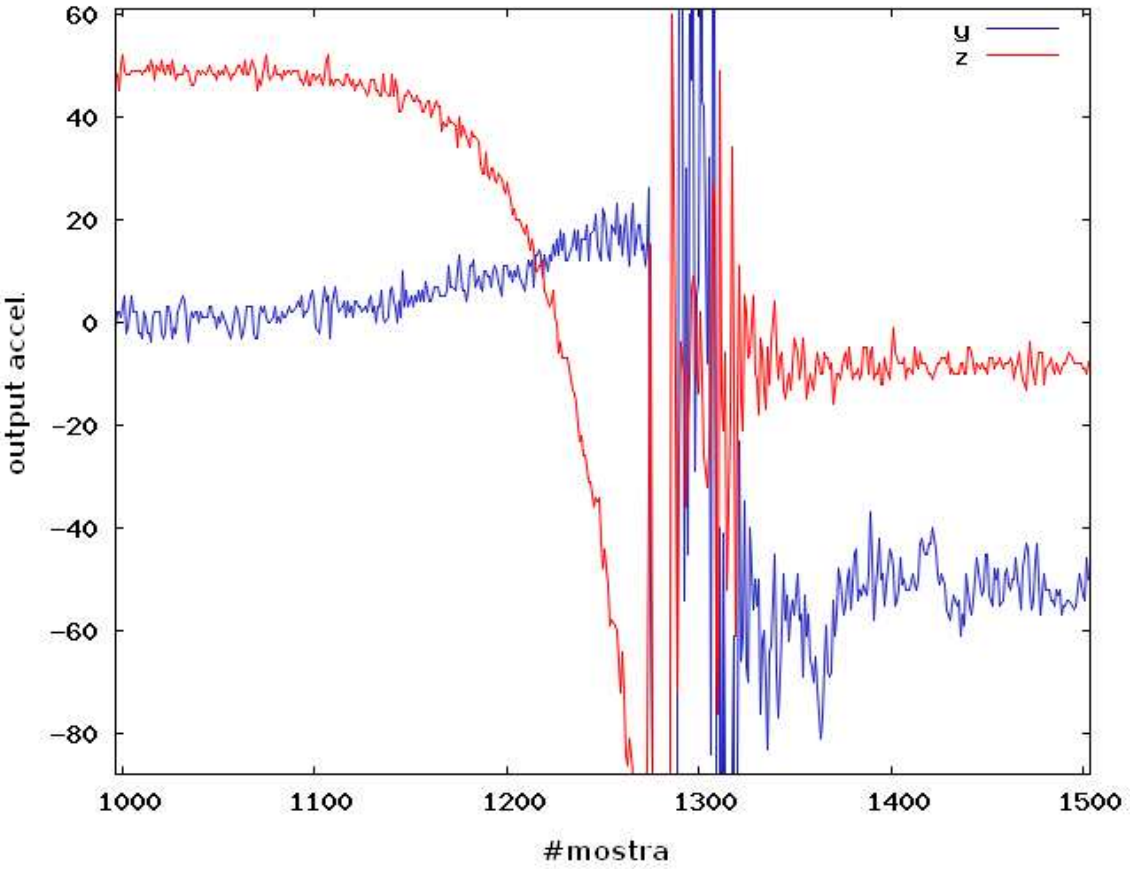


Figura 4-14: Amplició del moment de la caiguda (**Figura 4-13**)

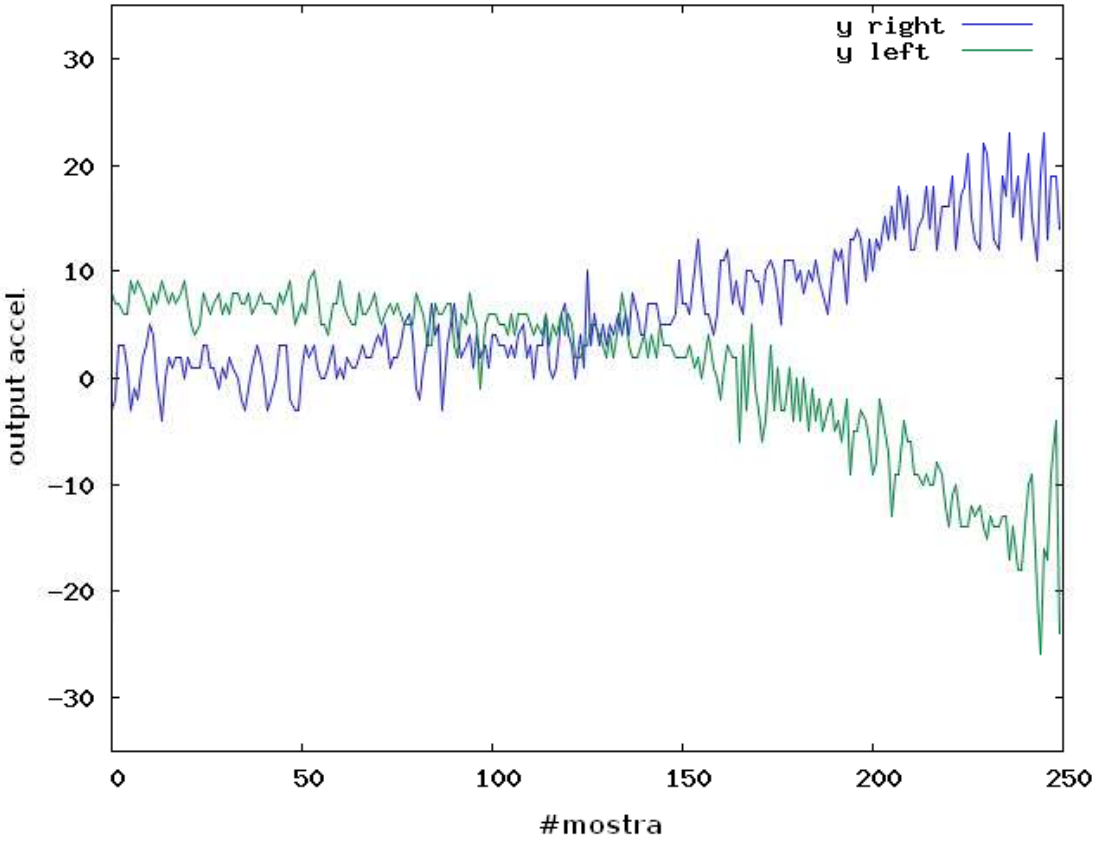


Figura 4-15: Eix *y* **Figura 4-11** (caiguda esquerra) i eix *y* **Figura 4-13** (caiguda dreta)

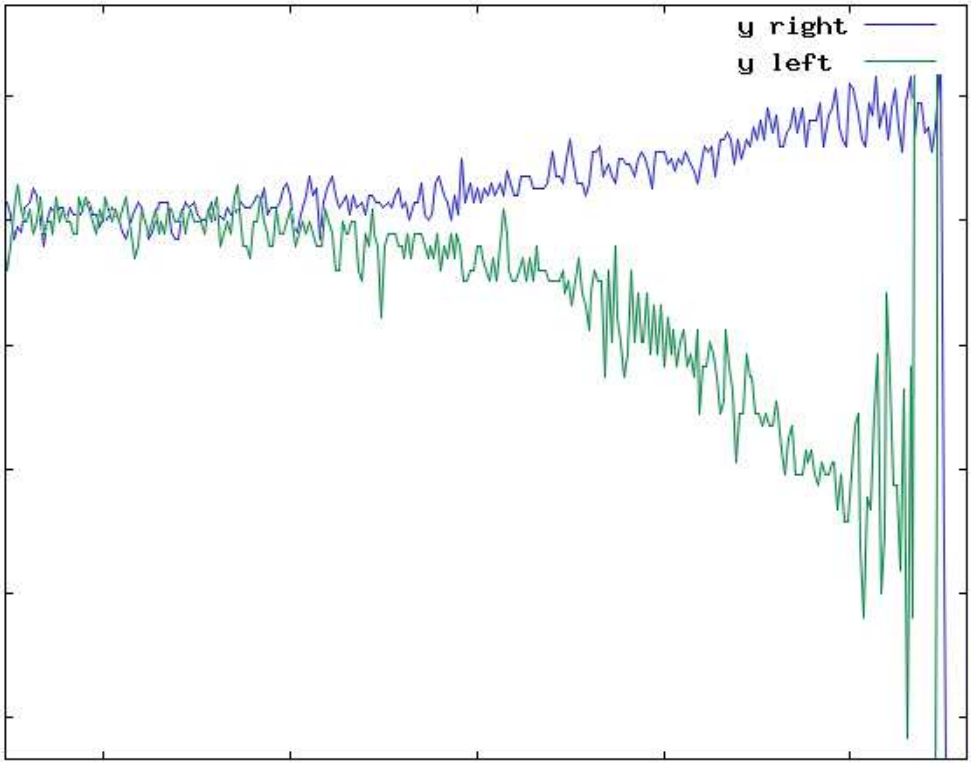


Figura 4-16: Adaptació d'una de les gràfiques per apreciar millor les tendències

Dades corresponents a l'estat estàtic —mostres de 0 a 400— de la **Figura 4-11** (caiguda a l'esquerra):

```
Mitjana de l'eix Y:          5.820
Mitjana de l'eix Z:          49.038


Valor màxim de l'eix Y:      12
Valor mínim de l'eix Y:      0

Amplada del rang Y:          13


Valor màxim de l'eix Z:      53
Valor mínim de l'eix Z:      44

Amplada del rang Z:          10
```

Dades corresponents a l'estat estàtic —mostres de 0 a 400— de la **Figura 4-13** (caiguda a la dreta):

```
Mitjana de l'eix Y:          2.487
Mitjana de l'eix Z:          49.140


Valor màxim de l'eix Y:      7
Valor mínim de l'eix Y:      -5

Amplada del rang Y:          13


Valor màxim de l'eix Z:      54
Valor mínim de l'eix Z:      43

Amplada del rang Z:          12
```

Sempre representat amb el color vermell, l'eix z decau en ambdues representacions, passant d'un valor estacionari 49 abans de la caiguda, a un valor proper a 0 després de la caiguda, un cop el pèndul ha caigut i es troba en repòs, completament en horitzontal.

Tant si el pèndul es decanta cap a la dreta com si ho fa en sentit contrari, el comportament de l'eix *z* és el mateix, per tant, per diferenciar el sentit de la caiguda és obligatori contemplar l'eix *y*.

Prenent com a referència el *perfil mestre* (**Figura 4-5**), una caiguda del pèndul cap a l'esquerra, representació de l'eix *y* de la gràfica (**Figura 4-11**), mostra una caiguda dels seus valors just en els instants que es produeix el descens. Els valors de la caiguda estan compresos entre la mostra 700 i 1000, visualitzant-se més clarament en l'ampliació de la gràfica, **Figura 4-12**. Encara que, no tant evident com el canvi de l'eix *z*, sí és perceptible i remarcable (20 punts per sota del valor en estàtic).

Comprovant ara la gràfica per una caiguda a la dreta (**Figura 4-13**), observem que el comportament de l'eix *y* és anàleg però en senti contrari; durant el breu instant de la caiguda, els valors de l'eix creixent respecte el seu valor en estacionari. Més detalladament, la **Figura 4-14** comprèn l'instant de la caiguda, identificant-se una moderada tendència al l'alça dels valors d'*y* durant la davallada de *z*.

Aleshores ens podríem preguntar, ¿quina necessitat tenim d'utilitzar dos eixos de l'acceleròmetre quan amb un de sol —eix *y*— és possible detectar la caiguda i identificar el sentit de la mateixa? És absolutament cert, únicament amb l'eix *y* ens basta per la construcció del pèndul invertit. Tantmateix, nosaltres fem servir tant l'*y* com el *z*. El motiu radica en el comportament d'ambdós durant la caiguda; si bé, l'eix *y* varia durant el breu instant que dura la caiguda —aproximadament 300 ms—, no ho fa d'una forma prou marcada com ho fa *z*.

Així doncs, fem servir *z* per identificar una caiguda i *y* per determinar-ne el sentit. Cada eix tindrà un tractament individual per accentuar precisament aquells aspectes importants per a la detecció prematura d'una deriva (paràmetres del control PID individuals per eix).

4.3.2. Servomotors

El principi d'una devallada del pèndul significa la presència d'una força que incideix sobre l'eix *y*. Sembla raonable aplicar una força, de com a mínim igual magnitud però en sentit contrari, amb el propòsit d'anul·lar la primera.

Una de les maneres de contrarestar dita força és efectuar un moviment de la plataforma on és muntat el pèndul cap el mateix sentit de la caiguda per tal de transmetre un força rectificadora en sentit contrari que recuperi la verticalitat de la tija. Per tal d'aconseguir-ho, utilitzem un parell de servomotors de corrent continua controlats per modulació per amplada de pols (PWM). Aquests motors només requereixen de tres senyals: alimentació, terra i senyal de control provinent del microcontrolador.

El seu principi de funcionament és molt bàsic. Existeixen dos paràmetres essencials:

- T: període del tren de polsos. Paràmetre constant.
- t_h: semiperíode alt. Paràmetre variable

La modulació per ampla de pols consisteix en enviar un tren de polsos amb període constant (T), però amplada de semiperíde alt variable (t_h). El període constant T és la referència temporal per distingir els polsos mentre que el control de la velocitat i el sentit de gir es fa respecte la durada del semiperíode alt —modulació en funció de t_h—.

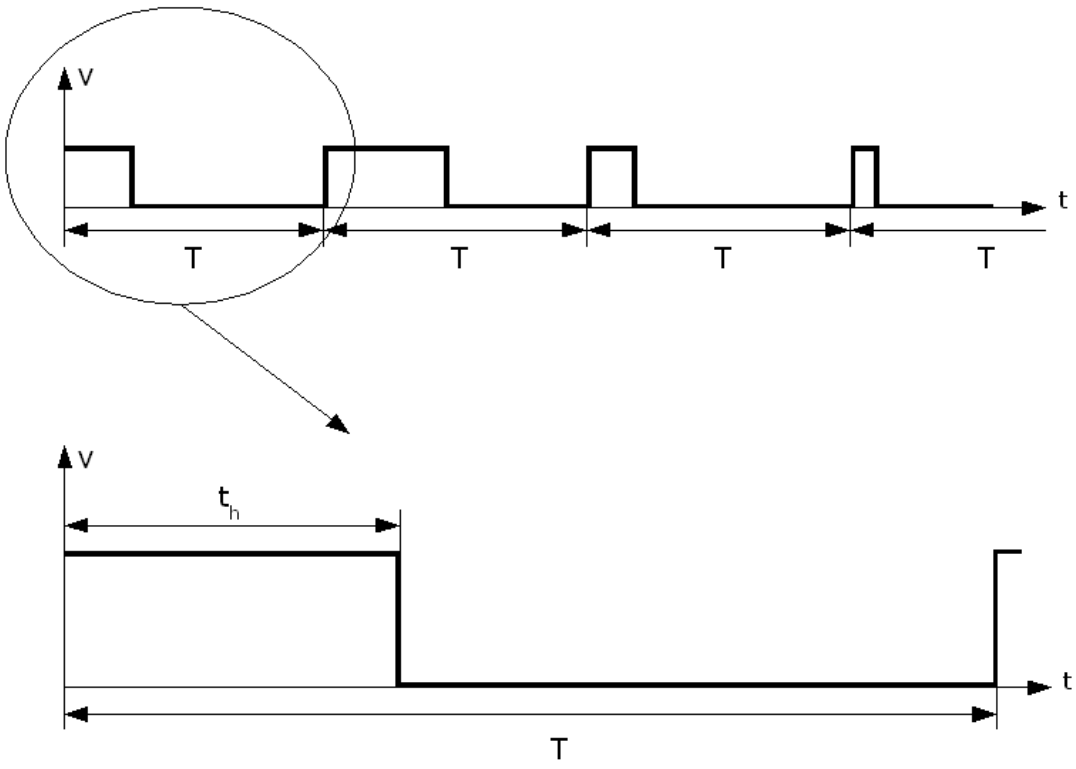


Figura 4-17: Exemple de tren de polsos per una modulació per amplada de pols (PWM)

En el cas dels nostres servomotors, considerant com exemple un servomotor ideal de característiques similars, tindrà un període T de durada 20 ms, trobant-se el *pas elemental* de la modulació (*pe*) en 1.5 ms. El pas elemental és la referència temporal per el paràmetre t_h; el servomotor amb un t_h igual a *pe* no girar en cap sentit, és manté quiet.

Quan el semiperiode t_h es troba més enllà de *pe* (t_h > *pe*) l'eix del servomotor gira vers un sentit. Com més distant és t_h de *pe* (t_h >> *pe*) més gran és la velocitat de gir. En cas contrari, quan el semiperiode t_h és més petit que *pe* (t_h < pe), el sentit de gir és invers a la situació anterior. Igual que abans, com més distant es troba t_h de *pe* (t_h << *pe*) més s'incrementa la velocitat de gir en aquest sentit (**Figura 4-18**[6]).

