Università degli Studi di Catania
Dipartimento di Matematica e Informatica
Corso di Laurea in Informatica Applicata

Daniela Mangione

Controllo di una sospensione cardanica

Relazione progetto finale

Relatore
Prof. Giuseppe Scollo
Correlatore
Prof. Corrado Santoro

Anno Accademico 2013/2014

Ai miei meravigliosi genitori, al mio fratellino Emanuele
e a quanti in questi anni hanno creduto in me.

Sommario

Questa tesi si propone di dare una soluzione al problema del controllo di una sospensione cardanica attraverso la programmazione di un sistema embedded.
Una sospensione cardanica è una struttura ad uno o più assi che garantisce all'oggetto, posto su di essa, stabilità e un posizionamento ottimale in una direzione angolare stabilita, nonostante i movimenti a cui il sostegno della sospensione potrebbe essere sottoposto. Si tratta di un meccanismo di base che risulta conosciuto già da tempo, ma che può essere migliorato integrandolo in un sistema embedded ottimizzato per il monitoraggio e il controllo dello stesso.
Per la creazione di un sistema di questo tipo ci si è affidati alla programmazione di un microcontrollore che effettua elaborazioni sulla base di dati raccolti attraverso un sensore esterno, collegato ad esso; in uscita si effettua la trasduzione dei dati mediante degli opportuni attuatori.
L'applicazione di un microcontrollore è volta a minimizzare gli errori di posizionamento e a garantire un controllo efficace.

L'intero codice sorgente inerente la programmazione del microcontrollore viene rilasciato nel pubblico dominio, sul sito ARS-lab[1] dell'Università degli Studi di Catania.

Indice

Capitolo 1

Introduzione

> *La tecnologia non tiene lontano l'uomo dai grandi problemi della natura, ma lo costringe a studiarli più approfonditamente.*
> (Antoine de Saint-Exupéry)

La tecnologia in continua evoluzione è un elemento ormai imprescindibile nelle attività della vita quotidiana e lavorativa, e tutti, chi più chi meno, la portano con sé sotto forma di dispositivi elettronici, ne sono un esempio le chiavette USB. Con il passare del tempo, talvolta nemmeno anni, ci vengono proposti prodotti sempre più sorprendenti dalla potenza di elaborazione talvolta esagerata e in grado di soddisfare i bisogni più eterogenei.
I dispositivi diventano *micro* e ciò li apre alla possibilità di essere inseriti in dispositivi più grandi in modo omogeneo. Nella ricerca di questo connubio prestazioni/dimensioni nascono idee alternative rispetto a quanto conosciuto, ne sono un esempio i microcontrollori.
L'avvento di micro-tecnologie ha poi dato vita a nuovi scenari, come la nascita di settori di ricerca tecnologica alternativi e discipline come la domotica e la telemetria che combinate insieme propongono prodotti per la salvaguardia della sicurezza in casa e il comfort quotidiano negli ambienti antropizzati (resi confortevoli dalla mano dell'uomo). La telemetria, in particolare, ha come ambito di ricerca l'uso di segnali per la localizzazione e il controllo dello stato di una o più apparecchiature a distanza. Per avere esempi concreti di sistemi intelligenti basti pensare agli smartphone, al sistema ABS delle automobili,

oppure ancora a sistemi meno sofisticati come fotocamere, forni a microonde e lavatrici.
Il problema che si intende risolvere in questa tesi è il controllo di una sospensione cardanica di cui si dovranno regolare i sostegni in modo tale da ottenere un movimento controllato mediante il quale favorire il corretto posizionamento di un oggetto in una direzione fissata. Tale oggetto potrebbe essere una webcam.
Si è deciso di affrontare questo problema dal punto di vista software, ovvero mediante la programmazione in linguaggio C di un firmware adeguato allo scopo. Per eventuali, successive, implementazioni di questa stessa soluzione migliorata o ulteriori nuove versioni, non si esclude la possibilità di utilizzare un approccio di tipo hardware.
Per questa prima parte ci si limiterà a dare delle definizioni inerenti a quelli che sono i termini ricorrenti a cominciare da quelli più in uso.
Una sospensione cardanica è un dispositivo meccanico ad uno o tre assi che facendo da sostegno ad un oggetto gli conferisce stabilità permettendogli di spaziare nelle tre direzioni mantenendo fissa la direzione malgrado le oscillazioni esterne si oppongano a ciò. Nella Figura 1.1 la sospensione viene mostrata come tre anelli e come si può notare l'anello verticale ruota intorno al perno V e sostiene l'asse CD. Il secondo anello può ruotare sull'asse CD e sostiene l'asse AB, intorno al quale può ruotare il terzo anello.