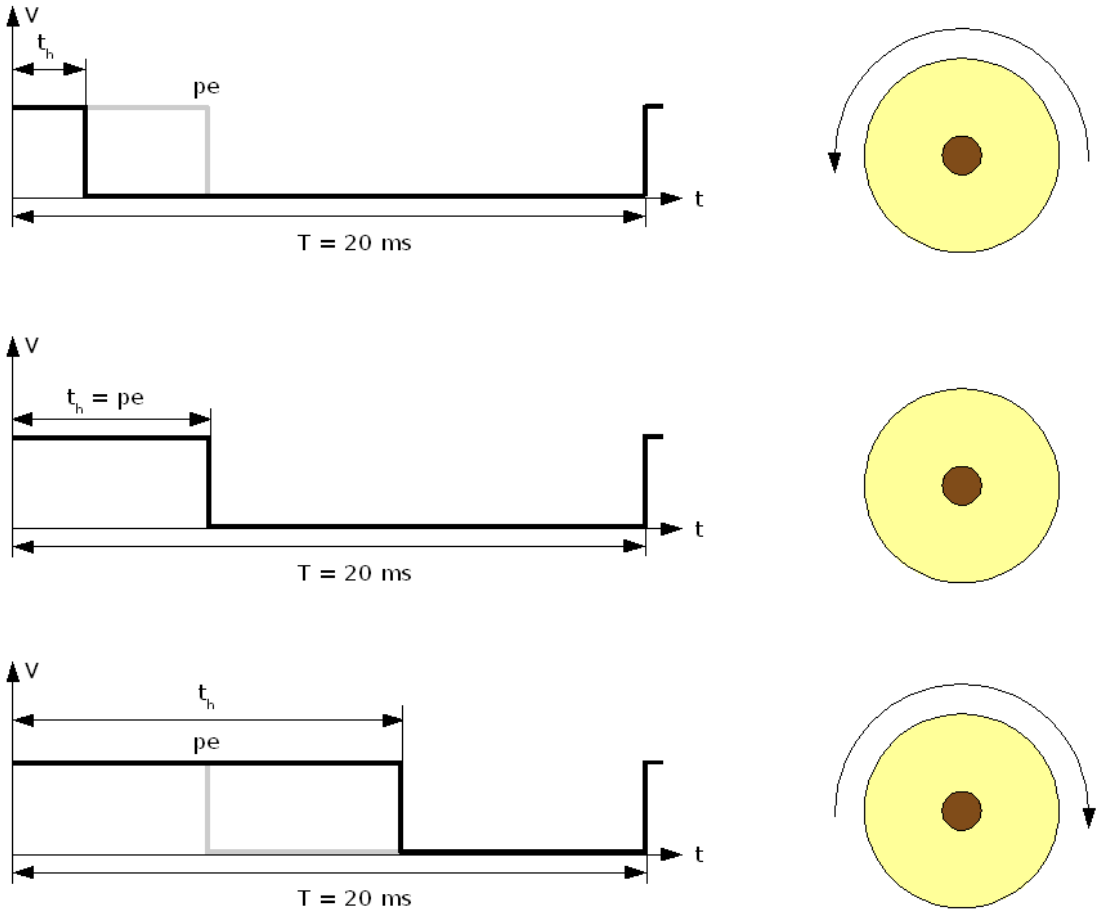


Figura 4-18: Control de gir i velocitat mitjançant PWM

[6] La **Figura 4-18** no està a escala segons els parametres esmentats, pe no mesura 1.5 ms.

4.3.2.1. Experimentació amb els motors: *pas elemental*

Sense un *datasheet* amb l'especificació tècnica dels servomotors, cal esbrinar una sèrie de dades respectives a cada servomotor per separat.

Sabem que la modulació per amplada de pols dels servomotors requereix d'un període T de 20 ms, i que el pas elemental *pe* d'ambdós ronda el t_h = 1.5 ms —condicions ideals—. A la pràctica, aquest pas elemental varia en cada servomotor. Per tant, el primer pas és esbrinar el *pe* individual de cada servomotor.

Aplicant el test a cada servomotor per separat, la metodologia consisteix en enviar inicialment un tren de polsos $t_h = 1.5 + \lambda$ ms (t_h lleugerament superior al *pe* teòric) durant un espai de temps conegut, suficientment llarg per observar el comportament de la roda associada al servomotor. Passat aquest espai de temps, decrementar t_h en una constant i tornar a repetir el cicle d'espera per observar de nou el comportament. D'aquesta manera, per a cada iteració, anem acotant més i més el rang de valors possibles de *pe*. Finalment, quan la roda atura el seu moviment de rotació, hem esbrinat un dels valors pels quals el servomotor no gira, valor que pertany al *rang de possibles valors de pe*.

```
Algorisme test_pe(ref rotacio, ref pe)
{
      constant T := 20 ms;
      constant lam := 1 us;
      var th := 1600 us;

      do
      {
            PWM(th,T);
            th := th – lam;
            wait 5 s;
      } while(rotacio > 0);      // Aturem quan no rotació

      pe := th + lam;
}
```

Arribats a aquesta situació, el següent pas és esbrinar l'amplada del *rang de possibles valors de pe* seguint amb el mateix mode d'operació: decrementar t_h en una unitat i observar durant un temps. Efectuar tantes iteracions fins que s'inicïi un moviment de rotació, aleshores, ens aturem.

Un cop acotat per davant i per darrera el *rang de possibles valors de pe*, la determinació de *pe* és tant fàcil com agafar el valor central del rang.

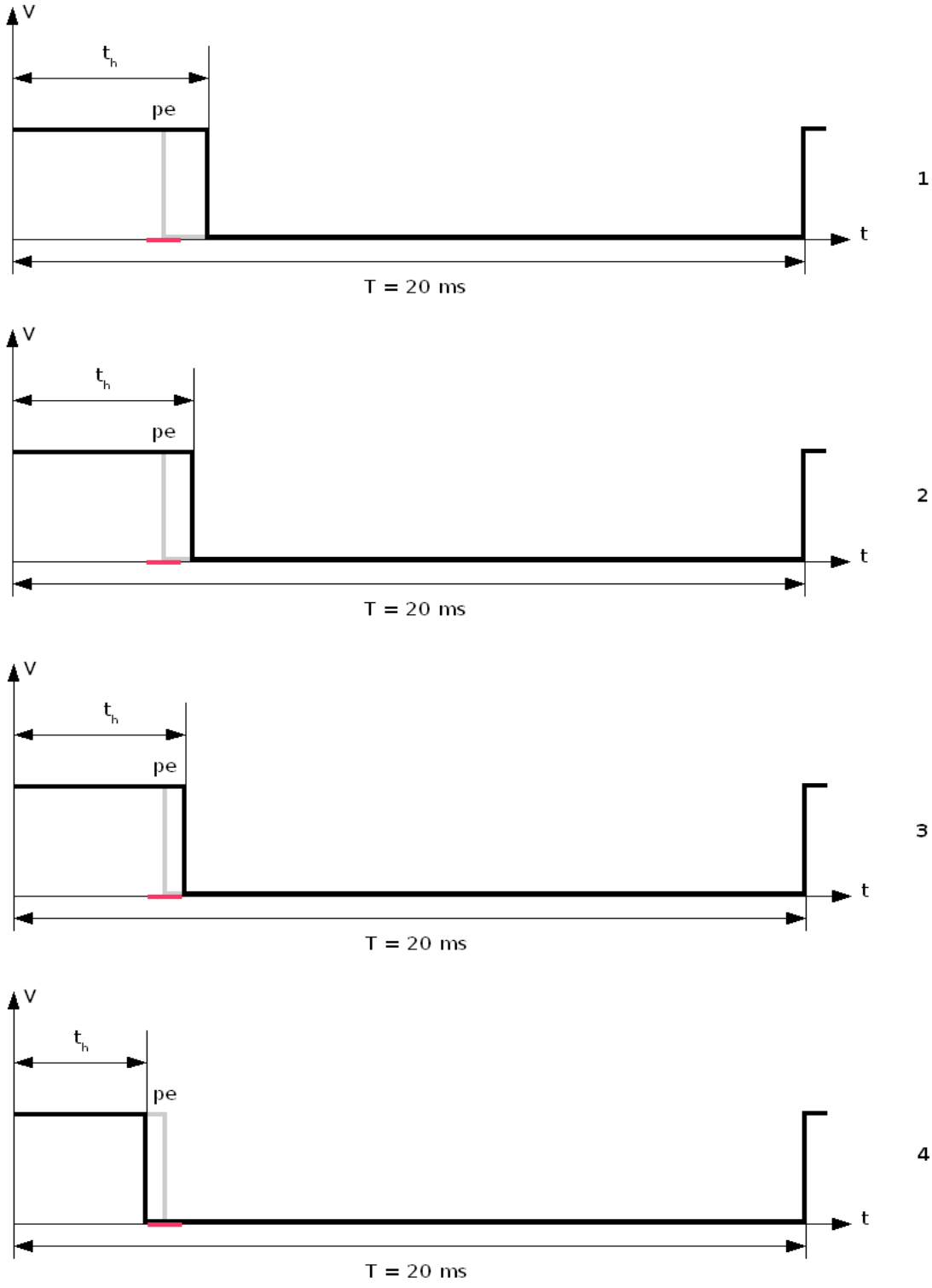


Figura 4-19: Procediment per la detecció del valor *pe*. La representació **1** parteix amb un $t_h > pe$. En **2**, ens aproximem per la dreta a *pe* (decrementem). En **3** acotem el *rang de possibles valors de pe* per la dreta. Finalment, en **4** acotem per l'esquerra

Després d'actuar tal i com esmentem, n'extrèiem els següents valors:

Possibles valors de pe, **roda esquerra:**
1478 ms
1479 ms
1480 ms
1481 ms
1482 ms (pe_l)
1483 ms
1484 ms
1485 ms

Possibles valors de pe, **roda dreta:**
1470 ms
1471 ms
1472 ms
1473 ms
1474 ms (pe_r)
1475 ms
1476 ms
1477 ms
1478 ms

4.3.2.2. Experimentació amb els servomotors: velocitat

Un cop sabem com operen els servomotors i sobre quins paràmetres s'ajusten, és l'hora de determinar tant la velocitat com la sincronització entre rodes.

Dues rodes amb velocitat de gir diferent provoca una desviació en la trajectòria esperada. Un desplaçament en sentit de l'eix *y* és molt més eficient (una acceleració menor) respecte a un desplaçament fora de la trajectòria d'*y*.

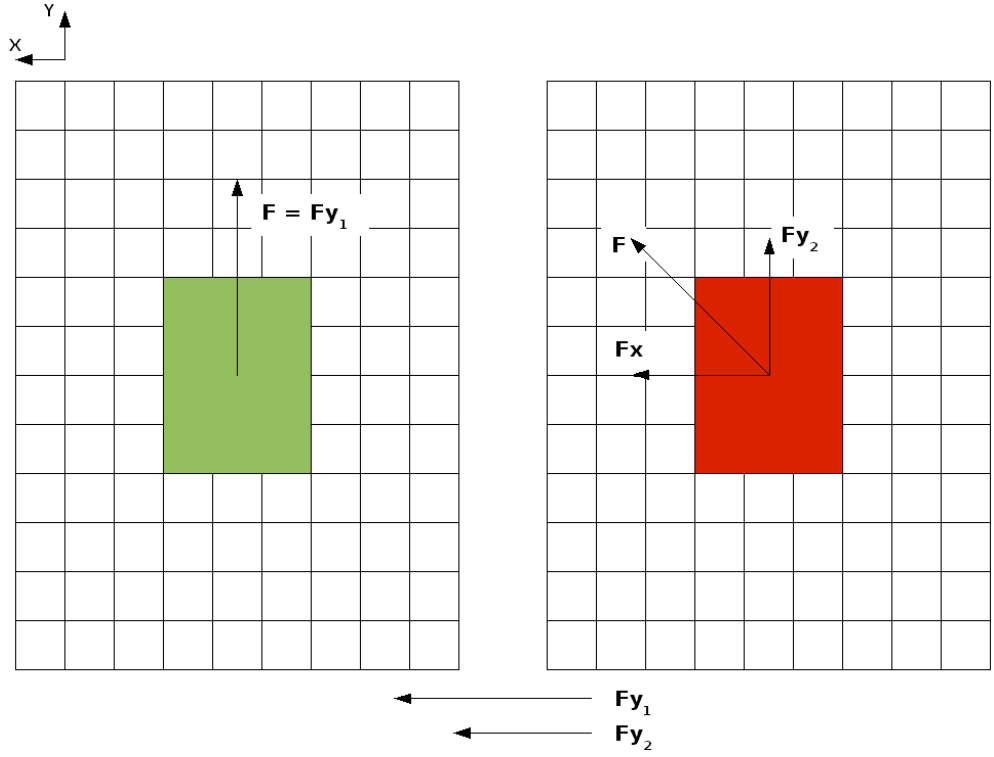


Figura 4-20: Demostració que l'aplicació d'una mateixa força F paral·lela a y és millor que en qualsevol altre direcció

Per tant, és important comprovar que donada una modulació equivalent per ambdues rodes, es segueix un trajectòria paral·lela a *y*.

La disposició dels motors, un muntat a la inversa de l'altre, fa que per un mateix sentit de gir, en un servomotor hàgim d'incrementar un *offset k* respecte *pe* en la senyal de modulació, mentre que per l'altra, sigui necessari restar *k* respecte *pe* per conseguir un mateix sentit de gir i teòricament, una mateixa velocitat de rotació.

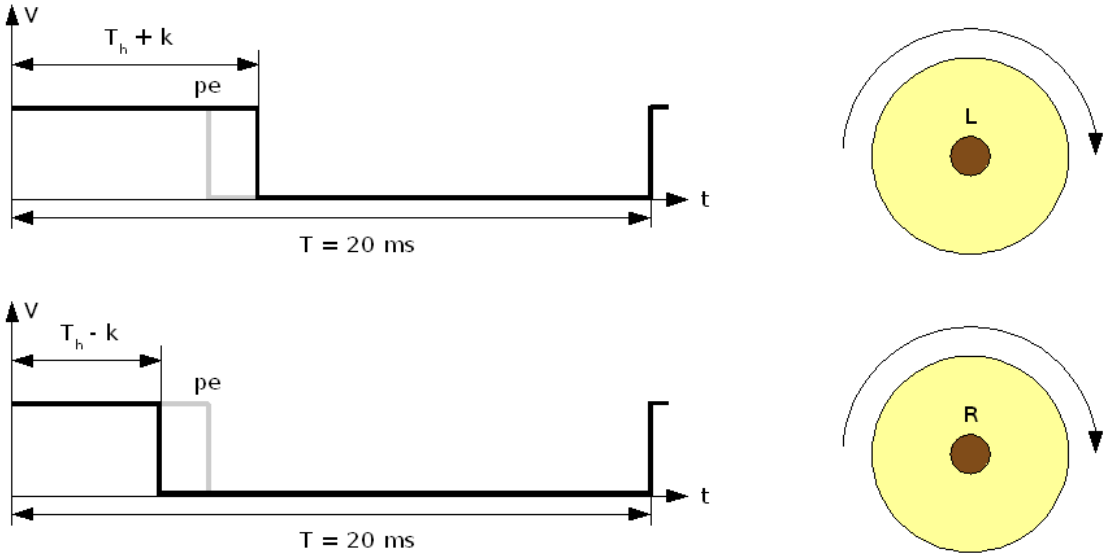


Figura 4-21: Exemple de rotació de les rodes en un mateix sentit. Quan incrementem k en el pols de l'esquerra, hem de restar k al pols de la dreta

Així doncs, el mode de procedir ha estat el següent: Partim d'un punt inicial conegut —punt de partida— i en un estat de repòs —**PWM**(pel, T) i **PWM**(per, T)—. Passat un temps prudencial, fem girar les rodes en un mateix sentit i velocitat —**PWM**(pe_l + inc, T), **PWM**(pe_r – inc, T)— durant un temps conegut ,suposem 5 s, per després aturar la rotació —**PWM**(pe_l, T), **PWM**(pe_r, T)—. Al llarg d'aquests cinc segons, la plataforma on estarà muntat el pèndul s'haurà desplaçat tot un tram. Posteriorment, es produirà una espera prudencial que durarà just el temps necessari, 15 segons, per anotar la distància desplaçada des del punt de partida. Seguidament es tornarà a posicionar la plataforma en dit punt, començant un nou cicle utilitzant `inc += cte` en cada nova iteració.

Actuant d'aquesta manera, sabem quina distància s'ha desplaçat durant un període de temps conegut (5 s); tenim aleshores la velocitat en funció del valor α*k, on $\forall\alpha$ = 0... número d'iteracions.

```
Algorisme test_vel()
{
      constant cte = 1;
      var inc = 0;

      while(inc > MAX)
      {
            PWM(pe_l + inc,T);
            PWM(pe_r - inc,T);
            inc := inc + cte;

            wait 5 s;

            PWM(pe_l, T);    // Aturem gir roda esq.
            PWM(pe_r, T);    // Aturem gir roda dreta

            wait 15 s;       // Anotem resultat i
                             // posicionem de nou
      }
}
```

Els resultats obtinguts es mostren en les següents gràfiques:

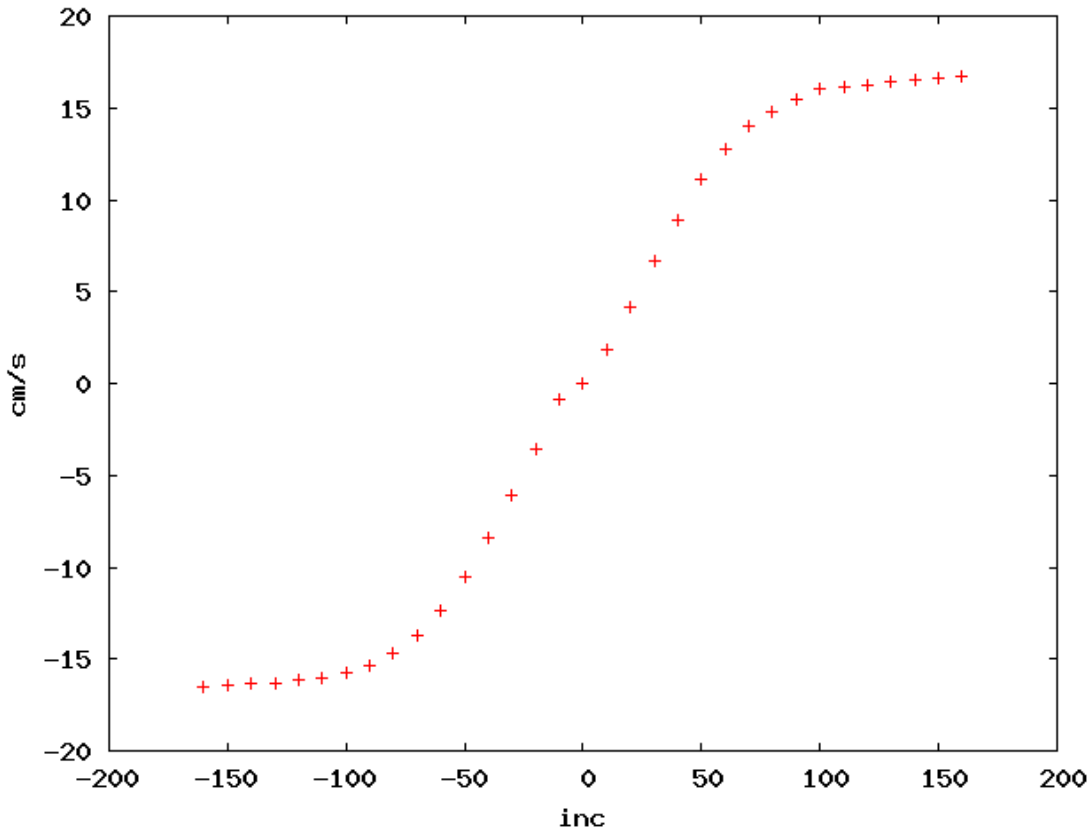


Figura 4-22: Els diferents valors obtinguts de pe – 160 a pe + 160. Els increments són de 10 en 10

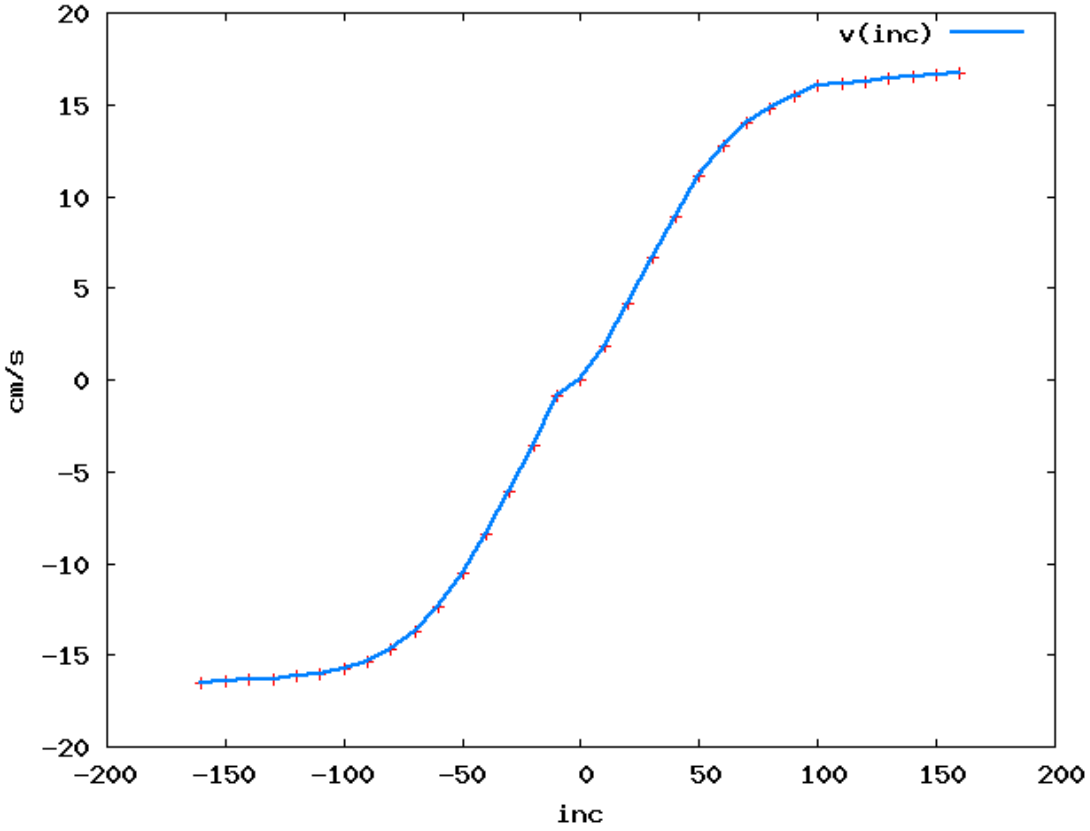


Figura 4-23: Linealització de la **Figura 4-22**

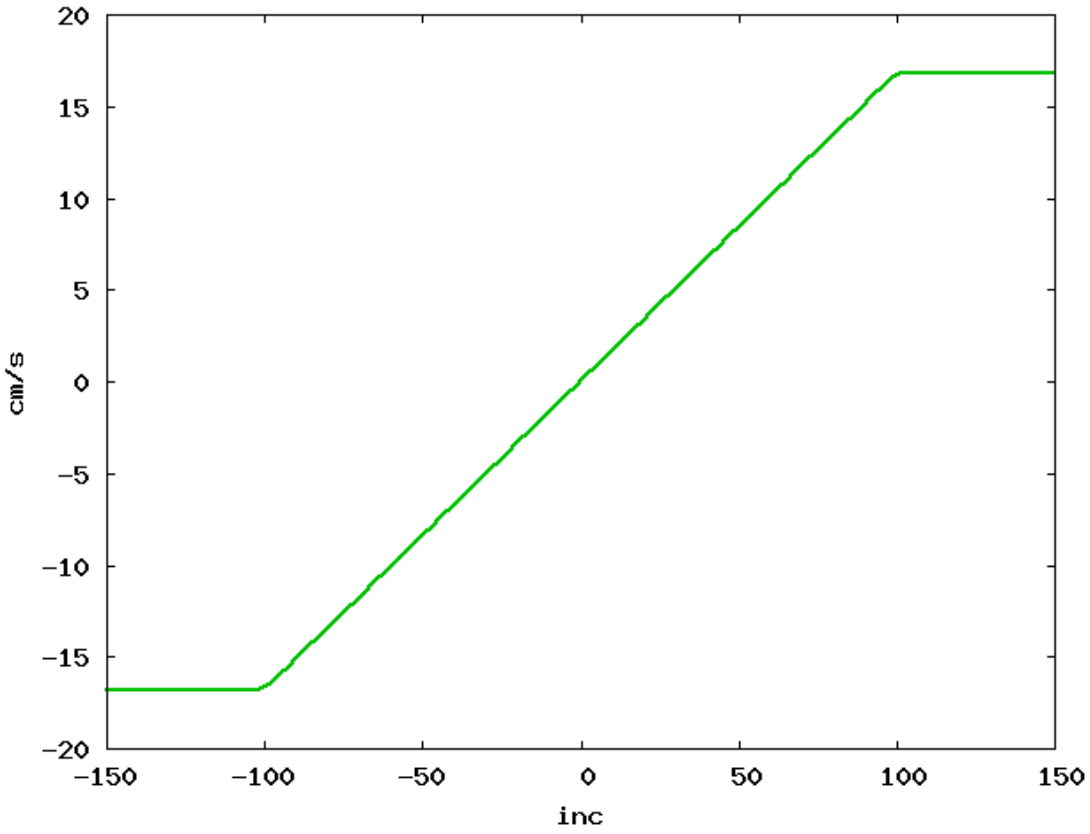


Figura 4-24: Idealització de la **Figura 4-23**

Els resultats de la prova evidencien dos fets:

- Partint de *pe*, un increment de l'amplada de pols provoca un increment proporcional en la velocitat de gir. S'estableix una relació lineal entre l'amplada de pols i la velocitat de gir.
- Existeixen uns valors llindar (100, -100). Més enllà d'aquests, la velocitat no correspon en proporció a l'increment. És més, superats aquests valors, la velocitat de gir és màxima o sigui, constant.

4.3.2.3. Apreciacions i consideracions

Una modulació PWM simple treballa de manera similar a un timer però utilitza dos registres a mode de *match registers*. Un primer registre conté el valor T i un segon t_h.

La senyal del PWM s'inicia amb un valor alt i decau quan el valor del $counter_{PWM}$ esdevé t_h. Després, el pols roman constant a zero fins vèncer el $counter_{PWM}$ a T, moment on aquest mateix es reseteja i el senyal del PWM comença de nou el pols —es repeteix la seqüència, la senyal esdevé 1—.

Per garantir el correcte funcionament, LPC2119 utilitza un esquema de *shadow registers* per protegir els registres de comparació. Què significa això? Vol dir que tots els canvis efectuats en els registres de comparació del mòdul PWM no tindran efecte —no s'actualitzaran per més escriptures que si facin— fins que el counter venci ($counter_{PWM}$ = T).

Quines conseqüències arrossega aquesta restricció per el nostre pèndul invertit? Bé, si el període T és 20 ms, significa que cada 20 ms podrem actualitzar el senyal del PWM. Si una caiguda té una durada pròxima als 300 ms, només podrem fer una modulació d'un tren de 15 polsos.

Considerant que cada 2.5 ms rebem una nova mostra de l'acceleròmetre, 7 de les 8 mostres enviades no modificaran l'amplada del pols (8 x 2.5 = 20 ms). Només la vuitena mostra, per ser la mostra que es rep cada múltiples de 20 ms, serà qui modularà.

Situant-nos en el pitjor escenari possible, si detectem una caiguda just quan hem actualitzat el mòdul PWM, disposarem només de 14 modulacions per actuar; dit d'una altra manera, començarem una modulació per la recuperació de la verticalitat del pèndul 20 ms més tard de la seva detecció.

```
Caiguda: 300 ms
Mostra: 2.5 ms
T = 20 ms

T / Mostra = 8 mostres per cada T
Caiguda / T = 15 modulacions

Actualitzacions:
   -  En el millor dels casos: 15 modulacions (300 ms)
   -  En el pitjor dels casos: 14 modulacions (280 ms)
```

4.3.3. Control PID

Tornant a les senyals proporcionades per l'acceleròmetre (**Figura 4-11, Figura 4-12, Figura 4-13 i Figura 4-14**), és notòria la necessitat d'un tractament posterior per accentuar els factors determinants d'una caiguda, especialment en el cas de l'eix *y* per remarcar el sentit de la caiguda.

Per a dur a terme un control de l'error —desviació de les mostres rebudes respecte una mostra de referència—, fem una implementació d'un PID —*Proporcional Integral Derivatiu*—.

Un control PID ens garanteix un seguiment de la desviació de l'error respecte el senyal rebut i una resposta clara davant d'un comportament anòmal[7].

En el nostre cas, l'error es mesura respecte una mostra de referència determinada, ja sigui per una mostra aleatòria escollida durant l'estat estàtic o bé, fent una mitjana de les mostres rebudes durant un cert interval de temps —durant tot aquest interval, el pèndul roman constant sobre la seva vertical, estàtic—.

En l'equació del PID intervenen tres parts:

[7] El control de l'error per part del PID significa que, en cas d'estar en un estat correcte, la senyal de sortida PID_{out} és zero (no hi ha error). En cas d'existència d'error, el valor absolut de la senyal PID_{out} ascendeix a mesura que l'error s'agreuja (les mesures de rectificació preses no són suficients)

- **Proporcional, *P*:** Valor proporcional a l'error actual (*e(t) = Mostra de ref. - Mostra actual*). L'error *e(t)* s'escala amb la constant *Kp*: $\boldsymbol{P_{out}} = K_p * e(t)$.

  Quan $K_p > 1$, l'error actual s'amplifica, proporcionant més guany del que hauria d'aporta. Contràriament, quan $K_p < 1$ l'error *e(t)* es fa menys sensitiu.

- **Integral, *I*:** La part integral manté una memòria dels errors anteriors fins l'actual —integració de l'error—. Dit valor s'escala a una constant K_i, $\boldsymbol{I_{out}} = K_i * (e(t) + e(t-1) + ...) * \Delta t$).

  Jugant amb els paràmetres K_i s'observa que la part integral és la responsable de marcar la tendència de la funció de control. Amb un *Ki* > 1, un conjunt d'errors amb mateixa orientació influenciarà decisivament el sentit de la gràfica. Amb $K_i < 1$, la memòria d'errors passats no tindrà tant de pes sobre la tendència general del senyal de control.

- **Derivatiu, *D*:** La part derivativa contempla la velocitat del canvi entre l'error actual i l'anterior, $(e(t) - e(t-1)) / \Delta t$, multiplicat per la constant K_d, $\boldsymbol{D_{out}} = K_d * ((e(t) - e(t-1)) / \Delta t)$.

  La part derivativa és adequada quan el senyal d'entrada no té una oscil·lació massa acusada —absència de soroll— del contrari, amb una constant $K_d > 1$ estaríem accentuant gran part de les interferències.

L'equació final ve donada per $\boldsymbol{PID_{out} = P_{out} + I_{out} + D_{out}}$.

4.3.3.1. Realització del PID i resultats

Cada eix té un tractament PID per separat, amb paràmetres K_p, K_i i K_d propis i escaients per a la finalitat de cada un —*z* detectar caiguda prematura. *y* definir sentit de la caiguda, **Figura 4-25**—.

La resolució dels valors K_p, K_i i K_d s'ha fet a partir de l'observació de les gràfiques resultants del PID provant diferents valors per a cada constant.

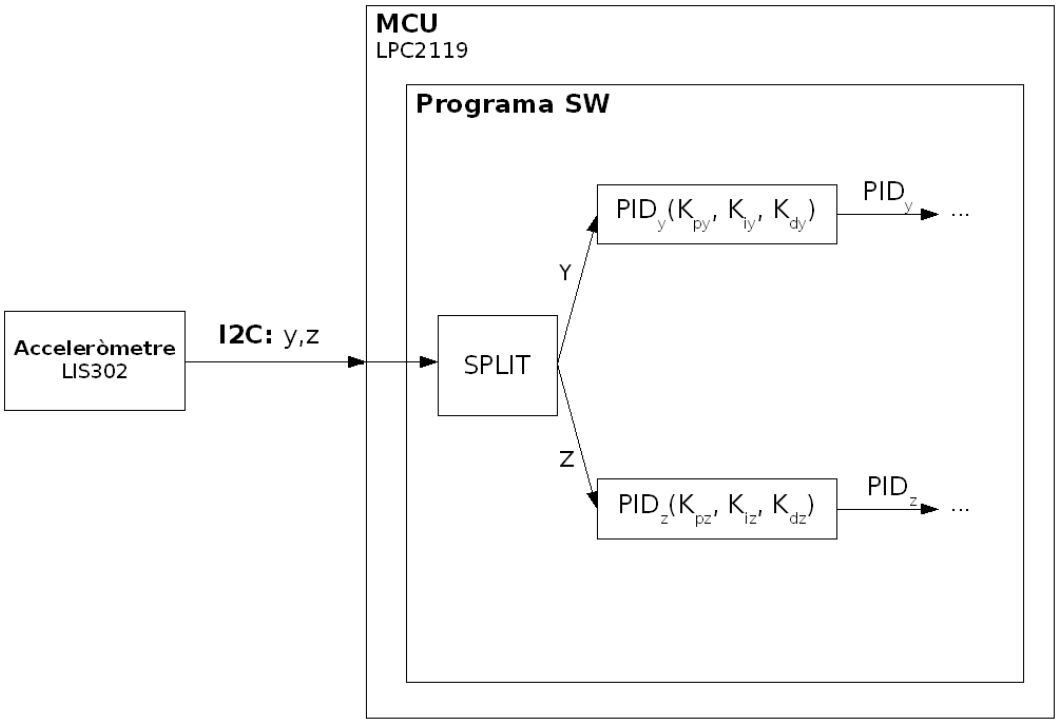


Figura 4-25: PID per a cada eix

```
Algorisme  PID()      //[Wiki02]
{
   previous_err = 0;
   integral = 0;
   start:
      err = setpoint - actual_position;
      integral = integral + (err*dt);
      derivative = (err - previous_err)/dt;
      output = (Kp*err) + (Ki*integral) +(Kd*derivative);
      previous_err = err;
      wait(dt);
      goto start;
}
```

Fent una codificació literal en C de l'algorisme PID [Wiki02] i utilitzant les mateixes dades capturades per representar la **Figura 4-10**, obtenim, amb diferent valors de *Setpoint*, la **Figura 4-26** i **Figura 4-27**.

Tot i trobar-se en un estat estàtic durant tota la seqüència i assegurar que el valor de referència adoptat per ambdós PIDs és el valor de la mitjana de totes les mostres rebudes, tant per l'eix *y* com per *z* —valor enter arrodonit—, els valors de control PID_y i PID_z no es mantenen estables (**Figura 4-26**).