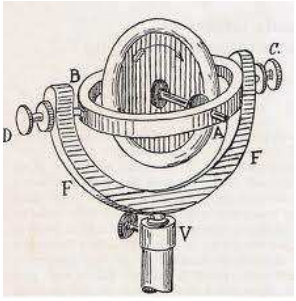

Figura 1.1: Sospensione cardanica

Figura 1.2: Bussola Galileo-Cardano

La sua invenzione viene ricondotta al matematico Gerolamo Cardano (1501-1576). Tipicamente, in tempi remoti, la si poteva trovare sulle navi come

sostegno per le bussole (Figura 1.2) o agli orologi nautici, così da mantenerle stabilmente poggiate sul piano orizzontale. Oggi sono strumenti utili in ambito aeronautico nelle fasi di beccheggio (Figura 1.3) o rollio (Figura 1.4) degli aerei e in quello militare nel controllo dei missili.

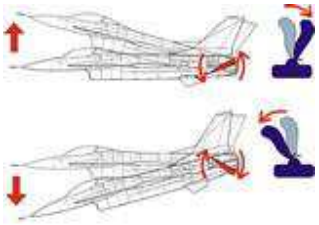

Figura 1.3: Beccheggio

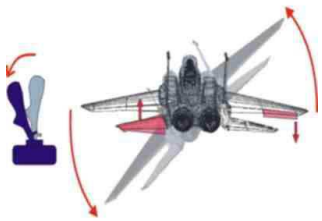

Figura 1.4: Rollio

I microcontrollori sono dispositivi elettronici integrati in un solo chip, versatili e perciò utilizzabili anche per controllare una sospensione cardanica. Sono autonomi dal punto di vista hardware poiché non contengono la sola logica computazionale ma anche memoria permanente (ROM), volatile (RAM) e un certo numero di canali input/output (I/O). Non hanno una potenza di calcolo eccezionale e possono contare su dispositivi esterni solo per ottenere informazioni e non come ulteriore sostegno in termini di prestazioni, ma ben si prestano ad attività mirate nel campo dei sistemi embedded.
Per sistema embedded si intende una famiglia di calcolatori che si occupa di risolvere sottoproblemi all'interno di sistemi più grandi oppure di servire applicazioni specifiche (Special Purpose).
A decretare il successo dei microcontrollori, oltre al costo accessibile, alla versatilità d'utilizzo e alle dimensioni che li rendono facilmente integrabili in altre progettazioni hardware, c'è anche l'applicazione di una politica *open hardware*. Si tratta della medesima politica *open source* relativa al software che viene estesa all'hardware in una modalità simile ancora legata a licenze e brevetti, seppure non manchino problematiche dovute alla distribuzione del manufatto e quindi a costi di produzione.

Capitolo 2

Sistemi embedded e microcontrollori

Un sistema di calcolo di tipo embedded si presta ad applicazioni specifiche in contesti dove le funzioni da svolgere sono raccolte in un bacino specializzato. Questi sistemi prendono anche il nome di *Special purpose*, ovvero unità con caratteristiche dedicate alla soluzione di un determinato problema. Si differenziano dai sistemi di tipo *General purpose* che invece sono, come suggerisce il nome stesso, ad uso generale; un esempio su tutti: i personal computer (PC) che possono essere specializzati mediante l'installazione di software. I sistemi finalizzati all'esecuzione di attività specifiche generalmente non sono riprogrammabili e conseguentemente non li si può usare per lo svolgimento di compiti diversi da quelli per cui sono nati. Tuttavia sono sistemi di dimensione spesso ridotta e per questo integrabili in sistemi più grandi per applicazioni riguardanti il monitoraggio e il controllo di un processo o di un dispositivo. In un certo senso, per la loro versatilità e l'ingombro fisico quasi del tutto inesistente, i sistemi embedded possono essere, ciascuno, la soluzione ad un sottoproblema relativo ad un problema più ampio. Possiamo riferirci a questa famiglia di calcolatori definendoli *sistemi reattivi* poiché l'esecuzione di una routine di istruzioni è legata allo scatenarsi di un evento specifico, captato dai *sensori*, a cui il software è programmato per reagire mettendo in azione elementi hardware detti *attuatori*. I sensori possono es-

sere integrati sulla scheda oppure essere di tipo esterno. In ogni caso il loro compito sarà la raccolta di dati dall'ambiente circostante al calcolatore che saranno successivamente elaborati. Gli attuatori invece sono gli elementi mediante i quali il calcolatore agisce sull'ambiente esterno al sistema embedded.
In un sistema embedded, un microprocessore è quella componente hardware adibita all'elaborazione dei dati e ai calcoli utili al loro trattamento. È il cuore del sistema e si appoggia ad altre risorse che nel caso dei PC possono essere ingenti e quindi in quel caso ci troviamo davanti a microprocessori di una certa dimensione capaci di elaborare molte informazioni per volta in tempi brevissimi. Quando invece il microprocessore è integrato in un chip e montato su una scheda che ospita al suo interno, anche, una piccola quantità di memoria ROM, una modesta quantità di RAM e pochi collegamenti di tipo I/O allora in quel caso possiamo designare quel componente con la denominazione microcontrollore.
Il microcontrollore (MicroController Unit- MCU) che viene usato spesso nei sistemi embedded, è un dispositivo di dimensioni contenute, molto versatile, potenzialmente in grado di essere la risposta per un certo numero di problemi legati al monitoraggio dello stato e al controllo di apparecchiature elettroniche. Ad oggi sono disponibili 3 fasce di microcontrollori che si distinguono per capacità elaborativa e per l'ampiezza del bus: 8, 16 e 32 bit. Le componenti principali sono: l'unità di elaborazione (CPU); memoria permanente (ROM, EPROM, FLASH); memoria volatile (RAM e EEPROM); oscillatore interno o esterno; porte I/O e/o GPIO (General Purpose I/O) configurabili; gestione interrupt; moduli aggiuntivi eventuali (contatori, timer, moduli di comunicazione, interfacce di visualizzazione, ecc). Sono dotati di una memoria di massa molto ridotta che al suo interno ospita un software, anch'esso specializzato, che prende il nome di *firmware* e agisce molto da vicino con l'hardware. Tuttavia esistono anche sistemi in grado di ospitare un sistema operativo (SO), come nel caso dei calcolatori Raspberry che sono compatibili anche con l'uso di una versione ad hoc di Linux.
Mediante il sostegno dei dati, posti nella seguente tabelle, poniamo in evidenza le caratteristiche che contraddistinguono i microprocessori dai microcontrollori.

Caratteristiche	**Microcontrollori**	**Microprocessori**
Velocità massima del clock	200 MHz	4 GHz
Potenza elaborativa(MegaFlops)	200	5.000
Potenza min dissipata in fase di elaborazione	0.001 Watt	50 Watt
Num pezzi venduti in un anno	11.000 Mln	1.000 Mln

Commentando quanto espone la Tabella 1, notiamo che quanto a potenza elaborativa così come anche a velocità del clock, i microprocessori non hanno eguali e sono nettamente più rapidi e potenti dei microcontrollori, ma che conseguentemente hanno anche un maggior impatto sulla dissipazione energetica.
Focalizzando la riflessione anche e soprattutto sul numero di pezzi venduti,di gran lunga a favore dei microcontrollori, che stanno trovando un certo riscontro sul mercato, i motivi del loro successo, misurato dalla crescita della domanda, sono probabilmente imputabili:

- innanzitutto al basso costo che permette loro di essere utilizzati come dispositivi sostitutivi rispetto ai ben più onerosi circuiti integrati tradizionali;
- alla versatilità di utilizzo e all'autonomia funzionale che riduce a zero la necessità di componenti esterni per effettuare elaborazioni;
- ai tempi brevi di introduzione sul mercato che li hanno resi facilmente reperibili a quanti ne fanno richiesta;
- a quanto sia relativamente facile programmarli e alla possibilità di riprogrammare il firmware grazie agli strumenti e alla documentazione presente in rete, favorita dall'attuazione di una politica distributiva Open Source Hardware;
- all'inventiva degli utenti della rete Internet che intorno a questi progetti, a queste innovazioni, ha creato delle comunità, ha diffuso idee, ha

messo in luce delle problematiche e quindi permesso un miglioramento dei prodotti.

Gli ambiti applicativi sono tanti proprio per la versatilità di questi oggetti, ne sono esempi: nel settore automobilistico i sensori di parcheggio e gli antifurto, nella telefonia gli smartphone, i sensori per la regolazione automatica della luminosità.

Capitolo 3

Open Source Hardware

Figura 3.1: Logo Ufficiale

Il concetto di Open Source denota la condivisione di materiali quali documenti, schemi e codici utili alla piena divulgazione di un proprio progetto mediante la rete Internet. Lo stesso concetto valido per il software può essere esteso all'hardware benché sia presente almeno una differenza oggettiva non da poco: la presenza di un componente tangibile.
Definiamo hardware tutte quelle componenti elettriche, elettroniche, magnetiche, ottiche e meccaniche che caratterizzano l'architettura di un calcolatore.

3.1 Definizione Ufficiale

Lo scorso 10 Febbraio, dalla collaborazione di molte personalità illustri, come Massimo Banzi del team Arduino, è nata la definizione ufficiale e completa di Open Source Hardware. Essa è basata sulla definizione di Open Source

Software e sulla definizione Open Source Definition di Bruce Perens e degli sviluppatori Debian.

> "L'Open Source Hardware (OSHW) è il termine di artefatti tangibili- macchine, apparecchi, o altre cose fisiche – il cui design è stato rilasciato al pubblico in modo tale che chiunque possa fare, modificare, distribuire e utilizzare queste cose". [2]

Quindi, quando si parla di Open Hardware (trad. Hardware libero) in particolare si ci riferisce alla divulgazione di schemi logici, progettuali e manuali riguardanti il manufatto che spesso viene accompagnato da strumenti software (tool) utili tanto all'utilizzo quanto alla modifica del progetto stesso.
A differenza del software, l'hardware richiede la produzione di risorse fisiche per la creazione dei beni materiali. Di conseguenza chi ne produce sotto licenza OSHW ha obbligo di mettere in chiaro che tali strumenti non verranno fabbricati, venduti né coperti da garanzia dal progettista originale.

> "L'espressione "Software Libero" si riferisce alla libertà dell'utente di eseguire, copiare, distribuire, studiare, cambiare e migliorare il software". [3]

Per libertà si intende in particolare quella di poter adattare il software alle proprie esigenze, quindi poterlo ridistribuire e pubblicarlo senza vincoli.

3.2 Copyright e diritto d'autore

Il Diritto d'autore conferisce al programmatore diritti di tipo patrimoniali. Secondo la legge italiana questo conferimento si arricchisce di una particolarità rispetto alla normativa applicata negli stati europei dove vige la cosiddetta "civil law", ovvero, l'autore pur cedendo i diritti patrimoniali della sua opera, può conservare un certo controllo su di essa. Cioè gode anche di diritti di tipo morale che sono per legge: irrinunciabili, incedibili e perpetui.
Copyright (simbolo: ©) è una parola inglese la cui traduzione letterale coincide con "diritto di copia" e rappresenta l'insieme delle normative sul diritto d'autore in vigore.

Il software, in quanto creazione intellettuale, si colloca giuridicamente nella categoria dei beni immateriali. Ad essi si applica il diritto d'autore.