Variant el *Setpoint* obtenim un nou comportament erràtic (**Figura 4-27**) que en cap cas és l'esperat —busquem l'estabilitat a 0 d'ambdós valors PID_y i PID_z—.

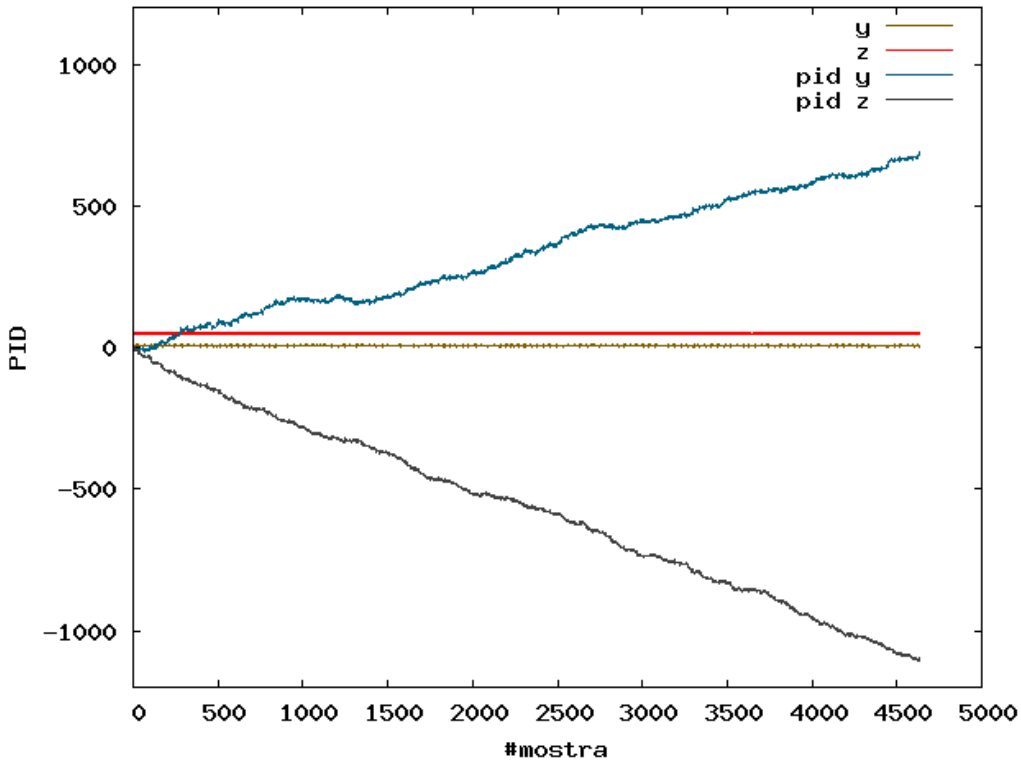


Figura 4-26: PID clàssic, **Setpoint:** (y = 6, z = 49)

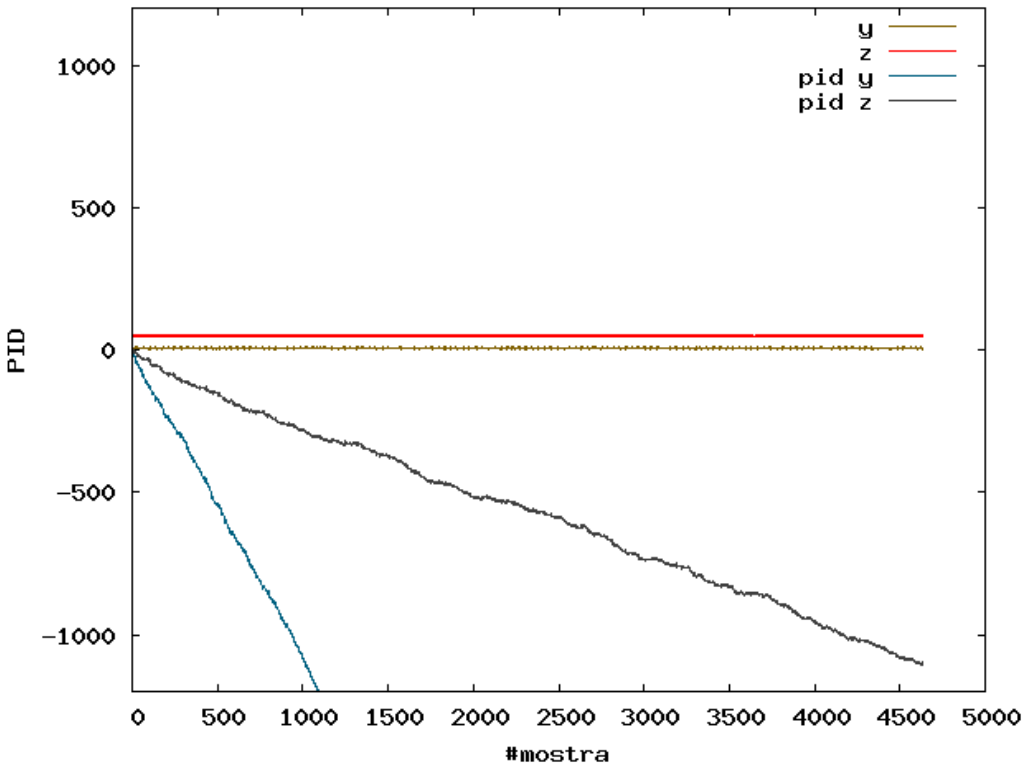


Figura 4-27: PID clàssic, **Setpoint:** (**y = 5**, z = 49)

Òbviament, un procediment PID reclama una acció rectificadora per restablir el valor de referència. Actuant només amb les dades, efectuem una observació passiva: estudiem la resposta del PID sense prendre'n mesures. Sembla sensat que, el fet de no actuar, deteriori el valor de control.

D'altra banda, això no justifica el comportament del PID en condicions estàtiques. Com hem mencionat, assegurem que el valor del *Setpoint* és la mitjana arrodonida a l'enter més proper de totes les mostres, tant per *y* com per *z*. És d'esperar doncs, que assumint el grau d'error d'una mostra —tolerància ±7—, una desviació soferta per un error negatiu quedi posteriorment equilibrat per un error positiu.

El concepte erroni radica en la forma de tractar les dades. Un procés PID treballa amb magnituds continues —decimals— però les nostres dades són discretes, evitant en la mesura del possible treballar amb operacions de punt flotant. Si recordem la mitjana dels valors de *y* i *z* de la **Figura 4-10** (pèndul en la seva vertical):

```
Mitjana de l'eix Y:          5.880
Mitjana de l'eix Z:          49.228
```

comprovem que no són valors exactes. La perdua dels valors decimals indueix a error el comportament de PID, degradant el seu valor de sortida de forma progressiva.

El problema incideix especialment en la part d'integració, i en menor mesura, en les altres dues parts. En el cas de la **Figura 4-26,** fixar el *Setpoint* a *(y = 6, z = 49)* suposa afegir un possible guany víric a l'eix *y*, pel simple fet de supravalorar el valor del component *y* del *Setpoint* (la mitjana de les mostres rebudes estarà per sota del valor *y* de referència, induint un error positiu fals). Contràriament, per l'eix *z* infravalorem el seu valor al arrodonir-lo a la baixa respecte la mitjana decimal, introduint un guany víric negatiu (error negatiu fals).

Tant en la part ***P*** com en la ***D*** esdevé un mal menor; una introducció d'un guany relatiu que desfigura sense gaire importància el valor de control PID_{out} sortint. Més crítica és la part ***I***. El procés d'integració utilitza l'error present i passat —memòria d'error— per marcar la tendència de la gràfica. Si es comet un error que introdueix un guany, aquesta porció afegida —o restada— es veu perpetuada al llarg de tota

la seqüència de control posterior. Després de la recepció de varies mostres, el component I_{out} arrossega amb ell tot una sèrie de guanys adquirits que desvirtuen completament els respectius valors de control PID_{out} (veure **Figura 4-26**). Pitjor encara quan $K_i > 1$, cas que ens interessa per ser decisiu en l'accentuació de la tendència de la gràfica.

Si observem la **Figura 4-27**, el *Setpoint* és *(y = 5, z= 6)*. El component *z* té el mateix valor d'abans i per tant, PID_z té comportament idèntic a l'anterior gràfica examinada; ara però, el component *y* de referència es troba per sota la mitjana decimal. Aquesta infravaloració provoca una ràpida acumulació de guanys negatius que precipiten PID_y, allunyant-lo de l'estabilitat que suposaria romandre en l'eix d'abscisses, amb més prestesa que la gràfica anterior (**Figura 4-26**), pel fet d'estar més lluny del seu valor en la mitjana decimal.

4.3.3.2. Modificació del PID. PID amb memòria relativa, PIDM

Per fer front al problema de la degradació del control del PID hem introduït una modificació en la corresponent part d'integració del PID, deixant intactes la part proporcional i derivativa —l'error que introdueixen és assimilable—.

L'efecte memòria que posseeix la part ***I*** fa persistents tots els errors al llarg de tota la seqüència de control. La modificació consisteix en establir un horitzó de memòria tenint presents només els últims *n* errors més recents. No evitem la introducció d'un guany víric però, sí limitem la seva persistència a *n* iteracions del procés de control PID (veure **Algorisme** `PID_M()` a continuació). Actuant d'aquesta manera, aconseguim acotar els valors de control dins uns màxims i mínims preestablerts.

Així doncs, renunciem a l'objectiu d'estabilitzar el valor PID a 0 en estàtic, per treballar dins d'un marge definit entre el llindar mínim i màxim. Aleshores, ens podem preguntar, ¿quin avantatge suposa l'aplicació d'aquest nou PID, podent treballar directament amb les mostres entregades per l'acceleròmetre? Certament, aquest nova implementació no elimina el molest efecte d'aplicar una tolerància sobre el senyal de control. No obstant això, el tractament amb PID fa predictible un caiguda molt abans de ser perceptible directament des de les mostres de l'acceleròmetre.

Seguidament, la nova implementació del PID junt amb les gràfiques resultants:

```
Algorisme  PID_M()
{
   previous_err = 0;
   integral = 0;
   cI[n] = { 0 }, i = 0;

   start:
      error = setpoint - actual_position;

      integral = integral + (err – cI[i])*dt;
      cI[i] = err;
      i = (i + 1) % n;

      derivative = (err - previous_err)/dt;
      previous_err = err;

      output = (Kp*err) + (Ki*integral) + (Kd*derivative);

      wait(dt);

      goto start;
}
```

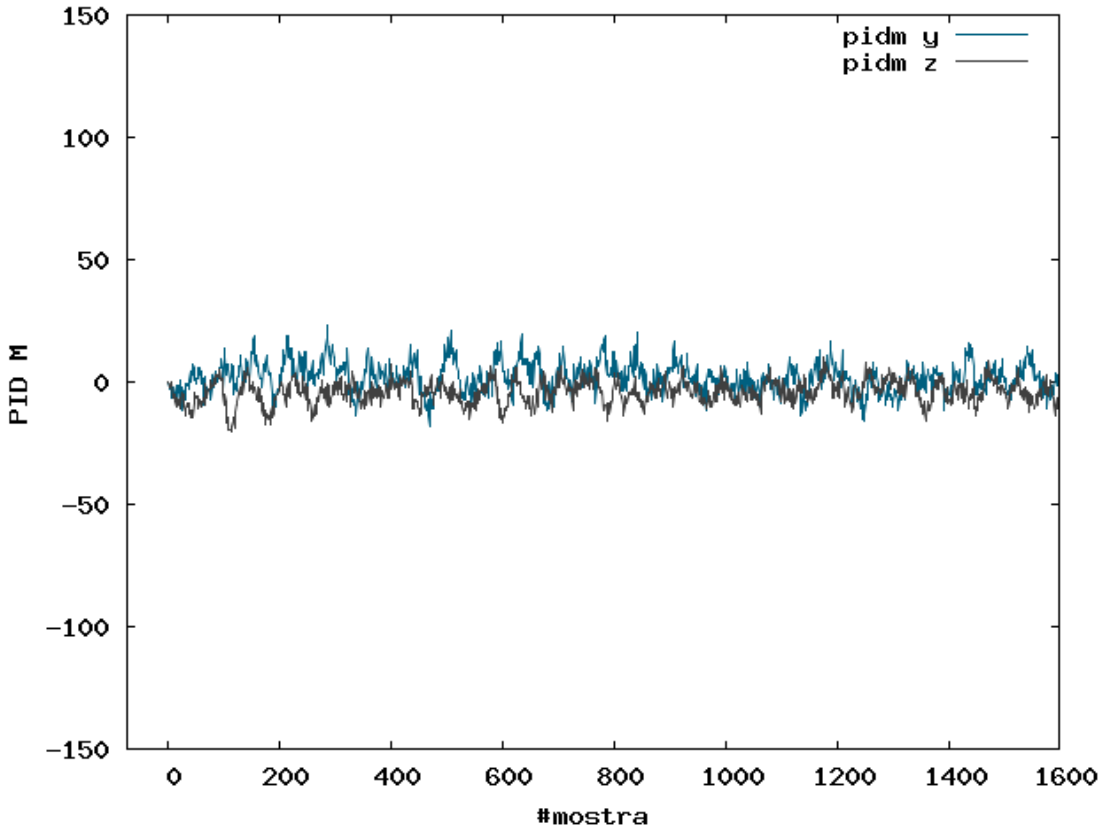


Figura 4-28 Només senyals dels PIDMs

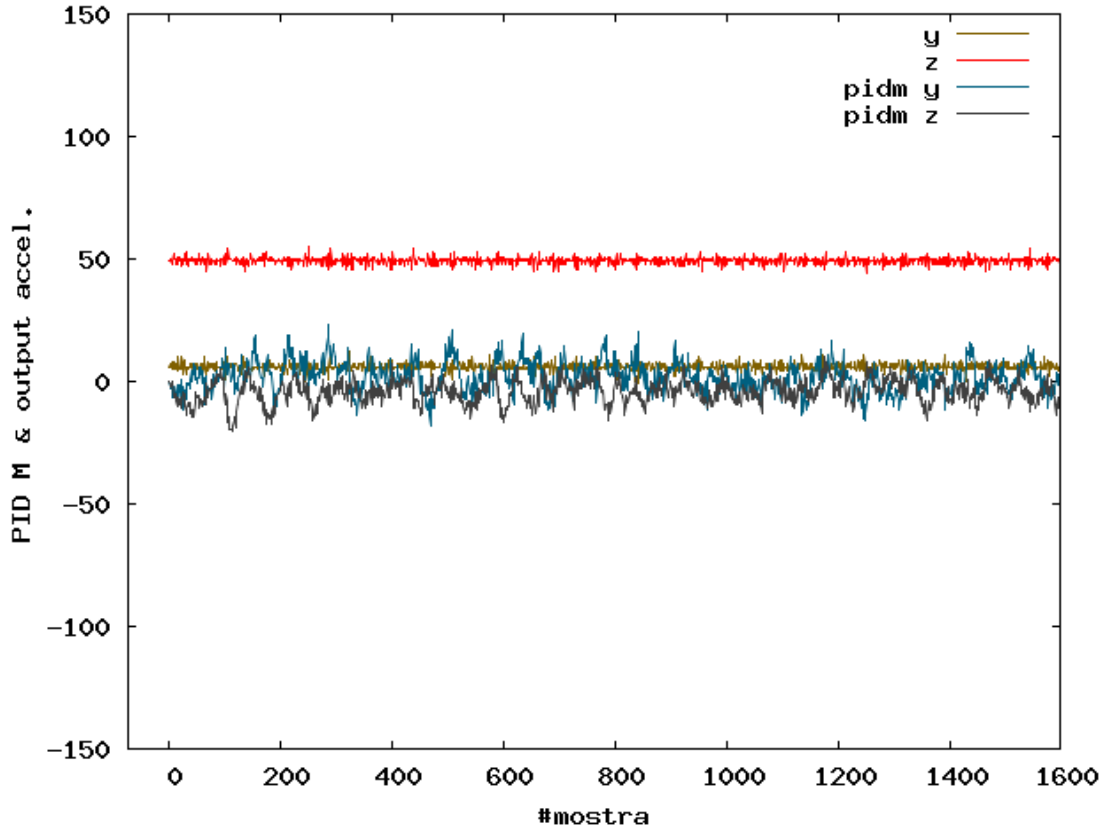


Figura 4-29 Senyals dels PIDMs i de l'acceleròmetre. Pèndul en estàtic

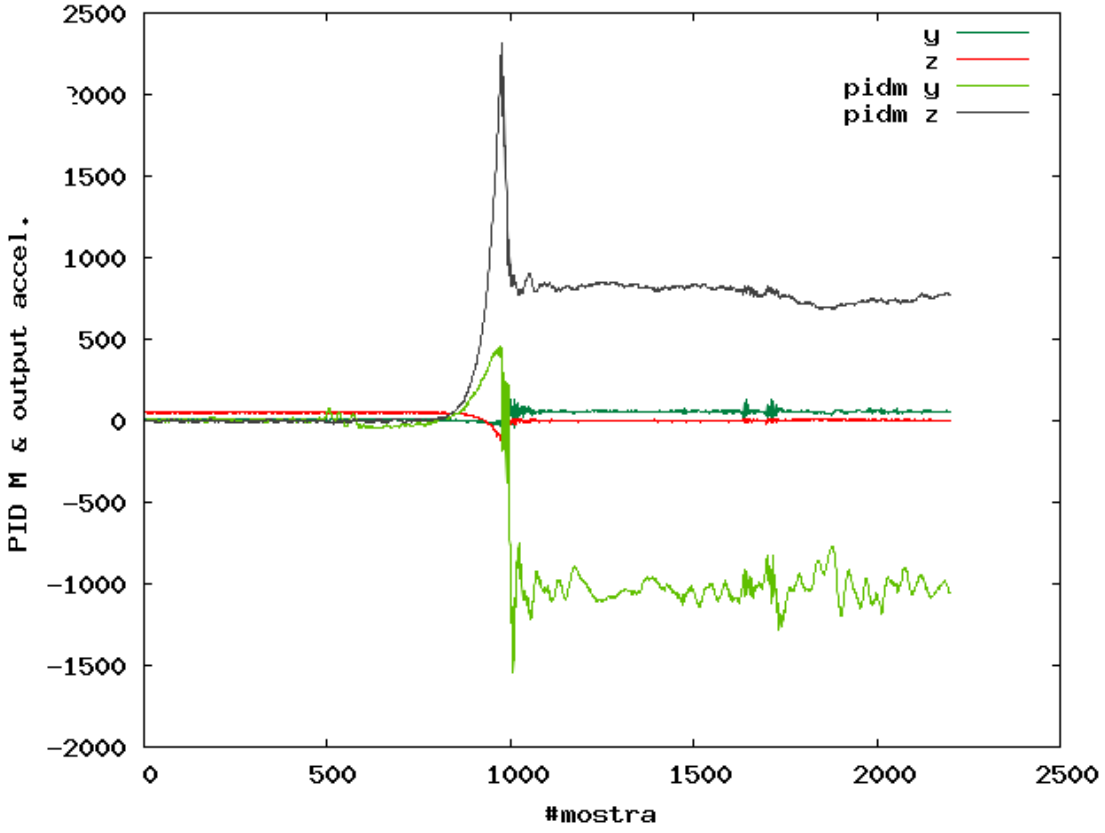


Figura 4-30 Senyals dels PIDMs i de l'acceleròmetre.Caiguda lliure a l'esquerra (**Figura 4-11**)