3.3 Licenze

Una licenza d'uso è un contratto che accompagna il prodotto software/hardware e specifica le modalità secondo le quali l'utente potrà utilizzarlo, quindi quali sono i suoi diritti e gli obblighi a cui deve sottostare. La licenza viene impostata da chi detiene il copyright del prodotto e la sua validità dipende dalla presenza del diritto d'autore.
Piuttosto che creare di volta in volta, per ciascun prodotto, una nuova licenza alcuni produttori si riuniscono nell'impiego di una stessa, preesistente di stampo open source. Quindi si parla di licenze condivise, tra le quali troviamo: LGPL, GPL, copyleft, MIT, TAPR, CERN, ecc.

Criteri di distribuzione delle licenze

Sono in tutto 12 i criteri che la licenza open source hardware dovrà avere per essere considerata valida ai fini della distribuzione di un prodotto.

1. Documentazione: la documentazione dell'hardware va rilasciata insieme al prodotto fisico oppure su siti internet e altre fonti ben pubblicizzate affinché sia chiaro dove prelevarla. Tale materiale deve includere i file di progetto, quindi gli schemi senza offuscamenti, in un formato adeguato e tale da poter essere modificati.

2. Scopo: all'interno della documentazione stessa dovranno essere segnalate le parti del progetto che sono state rilasciate sotto licenza, laddove non sia l'intero progetto posto a disposizione.

3. Il software necessario: dovrà essere rilasciato sotto licenza approvata da OSI (Open Source Iniziative) ciò che a livello software sarà necessario al funzionamento dell'hardware, oppure si dovranno documentare

adeguatamente le interfacce in modo che possa essere scritto il codice che consente al dispositivo di adempiere alle sue funzioni.

4. I lavori derivanti: ciò che viene ricavato da precedenti lavori già sotto licenza OSHW potrà essere rilasciato ancora sotto i medesimi termini. La licenza deve anche garantire la possibilità di produrre derivati del prodotto originale al fine di poterli vendere e di distribuire i prodotti creati dai file di progettazione.
   Sarebbe un vincolo troppo grande e certamente qualcosa di controproducente impedire il ricavo di introiti da parte dei progettisti relativamente ai loro lavori.

5. Ridistribuzione libera: la licenza non può limitare la vendita o la donazione della documentazione del progetto. La licenza non richiede royalty o tasse relative alla vendita del lavoro derivato.

6. Attribuzione: la licenza può richiedere documenti derivanti e comunicazioni di copyright e che l'utente finale abbia diritto ad accedere a tali informazioni nell'utilizzo del dispositivo.

7. Nessuna discriminazione verso un gruppo di persone.

8. Nessuna discriminazione riguardante campi di ricerca e di applicazione.

9. Distribuzione della licenza: i diritti della licenza si applicano a tutti coloro ai quali il lavoro viene redistribuito senza avvalersi di ulteriori licenze per queste parti.

10. La licenza deve non essere specifica per un prodotto: i diritti concessi non devono dipendere dalla parte del lavoro che si è svolto all'interno del progetto. Tutti dovrebbero avere gli stessi diritti concessi per il lavoro originale.

11. La licenza non deve limitare altro software/hardware: non si devono porre restrizioni ad altri elementi che sono integrati nel lavoro di licenza, ma non derivanti da esso. Per esempio non si deve insistere sul fatto che

tutti gli altri componenti hardware siano distribuiti con licenza open source.

12. La licenza deve essere tecnologicamente neutrale: non può cioè essere basata su tecnologia individuale, parti o componenti o lo stile di interfaccia d'uso.

3.4 Brevetti

Le idee sono frutti di una mente caparbia che si ingegna per la risoluzione di una problematica personale e queste possono essere condivise con il mondo come anche chiuse, metaforicamente parlando, nel cassetto. Ma nel momento stesso in cui si crede nei propri progetti e si hanno delle aspettative su di essi, allora è bene che vengano depositati sulla rete allo scopo di far conoscere la propria risposta ad un problema comune e porvi rimedio. Dando il proprio contributo alla causa, intanto, si potrà migliorare il proprio stesso progetto e portarlo ad un nuovo livello, grazie ai feedback di quanti sono disposti a provarlo, in secondo luogo si potrebbero anche aprire scenari su un problema che non era stato fino a quel momento molto curato. Se poi si è convinti di avere in mente l'idea del secolo e che la realizzazione di un prototipo, qualsiasi cosa sia, possa essere innovativa e se ne vogliono avere pieni diritti, allora sarà il caso di brevettarla.
Un brevetto è un documento giuridico, descritto in linguaggio tecnico e rilasciato da un ente preposto, che conferisce all'autore una sorta di monopolio temporaneo in un territorio dell'innovazione e quindi garantisce al titolare il diritto esclusivo di sfruttamento, che nessuno potrà utilizzare nei propri lavori tranne a non ricevere un deroga.

> "Un brevetto tutela e valorizza un'innovazione tecnica, ovvero un prodotto o un processo che fornisce una nuova soluzione a un determinato problema tecnico. È un titolo in forza del quale viene conferito un monopolio temporaneo di sfruttamento sull'oggetto del brevetto stesso, consistente nel diritto esclusivo di realizzarlo, di disporne e di farne un uso commerciale, vietando

> tali attività ad altri soggetti non autorizzati. [...]. È importante notare che un brevetto non attribuisce al titolare la "libertà d'uso" o il diritto di sfruttare la tecnologia coperta dal brevetto, ma solo il diritto di escludere dall'utilizzo dello stesso altri soggetti." [4]

La scadenza non è prorogabile e nemmeno rinnovabile.
L'invenzione per essere brevettata deve soddisfare i seguenti requisiti.

- Essere originale e innovativa rispetto a quanto già conosciuto;
  Offrire un miglioramento relativamente ai risultati tecnici già presenti.
- Avere un'applicazione pratica.
  Essere descritta in modo chiaro e completa in modo da risultare pienamente comprensibile.
- Non deve essere stata predivulgata né da chi vuole brevettare né da terzi.

Il meccanismo di funzionamento del brevetto è stabilito dalle leggi che variano a seconda delle nazione che lo disciplina. L'elenco di ciò che può essere oggetto di brevetto non è tassativo, ma esclude a priori:

- le scoperte, come quella di elementi presenti in natura;
- le teorie scientifiche;
- i metodi matematici;
- i metodi per il trattamento chirurgico o terapeutico del corpo umano.

Depositare un brevetto e quindi mantenerlo anno per anno impone un costo che nel caso di invenzioni industriali non è indifferente. In particolare i costi assumono cifre più importanti al superamento del quinto anno. Si presume che l'autore dopo i primi 5 anni cominci a guadagnare qualcosa di più consistente e quindi non abbia problemi a sottostare ad una tassazione superiore. In Italia, il solo deposito di un brevetto per vie telematiche ha un costo pari a 50,00 euro, mentre per via cartacea varia a seconda del numero delle pagine

da 120 euro fino anche ai 600 euro.

Curiosità: In Italia, il primo brevetto venne depositato nel 1421, con una durata di 3 anni, a nome di Filippo Brunelleschi, architetto fiorentino, la cui idea fu una chiatta con mezzi di sollevamento che si rivelò utile nel trasporto del marmo attraverso il fiume Arno per la costruzione del Duomo di Firenze.