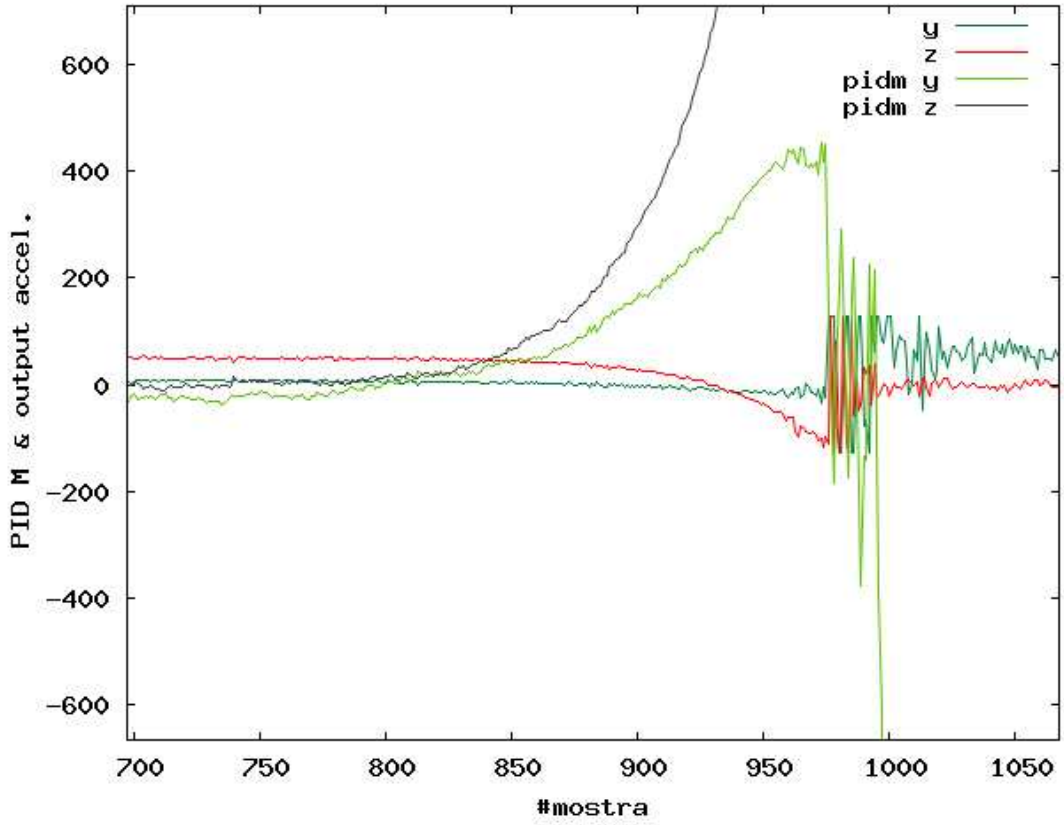


Figura 4-31 Ampliació moment de la caiguda (**Figura 4-12**)

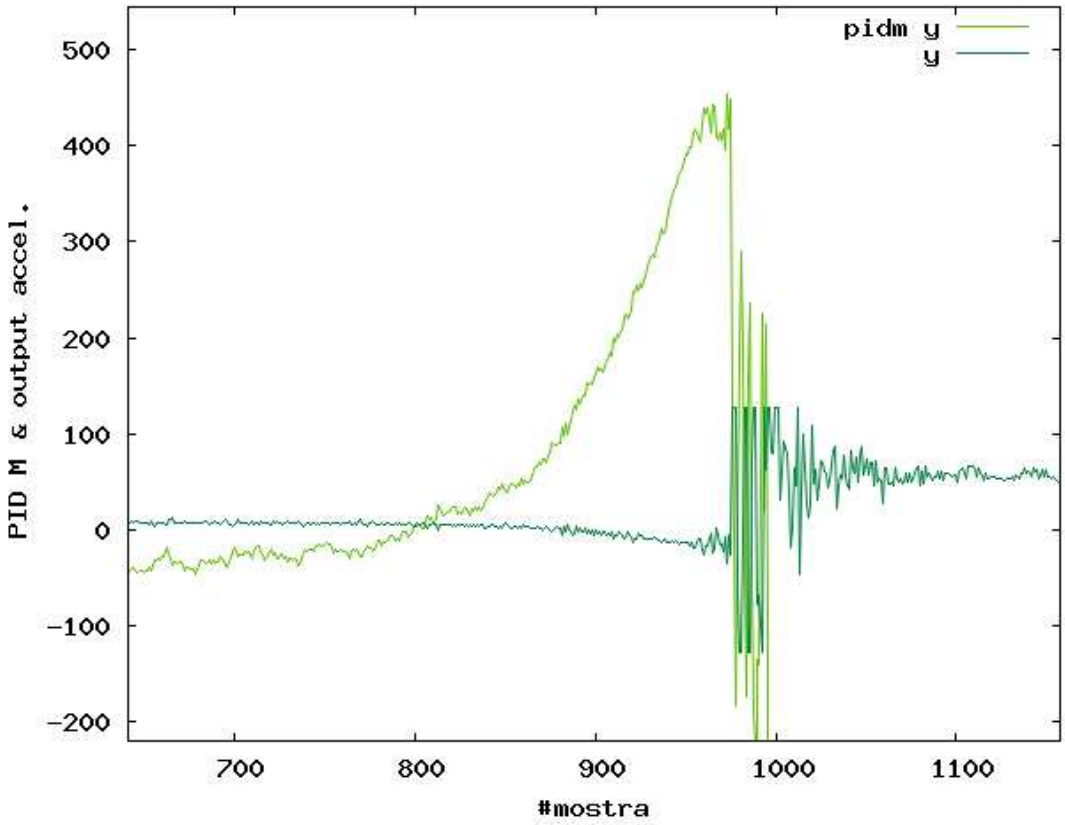


Figura 4-32 Senyals *y* i $PIDM_y$. Una quasi imperceptible daballada en *y* es converteix en una notoria pujada en $PIDM_y$

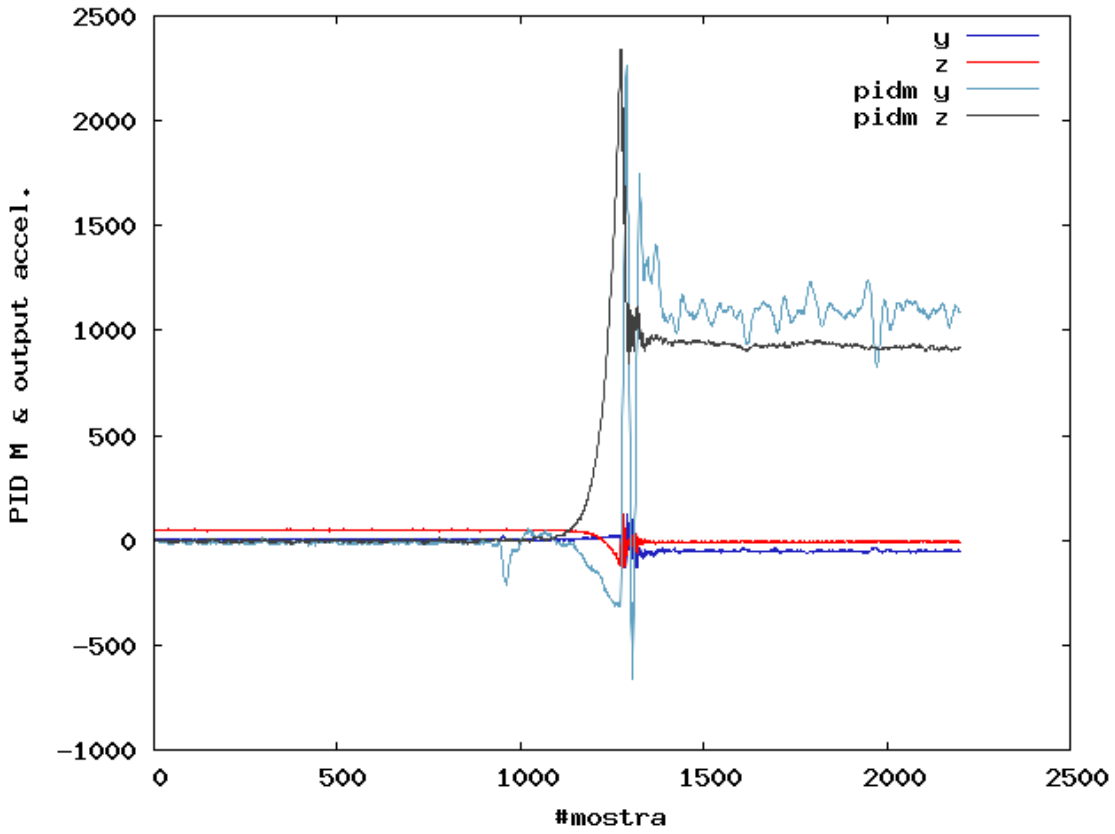


Figura 4-33 Senyals dels PIDMs i de l'acceleròmetre.Caiguda lliure a la dreta (**Figura 4-13**)

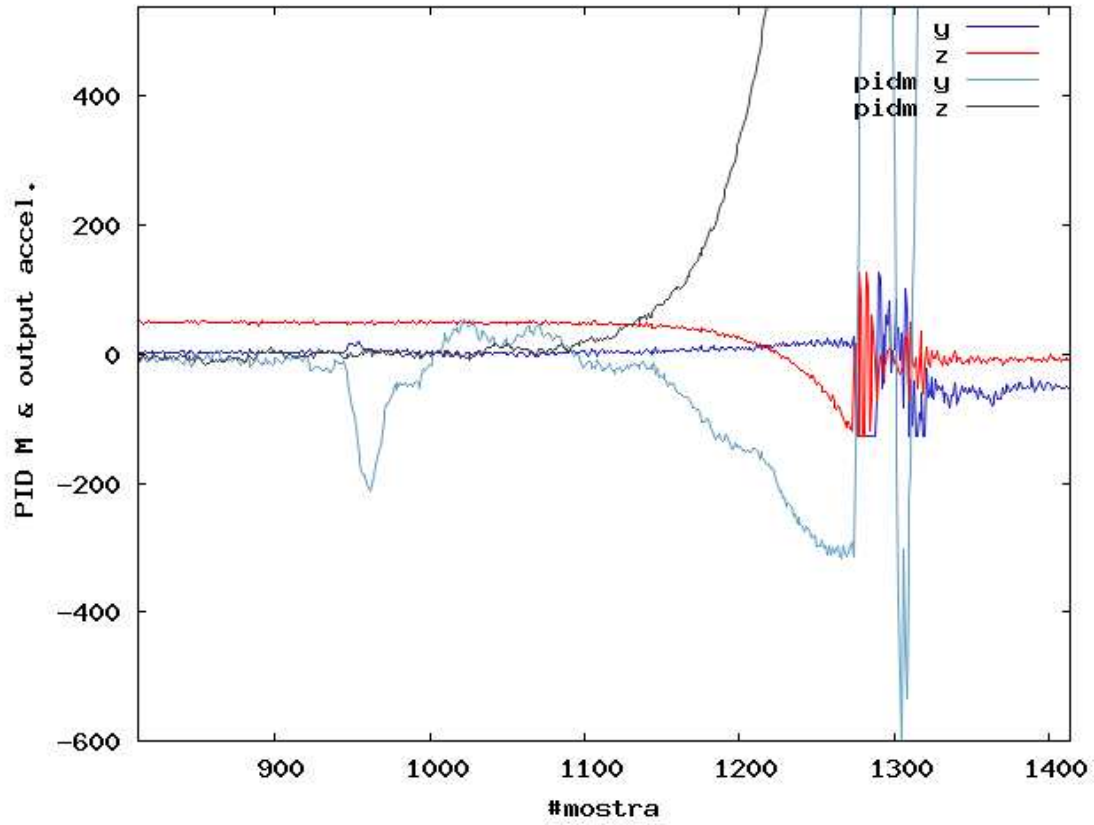


Figura 4-34 Ampliació moment de la caiguda (**Figura 4-14**)

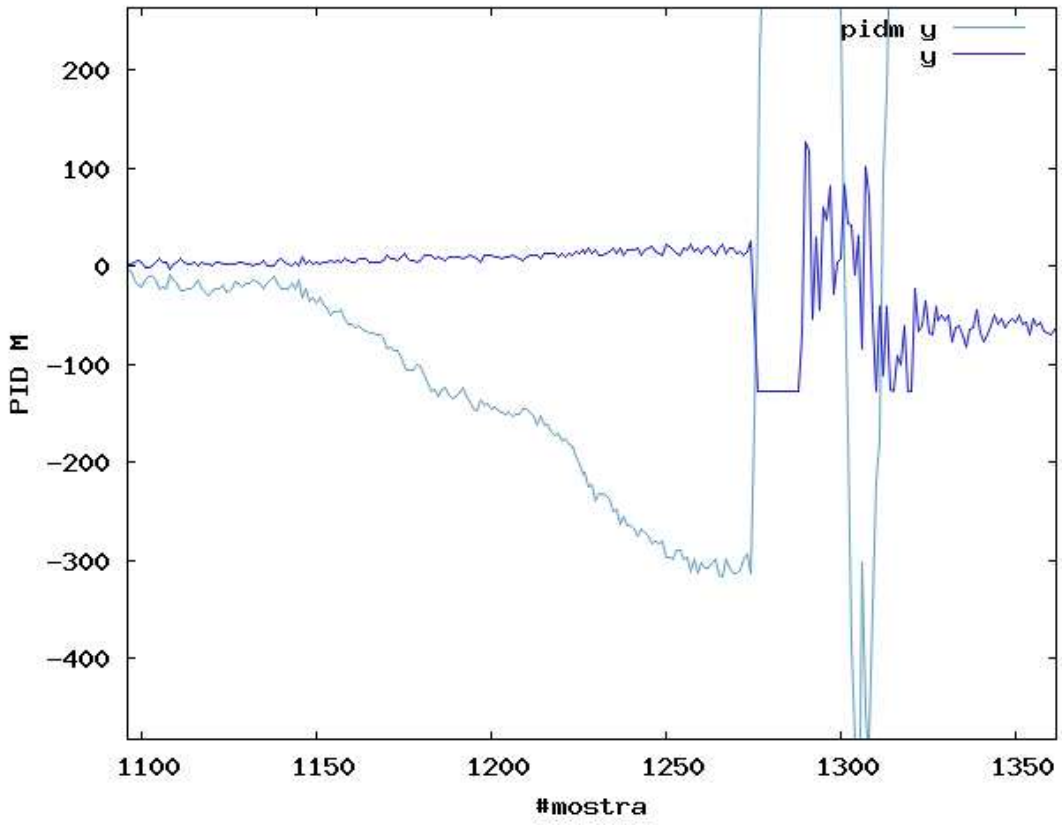


Figura 4-35 Senyals *y* i $PIDM_y$. Una petita pujada en *y* es converteix en una significant baixada en $PIDM_y$

```
Constants PIDM_Y:

      Kp: 0.5
      Ki = 1.25
      Kd = 0.25

Constants PIDM_Z:

      Kp: 0.5
      Ki = 1
      Kd = 0.25
```

Contemplem el comportament de les gràfiques anteriors. Observem que en totes les gràfiques, més visiblement en les figures **Figura 4-28** i **Figura 4-29**, els valors PIDM pateixen d'una oscil·lació més acusada respecte la ja inherent oscil·lació de les mostres de l'acceleròmetre. El fenomen, lluny de poder-se atenuar amb nous valors per a les constants, suposa a la pràctica la necessitat de jugar amb una tolerància major .

L'establiment d'un valor per la corresponent tolerància s'ha resolt a partir d'observacions en proves empíriques. Per les gràfiques anteriors, considerem una tolerància a partir de ±25 per valor de PIDM. Tot i així, aquest valor és relatiu,

sempre en funció tant de les constants del PIDM triades com del paràmetre *n* que marca l'horitzó de memòria de la part ***I***.

Per altra banda, veiem quina reacció pren davant una caiguda. Quan, degut a un inici de caiguda, l'eix de les *z* comença a decaure, el control $PIDM_z$ detecta ràpidament la diferència entre la mostra de referència i la mostra actual. És per aquest motiu que $PIDM_z$ creix positivament, perquè la component *z* de la nova mostra sempre serà més petita —diferencia positiva— en una situació de caiguda.

$PIDM_y$ és similar al cas $PIDM_z$ però sense buscar un resultat tant abrupte, cercant una resposta més lineal per, amb posterioritat, treballar sobre els valors $PIDM_y$ —conversió del valor en modulació pels servomotors—. La constant més significativa del procés és la constant d'integració. Amb $K_i > 1$ es pretén accentuar el sentit de la caiguda a partir d'una detecció incipient de la precipitació per par de *z*. Recordem que durant una davallada de la tija, l'eix *y* presenta una dèbil sensibilitat al canvi (**Figura 4-12**, **Figura 4-14** i **Figura 4-16**).

Per una caiguda a l'esquerra (**Figura 4-12)**, la diferència entre l'*y* de referència i l'*y* actual és positiva —l'eix *y* pateix una caiguda de valors— produint-se una crescuda en $PIDM_y$. A la inversa, per una caiguda a la dreta, l'error és negatiu, representant-se en una baixada de $PIDM_y$.

Per la figura **Figura 4-30**, la detecció de la caiguda es produeix a partir de la mostra 825 i dura fins la mostra 974. Això suposa un interval de 150 mostres. Considerant que cada nova mostra es rep en intervals de 2.5 ms, aleshores la caiguda de la figura té una durada de 375 ms.

En el cas de la **Figura 4-33**, la detecció esdevé durant la mostra 1109, allargant-se la caiguda fins la mostra 1273. L'interval definit durant la caiguda engloba 165 mostres, fet que suposa una durada de 412.5 ms.

Per a la caiguda a l'esquerra —375 ms, **Figura 4-36** i **Figura 4-37**— són possibles fins a 18 modulacions —17 en el pitjor dels casos—. En el cas de la caiguda a la dreta —412.5 ms, **Figura 4-38** i **Figura 4-39**— són possibles 20 modulacions —19 si la detecció es produeix durant l'actualització del PWM—.

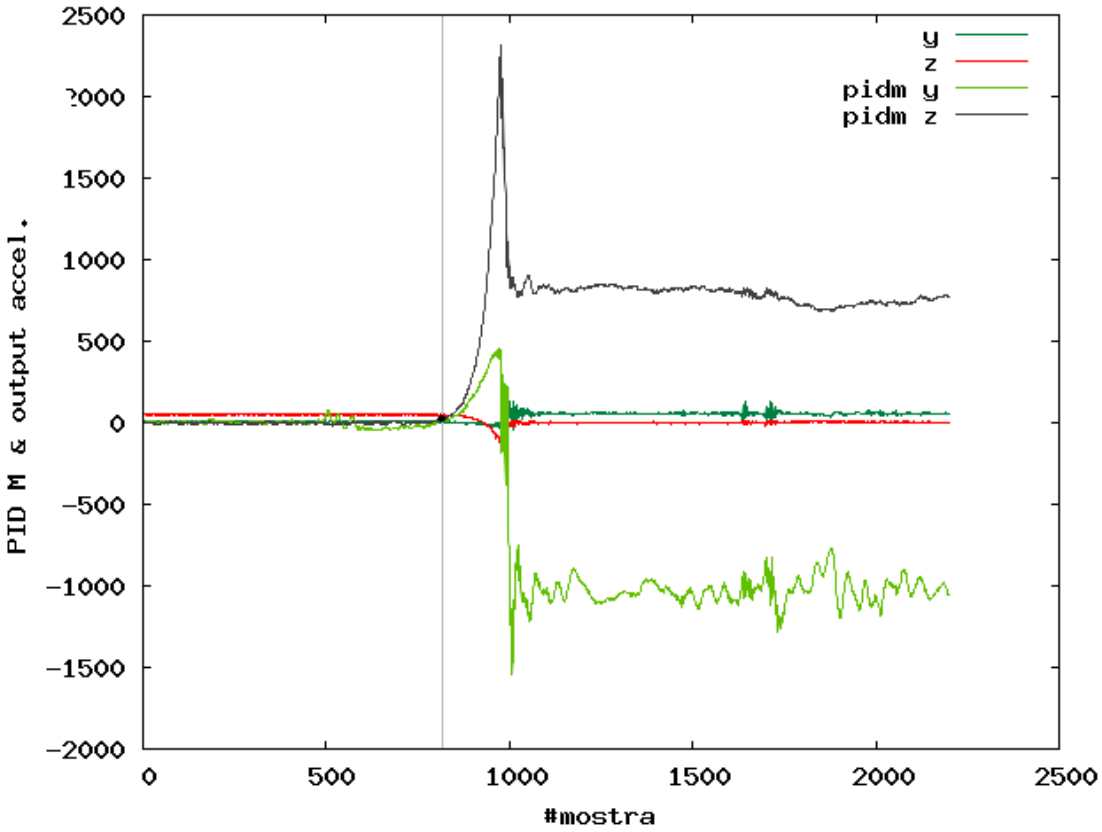


Figura 4-36 Senyalització de la detecció de la caiguda (mostra 825). La finalització de la caiguda es produeix 375 ms després de la detecció (mostra 974). Caiguda a l'esquerra.

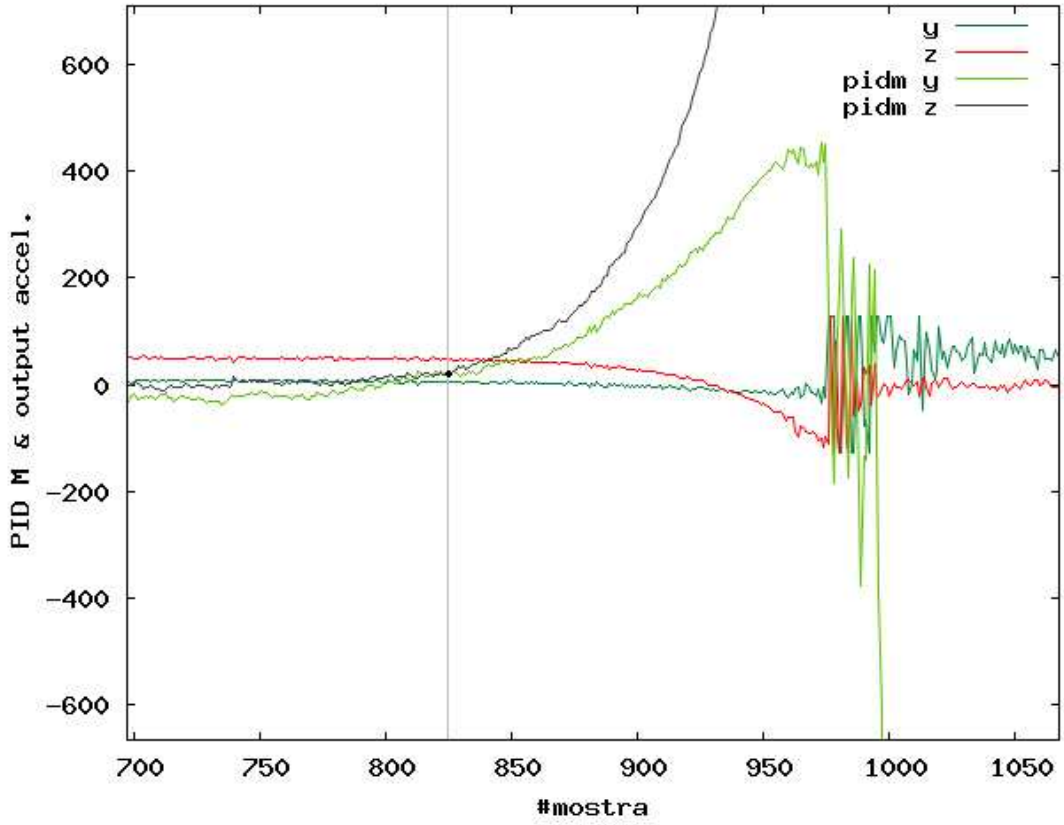


Figura 4-37 Ampliació punt de detecció, **Figura 4-36**

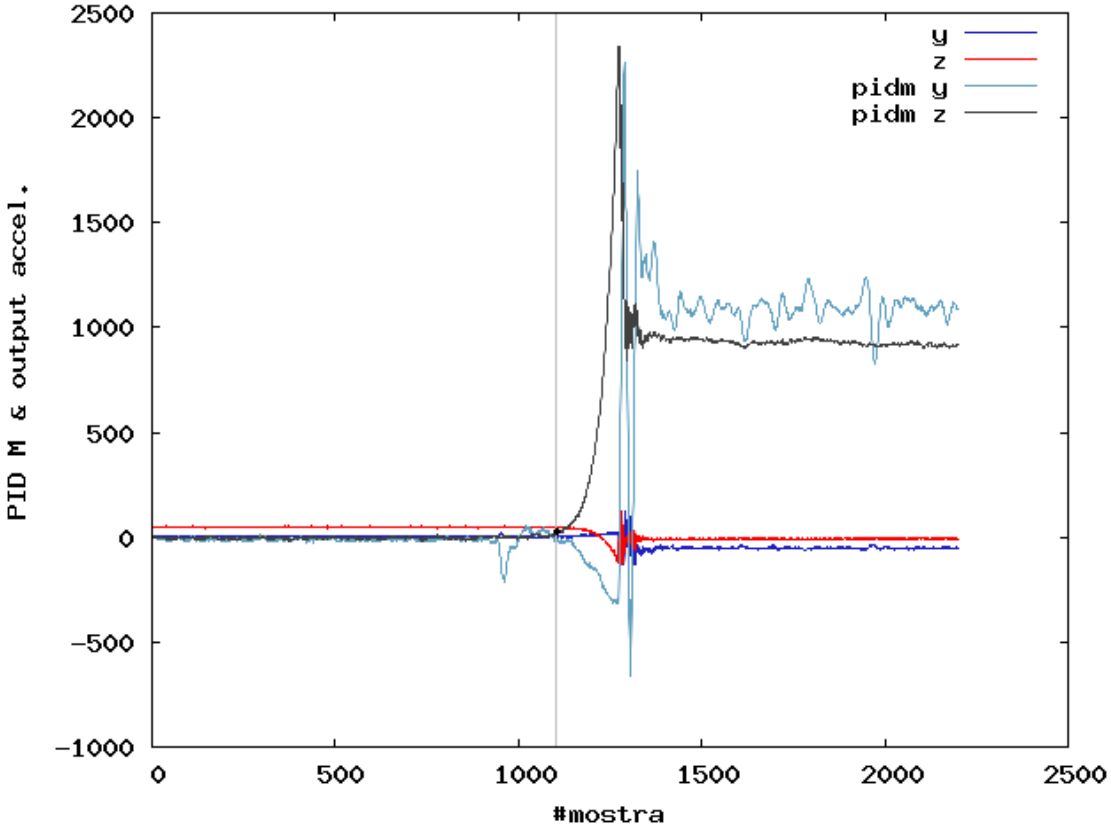


Figura 4-38 Senyalització de la detecció de la caiguda (mostra 1109). La finalització de la caiguda es produeix 412.5 ms després de la detecció (mostra 11273). Caiguda a la dreta.

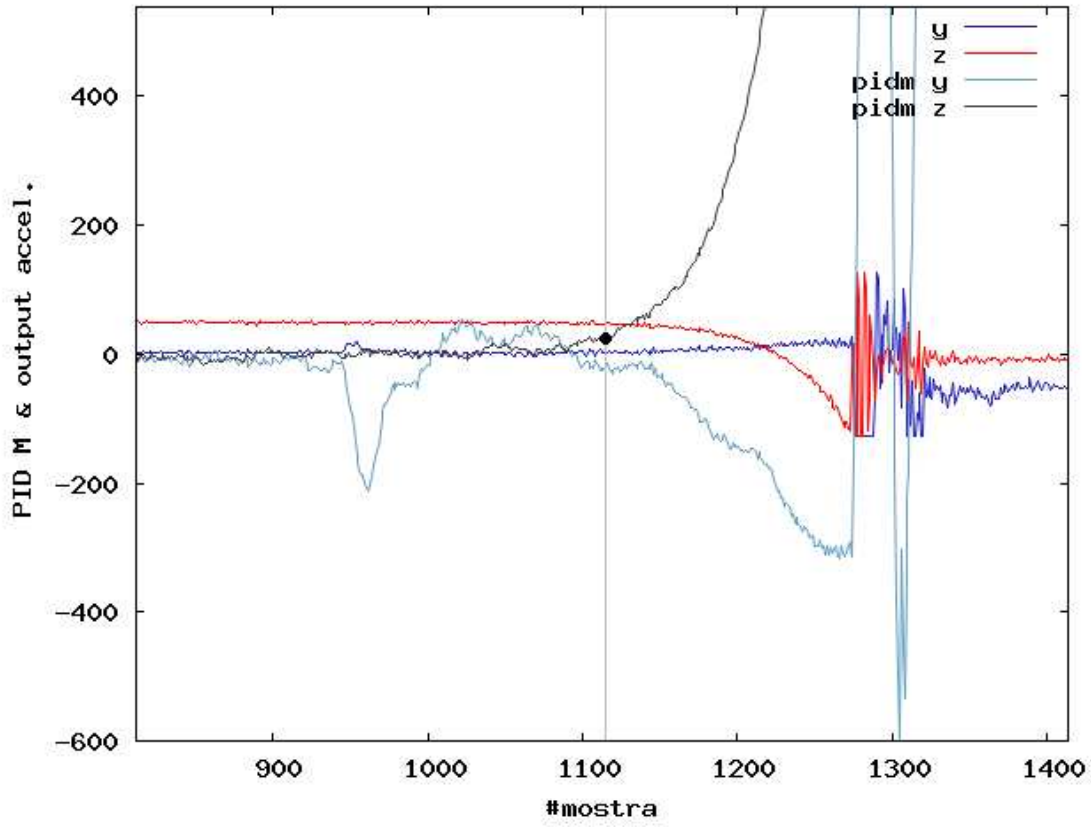


Figura 4-39 Ampliació punt de detecció, **Figura 4-38**

Existeixen altres aspectes i refinaments per aconseguir un control PID més efectiu. De fet, l'explicació teòrica del PID com de la seva modificació és una breu presentació de la nostra implementació per a la cerca d'un control fàcil i ràpid. No és pas ni pretén ser, un estudi exhaustiu sobre el control del senyal.

4.3.4. Modulació dels servomotors

Com ja sabem, el control del sentit i velocitat de gir es fa a través de la modulació per l'amplada de pols. Ara bé, la modulació dels servomotors no es pot fer directament amb els valors que ens proporciona el control PIDM. Ens cal una correspondència que transformi valors PIDM a valors vàlids per a la modulació.

A més a més, volem una solució que sigui transversal: la funció de modulació ha de ser independent dels valors PIDM. Per la forma d'experimentar que hem estat duent a terme al llarg de tot el projecte, no és admissible dissenyar una funció adhoc *PIDM – Funció modulació*. Primer, perquè la funció modulació triada pot no ser l'adequada i segon, relacionat amb el primer punt, canviar la funció de modulació per una de nova suposa reescriure de nou la codificació. Per cada funció de modulació provada, cal una nova funció escrita i testejada. Massa esforç i escàs temps.

Aleshores, en què ens basem per resoldre el control dels servomotors? Com bé hem esmentat en anterioritat, utilitzem $PIDM_z$ com a disparador d'una caiguda prematura, delegant a $PIDM_y$ la funció clau per a la resolució de la modulació.

$PIDM_y$ ens indica, des de l'inici de la detecció fins al final de la davallada, l'estat en que es troba la caiguda del pèndul. Per valors propers a zero, ens trobem en el principi de la caiguda; per valors pròxim al llindar —les observacions ratifiquen que $320 < |PIDM_y| < 400$— evidencia una estat de caiguda avançada. Afegit a més, el *pseudo-comportament* lineal de $PIDM_y$, la tasca de mapeig $PIDM_y \rightarrow$ *valor modulació* resulta trivial. Ara bé, la correspondència efectuada entre valor $PIDM_y$ i valor modulació no és directe. Utilitzem una solució tabular, emprant $PIDM_y$ com a índex d'una *taula de modulació* d'*N* valors. La taula, una ***look up table***, contindrà per cada posició, un valor a mode d'*offset* —respecte el *pe*— escaient a l'estat de la caiguda —el valor $PIDM_y$—.

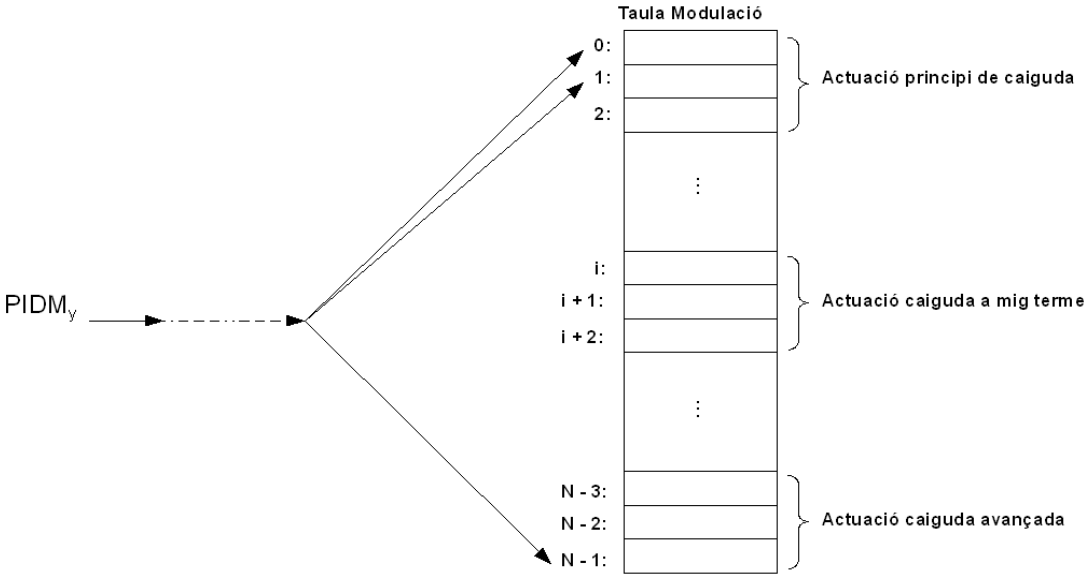


Figura 4-40 Utilització d'una *look up table*, sent PID$_y$ l'índex que apunta a l'offset que defineix una modulació resultant

```
const int PIDM_Y_MAX = 350;
const int PIDM_Z_FALL = 27;
const int N = 64;
const int modulacio[N] = { /* Valors funció modulació */};

Algorisme  Modulació(pidm_y, pidm_z) : int
{
   int offset = 0;
   int idx;

   if(pidm_z > PIDM_Z_FALL)
   {
      idx = round( abs(pid_y)*((N-1)/PIDM_Y_MAX) );
      idx = (idx > N-1) ? N-1 : idx;        //Precaució

      offset = modulacio[idx];
      offset = (pidm_y < 0) ? –offset : offset;
   }

   return offset;
}
```

L'algorisme implementat comprova inicialment el valor PIDM$_z$ abans de fixar un valor de modulació (en cas contrari, l'offset de *pe* és 0). Si PIDM$_z$ supera un valor llindar —valor escollit amb prou marge d'error com per identificar una caiguda sense dubte— s'empra PIDM$_y$ per trobar l'offset adequat per l'estat de la caiguda. PIDM$_y$ necessita d'un tractament previ abans de fer-lo servir com a índex de la taula.

El signe de $PIDM_y$ ens indica el sentit de la caiguda —$PIDM_y$ positiu, caiguda a l'esquerra. PID_y negatiu, caiguda a la dreta—. Obtingut el sentit, ens quedem amb el valor absolut $|PIDM_y|$. Aquest, comprés entre `0` i `PIDM_Y_MAX`, ens determina l'estat de la caiguda.

Per utilitzar-lo com a índex però, hem de readaptar-lo amb la següent operació:

```
 idx = round( |pidm_y| * ( N-1) / PIDM_Y_MAX )) );


on pidm_y és el valor de sortida del procés de
control PID de l'eix y.
PIDM_Y_MAX és el valor maxim que pot assolir
pid_y.
N és la mida de la look up table.
idx serà l'índex a la look up table.
```

Amb l'índex `idx` actualitzat, podem accedir al valor `offset` necessari per contratrestar la caiguda. Només falta aplicar sobre offset el signe en el sentit de la caiguda i retornar el valor.

Finalment, una mostra simplificada de l'algorisme general que es segueix:

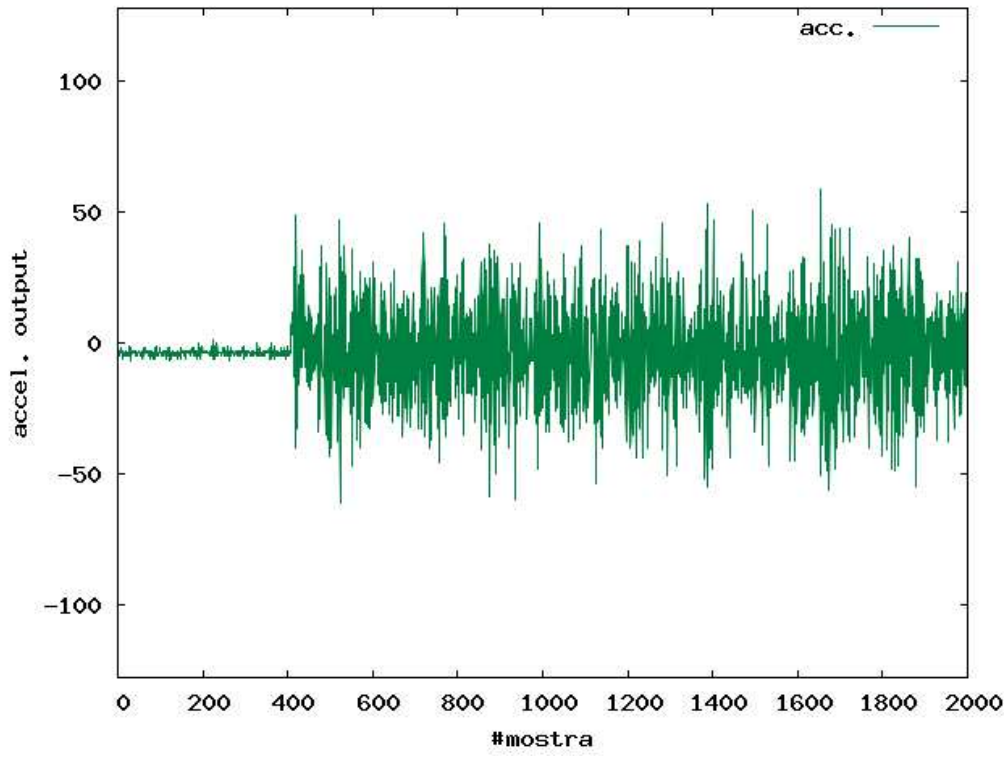
```
constant int LWHEEL, RWHEEL;
constant int pe_l, pe_r, T;

Algorisme  main(void)
{
   int offset;
   int y_sp, z_sp;
   int pidm_y, pidm_z;

   start_accelerometre();

   while(true)
   {
      y_sp = getY();             // Obtenim mostra y
      z_sp = getZ();             // Obtenim mostra z

      pidm_y = PIDM(Y, y_sp);    // Valor de control y
      pidm_z = PIDM(Z, y_sp);    // Valor de control z

      offset = Modulacio(pidm_y, pidm_z);
      PWM(LWHEEL, pe_l + offset, T);  // Mod. roda esquerra
      PWM(RWHEEL, pe_r – offset, T);  // Mod. roda dreta
   }
}
```

4.4. Funció resposta

Abans d'aplicar una resposta, ens hauríem de preguntar si amb els servomotors que disposem podem rectificar una caiguda del pèndul. Per demostrar-ho, hem fet un senzill experiment que ha ratificat la sospita de la incapacitat de rectificació dels servomotors. L'experiment consisteix en: just detectada una caiguda aplicar la màxima velocitat cap el sentit d'aquesta. Si el pèndul cau en sentit contrari, aleshores disposem d'uns actuadors amb suficient reacció com per revertir el sentit de la caiguda, i per tant, també per poder mantenir el pèndul estable. Això no ha estat així. Tot i la detecció prematura que es fa d'una caiguda, els servomotors no són capaços d'invertir el sentit de la davallada.

Afegit al problema anterior, hem efectuat un segon experiment que empitjora encara més la possibilitat d'èxit.

Totes les gràfiques de l'acceleròmetre mostrades fins ara han estat captades en repòs. Durant l'experiment, volíem comprovar quina relació *acceleració-amplada de pols* existeix quan es passa de repòs a una velocitat de gir constant. Indirectament, també era intenció comprovar la transmissió de soroll sobre les



dades de l'acceleròmetre en situació de moviment. Fixant un eix —eix *y*— en el sentit del desplaçament i aplicant un desplaçament a través dels servomotors, s'aconsegueixen els resultats representats en la **Figura 4-41**.

Es diferencien dues parts. La primera, estant la plataforma en repòs, l'acceleròmetre proporciona una sèrie de mostres dins un marge de tolerància acceptable. Un cop començada l'arrancada, les mostres rebudes són greument afectades per les vibracions transmeses pel moviment. A més, tot i amplificar-se el soroll, la gràfica no determina en quin sentit és produeix el desplaçament ni quina és l'acceleració del mòbil.

Per tant, ens trobem en una situació on no podem assajar amb cap funció resposta perquè ens manca tant d'un actuador capaç de rectificar la caiguda, com d'un sensor que no es vegi afectat, en gran mesura, per les pertorbacions paràsites inherents en un desplaçament.

A continuació s'introduiran un conjunt de funcions resposta on totes elles tenen un plantejament teòric, no havent estat provades.

4.4.1. Resposta lineal

Sembla evident que, la primera funció resposta a plantejar-se és aplicar un increment de la velocitat de gir proporcional a l'avenç de la caiguda, establint una relació entre grau de la tija respecte la vertical i velocitat de gir.

És evident que, una resposta proporcional a l'estat de la caiguda no és la millor resposta en la gran majoria de situacions, ja que moltes de les caigudes no poden ser neutralitzades.

Ara bé, un constant de proporcionalitat adequadament escollida podria modelar la resposta satisfactoria d'un petit reducte de caigudes.

4.4.2. Resposta no lineal

Una resposta no lineal (funció quadràtica, cúbica, exponencial...) és un intent per millorar el comportament de la resposta lineal. Per un petit canvi en l'estat de la caiguda correspon un gran increment de la velocitat de rectificació. Aquesta variació de velocitat —acceleració— induiria una força en sentit contrari sobre el pèndul, contrarestant la caiguda.

L'adopció d'aquesta resposta té però, dos inconvenients: primer, dona molt poc marge a la modulació —els valors es disparen dins d'un petit intèrval, anant de res a tot—. Segon, és possible que amb la intenció de contrarestar la caiguda vers un sentit, ens passem de la vertical i provoquem una caiguda en sentit oposat, produint-se un efecte oscil·latori en el pèndul que, de no ser controlat, l'oscil·lació podria ser crònica o agreujar-se.

4.4.3. Resposta múltinivell *look-up table*

Finalment, una resposta amb més d'una *look-up table* treballant a més d'un nivell. Una sola *look-up table* només pot modelar una funció. Un esquema jeràrquic múltinivell de *look-up tables* ens permetria solucionar el problema de l'oscil·lació crònica introduït en l'apartat anterior.

Per a la primer reacció —detecció de caiguda—, podríem implementar la *look-up table* anterior. Aconseguida la rectificació, just en el moment de passar per la vertical, tocaria un canvi de funció —canvi de taula— per atenuar el vaivé. Així doncs, per cada passada per la vertical, una nova *look-up table*, cada cop amb valors de velocitat de gir més petits per provocar una paulatina desacceleració. Actuant de tal manera, aconseguiríem un efecte oscil·latori que va a menys en cada anada i vinguda.

Un valor correcte estimat seria de l'ordre de dues o tres *look-up tables*, utilitzant una relació de proporcionalitat per acabar d'equilibrar —primera proposta—.

4.5. Resultats

Malgrat que l'acceleròmetre, en estable, proporciona una resposta positiva davant d'una caiguda; malgrat que el tractament de les mostres afavoreix una detecció a temps; malgrat que disposem de suficient potència de càlcul; malgrat tot, la materialització del pèndul invertit no és possible amb els materials dels que disposem, principalment, per la incapacitat manifesta de rectificar dels servomotors.

Observat el comportament del control PIDM directament des del dispositiu microcontrolador davant d'una caiguda (els valors d'acceleració *y* i *z* i valors $PIDM_y$ i $PIDM_z$ s'exterioritzen a través del port sèrie RS-232), la reacció és satisfactòria. És més, per uns instants sembla anular la caiguda del pèndul (veure **annex A**).

El problema radica en que la força per contrarestar la precipitació no és suficient. Dit d'una altra manera, l'acceleració aplicada com a resposta és capaç d'anular la caiguda però no de restablir el pèndul —recuperar la verticalitat—.

La solució apunta a uns actuadors, potser amb més potencia, però sí fonamentalment amb una relació de gir més elevada. És necessària una acceleració més alta. Per aconseguir-ho, o bé s'escurça el temps de resposta per arribar a la mateixa velocitat de gir, o bé s'assoleix una velocitat de gir més elevada dins el mateix marge de temps.

De totes totes, sense la satisfacció d'una acceleració adequada, sembla impossible recuperar la verticalitat perduda.

No obstant això, suposant de disposar d'uns actuadors que ens garanteixen els requisits demanats, toparíem amb l'acceleròmetre. Té dues problemàtiques: la relativa baixa taxa de mostres i l'estat d'inestabilitat en que entra quan es mou la plataforma —absoluta distorsió dels valors PIDM—.

El primer factor és pot esquivar amb més o menys encert utilitzant el control PIDM però per el segon, sembla no haver-hi remei. Fem notar que l'eix *y* sembla més sensible al soroll que l'eix *z,* presentant-se el pitjor dels casos donat que fem servir les dades d'*y* per aplicar la modulació.

La solució passa per un nou dispositiu immune a tal efecte o una compaginació mútua entre l'acceleròmetre i el nou dispositiu.

Finalment, també és un problema de base. L'estudi del pèndul invertit comporta tot uns coneixements que nosaltres hem intentat esquivar. De fet, la present memòria és un estudi del comportament del pèndul invertit en base a les observacions però, no és un estudi de la deguda resposta que cal prendre.

Sabem que el pèndul cau i podem determinar en quin estadi de la caiguda es troba, però, precisament per la intenció d'abstreure'ns del substrat matemàtic relacionat amb la caiguda, la resposta subministrada no rectifica correctament el comportament de deriva.

En tot cas, aquesta última part pot prendre el relleu algú més competent, disposat a centrar-se exclusivament en la resposta del pèndul invertit.

5. Conclusions generals

Després de nou mesos de projecte, és difícil resumir en unes línies tot aquest llarg període, principalment perquè aquest camí ha estat marcat per moments alts i baixos.

En l'àmbit personal, considero que l'evolució del projecte ha estat positiva: partint des de l'absoluta ignorància vers la plataforma, puc confirmar tenir-ne coneixença i saber-ne explotar les seves funcionalitats.

Durant els primers mesos, va esdevenir un procés d'aprenentatge fluid; un progrés lineal on cada dia s'avançava un pas més en el coneixement i familiariatzació tant del microcontrolador com de l'entorn de desenvolupament.

A mesura que es va anar descobrint les funcionalitats, contrastades en petits exemples a mode de prova, la proposta inicial del projecte va derivar cap a la creació d'un pèndul invertit. Sense percebre que potser era un repte massa ambiciós, el plantejament del pèndul invertit va suposar tot un canvi en el ritme del projecte. Ara calia barallar-se amb software —terreny fèrtil per un informàtic— però també amb tot aquells aspectes aliens a la programació. Sovint, durant el desenvolupament del pèndul invertit, el progrés s'aturava per falta de material, per errors de connexió, per dificultat en la mateixa construcció del pèndul... No obstant això, un cop construïda tota la infraestructura —no del tot finalitzada fins ben acabada aquesta memòria—, es va poder passar a un terreny purament experimental on calia ratificar les premisses plantejades en paper.

De nou, en aquesta nova etapa es va fer un gran salt, no exempte però de les seves dificultats. Gràcies a la reproducció del comportament del pèndul, es van extreure moltes de les conclusions exposades en la present memòria.

Independentment del resultat del pèndul invertit —impossibilitat de materialitzar-lo—, s'ha treballat amb tot un conjunt d'eines i instruments que han enriquit aquesta memòria. L'acceleròmetre o la tècnica de modulació per amplada de pols, no eren pròpiament temes que figuressin d'inici però, han tingut cabuda.

Així doncs, més enllà de presentar una descripció formal del dispositiu, s'ha pretès demostrar les seves virtuts en el camp pràctic. L'èxit és pot considerar relatiu però l'esforç ha valgut ben prou la pena.

Annex A: PIDM davant d'una reacció

Després de comprovar la impossibilitat dels servomotors per redreçar el pèndul, pot semblar interessant, si més no anecdòtic, observar com es comporta el control PID M davant l'intent d'aturar una caiguda.

Per altra banda, entre mostra i mostra de l'acceleròmetre —2.5 ms—, tenim un espai de temps on el core ARM roman en estat d'inactivitat (`IDLE`) gran part del seu temps. Simplificant l'algorisme general vist en el punt **4.3.4**, podríem resumir-lo en aquest cicle:

```
while(1)
{
      readAccel();    //Obtenció mostra Y i Z
      PIDM();         //Control PIDM per Y i Z
      modulacioMotors();

      IDLE;      // Surt IDLE per timer interrupt
}
```

La durada estimada per cada mètode és:

- **`readAccel():`** 0.17994 ms
- **`PIDM():`** 0.01524 ms
- **`ModulacioMotors():`** 0.00164 ms

Comprovem que la despesa de temps invertida en el còmput, 0.19682 ms —8% de la durada total del cicle—, és mínima front el temps d'espera inactiva —2.3 ms, 92% de la durada total del cicle—.

Aquests nombres posen en evidència la gran capacitat de càlcul del dispositiu LPC2119, tot demostrant que la no materialització del pèndul invertit no és precisament per culpa del microcontrolador.

Independentment de l'apreciació anterior, i disposant d'un marge d'inactivitat tan ampli, podem aprofitar aquest temps d'esperar per exterioritzar les dades amb les que LPC2119 treballa. Així, en cada iteració, bolquem els valors de les acceleracions *y* i *z*, i els valors computats $PIDM_y$ i $PIDM_z$ a través del port sèrie RS-232.

Aquesta metodologia ens permet capturar els valors d'un intent de rectificació real, tot visualitzant el seu comportament en l'ordinador.

Observem quina és la resposta PID M davant una caiguda amb intent de rectificar:

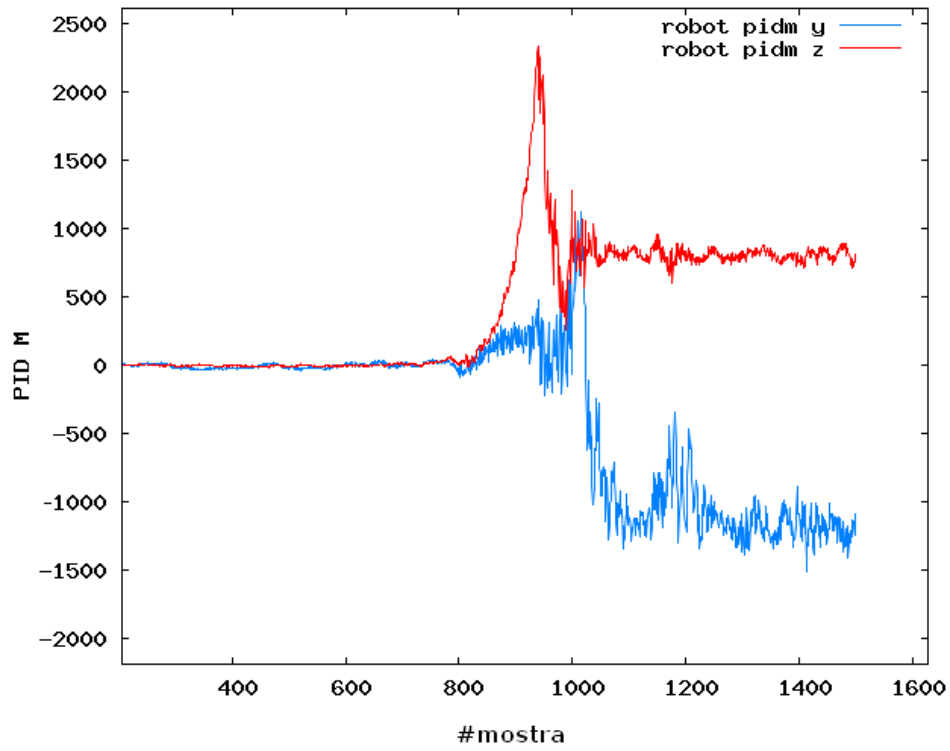


Figura A-1: Valors $PIDM_y$ i $PIDM_z$ des del dispositiu LPC2119 durant una caiguda a l'esquerra (**Figura 4-30**) amb intent de rectificació (desplaçament a l'esquerra)

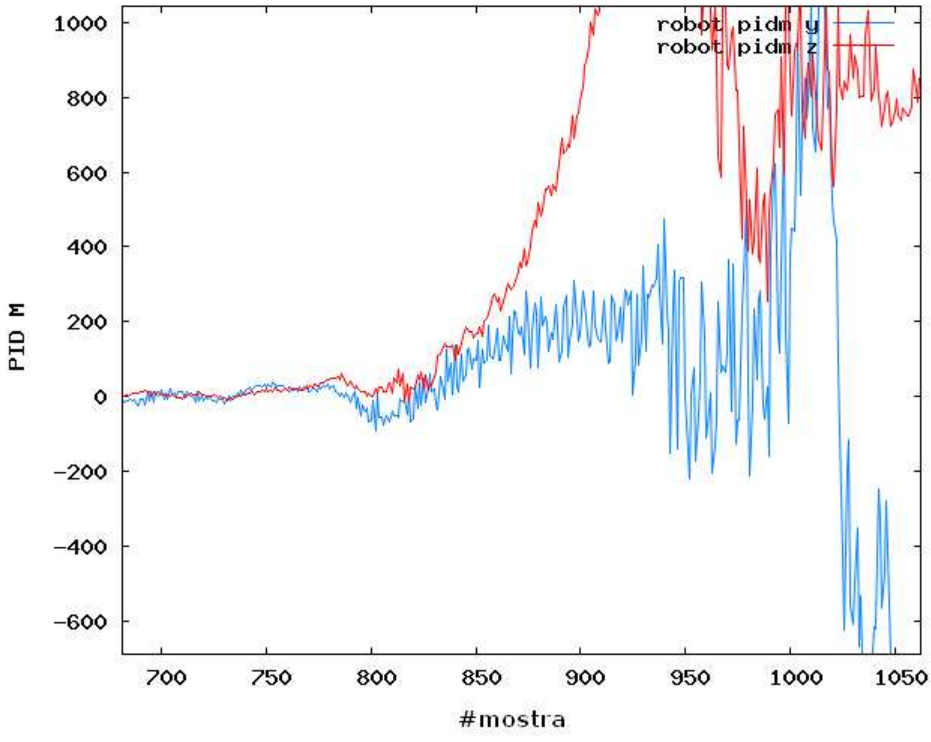


Figura A-2: Ampliació imatge en el moment de la caiguda (**Figura A-1**)

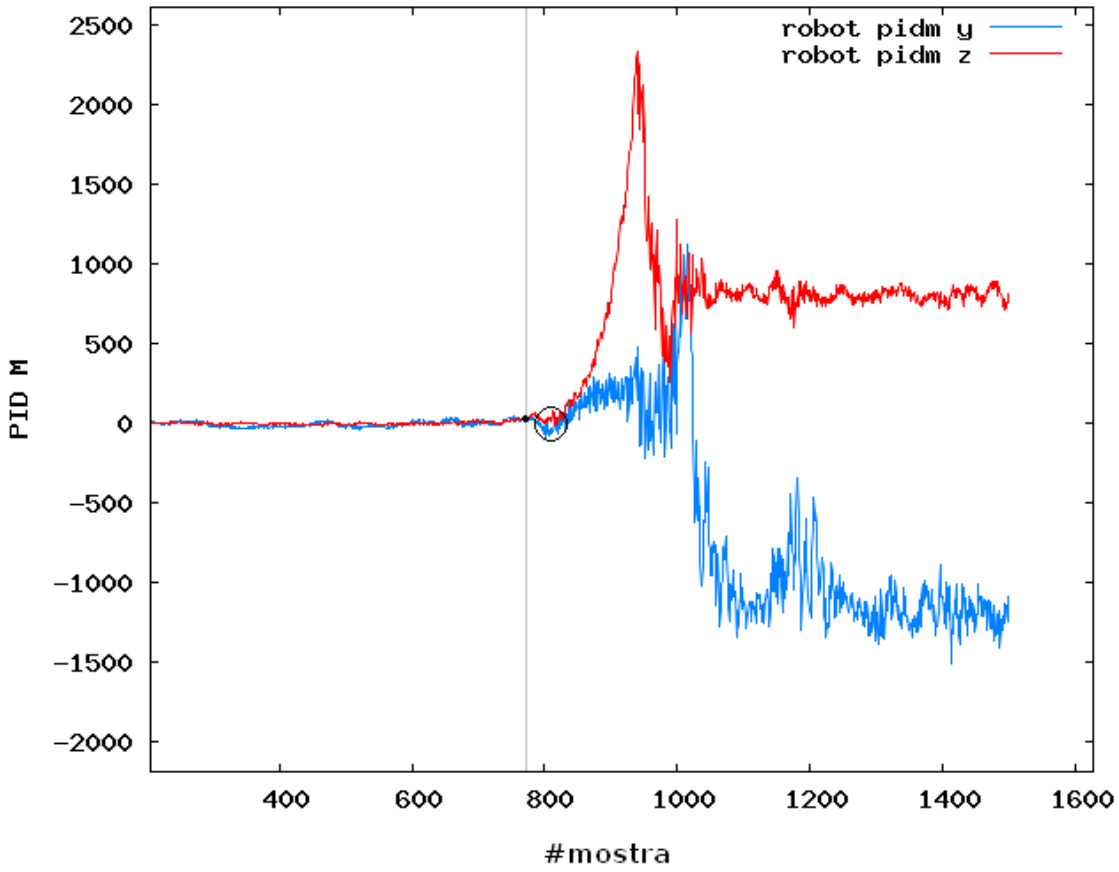


Figura A-3: Senyalització del moment de la caiguda i petita recuperació (**Figura A-1**)

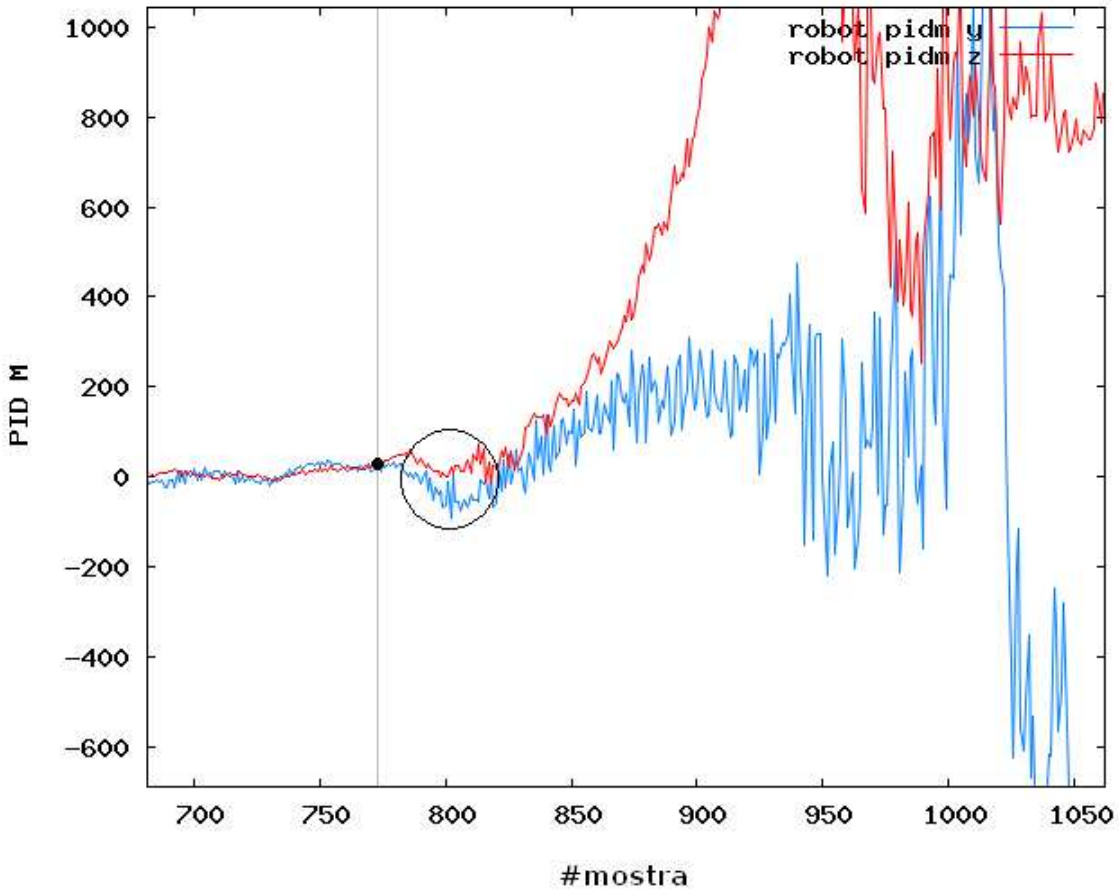


Figura A-4: Senyalització del moment de la caiguda i petita recuperació (**Figura A-2**)

Quina ha estat la prova i què ens diuen aquestes gràfiques? La prova tenia com a condició, un cop detectada la caiguda, aplicar la major velocitat possible en el sentit de la caiguda. Els resultats demostren que:

- En estat estàtic —sense moviment—, els valors de control es comporten segons la previsió —estables entorn l'eix d'abscisses—.
- A l'instant de la detecció de la caiguda —mostra 770— els valors de $PIDM_z$ creixen durant un breu interval, però després aquesta tendència s'aconsegueix anul·lar apropant els valors de $PIDM_z$ a 0 (**Figura A-4**). Finalment, després d'aquesta petita recuperació, $PIDM_z$ es dispara cap amunt.
- Just en el moment que es detecta la caiguda i s'activen els servomotors, s'aprecia, igual que $PIDM_z$, una petita rectificació en la tendència dels valors de $PIDM_y$ (**Figura A-4**). No obstant això, superada aquesta petita rectificació, $PIDM_y$ es descontrola. Sembla com si la portadora marqués una pujada però, aquesta està tan afectada pel soroll que fa inservible el senyal.

D'aquestes observacions s'extreuen les següents conclusions:

- L'eix *y* és més sensible al soroll. Aquest soroll no és tant perceptible en una caiguda lliure sense rectificació, però sí que es fa palès quan el robot es desplaça en un intent de recuperar la verticalitat.
- Les pertorbacions degudes al soroll fan necessari reajustar els paràmetres del procés de control $PIDM_y$. La solució passa per modificar les constants del procés $PIDM_y$ i fer un posterior filtratge d'altes freqüències.
- La rectificació només és possible aplicant una força major (major acceleració).

Bibliografia

ARM

- [ARM05] **ARM Ltd.**, *ARM Architecture Reference Manual*
- [ARM06] **ARM Ltd.**, *ARM7TDMI-S Technical Reference Manual*
- [AQRC07] **ARM Ltd.**, *ARM and Thumb-2 Instruction Set, Quick Reference Card*
- [Fubert00] **Steve Furber**, *ARM System-on-Chip Architecture*
- [Martin06] **Trevor Martin**, *Introduction to LPC2000*
- [Olaf07] **Olaf**, *ARM7 Performance*
- [Keil09] **Keil**, *One-Line Manuals*, http://www.keil.com/support/man_arm.htm
- [Pie02] **Ville Pietikäine**, *Embedded 3 ARM*
- [Ryz06] **Leonid Ryzhyk**, *The ARM Architecture*
- [PeSt04] **P. Knaggs, S. Welsh**, *ARM: Assembly Language Programming*
- [Anon00] **Anònim**, *ARM Processor Architecture (II)*
- [Wiki01] **Wikipèdia**, *ARM Architecture,* http://en.wikipedia.org/ARM_architecture
- [ARM09] **ARM Ltd.**, *ARM Information Center*, http://infocenter.arm.com/help/index.jsp

LPC2119

- [NPX06] **NXP**, *LPC2119, LPC2129, LPC2194, LPC2290, LPC2292, LPC2294 User Manual*
- [NXP106] **NXP**, *LPC2119, LPC2129 Data Sheet*
- [HIT05] **Trevor Martin,** *The insider's guide to the Philips ARM-7 based microcontrollers*

Miscel·lània

- [ST07] **ST**, *LIS302DL. Datasheet*
- [ST08] **ST**, *LIS302DL. Application Notes*
- [Fscale07] **Mathieu Forget,** *Solutions Based in Accelerometer*
- [Gnuplot07] **T. Williams & C. Kelley**, *gnuplot, An Interactive Plotting Program,* http://www.gnuplot.info/documentation.html
- [Wiki02] **Wikipèdia***, PID,* http://en.wikipedia.org/wiki/PID_controller
- [Parllx00] **Parallax**, http://www.parallax.com
- [Futur00] Futurlec, http://futurlec.com/index.shtml
- [Robot00] Robotshop, http://robotshop.ca/

Resum

Estudi de l'arquitectura i prestacions del microcontrolador LPC2119 tot implementant la proposta d'un cas pràctic.

En la besant teòrica, es fa una anàlisis acurat del dispositiu LPC2119, enumerant les principals característiques i exposant les seves parts, aprofundint sobretot en l'arquitectura i core ARM que incorpora.

En l'àmbit pràctic, s'introdueix el problema del pèndul invertit com a proposta per a ser integrada sobre un robot que exploti les funcionalitats del dispositiu integrat presentades a l'estudi teòric.

Resumen

Estudio de la arquitectura y prestaciones del microcontrolador LPC2119 implementando un caso práctico.

En la vertiente teórica, se efectuá un detallado análisis del dispositivo LPC2119, enumerando sus principales características y exponiendo sus partes, dando especial importancia a la arquitectura y core ARM que incorpora.

En el ámbito práctico, se introduce el problema del péndulo invertido como propuesta para ser integrada sobre un robot que explote las funcionalidades del dispositivo presentadas en el estudio teórico.

Abstract

Case study about LPC2119 microcontroller's architecture and performance realizing a practical case.

This paper explores in depth the LPC2119 device, listing its main features and exposing its components, with particular emphasis on ARM architecture and the ARM core inside.

On a practical level, we introduce the problem of inverted pendulum as a proposal to integrated it on a robot exploits the capabilities of the device presented in the theoretical case study