Capitolo 4

Architettura del sistema di controllo

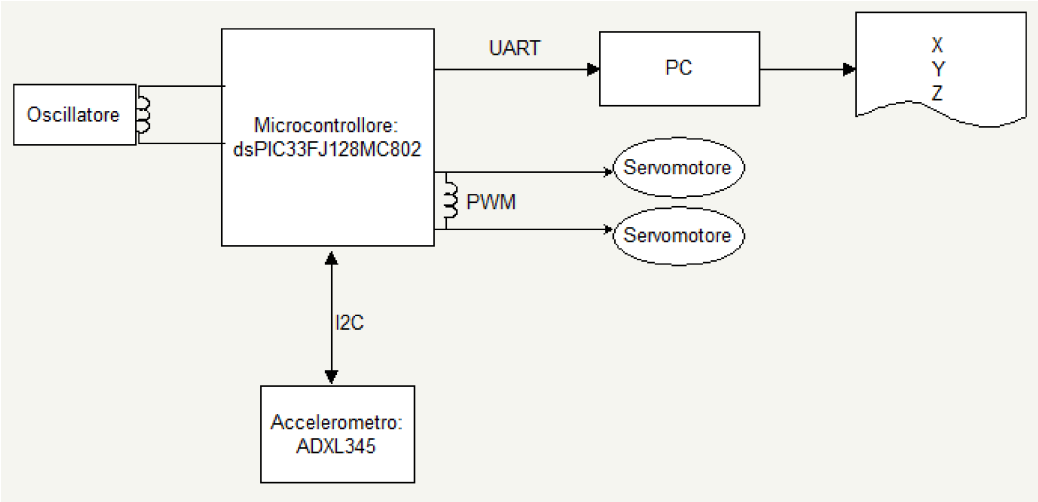

Figura 4.1: Architettura del sistema

4.1 L'idea progettuale

Si vuole realizzare il controllo di una sospensione cardanica mediante l'ausilio di un microcontrollore (dsPIC33FJ128MC802). Poiché al suo interno questo modello non dispone di sensori adatti alla misurazione di quanto utile

al controllo stesso, è indispensabile collegare al dispositivo un accelerometro esterno (ACXL345) e due attuatori rappresentati da due servo motori.
Il tutto è programmato in linguaggio C facendo uso dei relativi "datasheet" e del software IDE MPLAB X, forniti secondo una licenza di tipo open source hardware (Capitolo 3) mediante il sito del produttore (Microchip).
Facendo uso dell'accelerometro si raccolgono dati inerenti l'accelerazione angolare dell'oggetto posto sulla sospensione cardanica. Tali misurazioni sono inerenti i 3 assi X, Y e Z, attraverso le quali (mediante l'ausilio di calcoli appositamente impostati) sarà possibile ricavare ulteriori misure: pitch e roll.

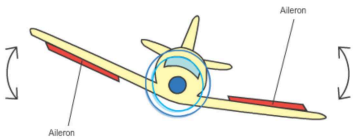

Figura 4.2: Roll

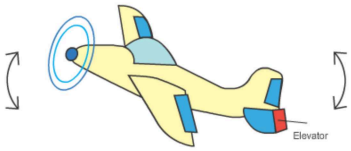

Figura 4.3: Pitch

Come è possibile notare dalle figure, per Roll si intende il movimento a "rotolare" verso sinistra o verso destra (corrispondente al rollio, Figura 1.4); mentre per Pitch il movimento a salire o a scendere dal basso verso l'alto o viceversa (corrispondente al beccheggio, Figura 1.3). La loro misurazione dà un quadro definito dei movimenti dell'oggetto a cui è collegato l'accelerometro e quindi permette di impostare una reazione della sospensione mediante l'uso degli attuatori.
Il ruolo degli attuatori sarà ricoperto da dei servomotori che genereranno una reazione proporzionata agli input rilevati dal sensore.
Per la visualizzazione di tali dati è stato doveroso impostare il dispositivo UART in modo tale da permettere una connessione a corto raggio di tipo bluetooth tra il microcontrollore e un computer di cui ci siamo serviti per la visualizzazione dei dati su schermo.

Di seguito viene mostrato il diagramma a blocchi funzionale che mette in evidenza le funzioni caratteristiche del sistema.

4.1.1 Funzione Arcotangente

La funzione (a due valori) arcotangente è necessaria per il calcolo dei valori di pitch e di roll.
La sua codifica impiega la funzione di libreria atan2 che prende le due proiezioni come parametri, valutandone sia il rapporto che i segni per la determinazione dell'arcotangente.

$Pitch = \arctan(\frac{-a_y}{-a_z})$

$Roll = \arctan(\frac{a_x}{\sqrt{a_y^2+a_z^2}})$
Ecco le righe di codice relativo alle funzioni arcotangente e conversione.

```
//calcolo pitch e roll
pitch = atan2(ax, (sqrt(pow(ay, 2)+ pow(az, 2))));
roll = atan2(-ay, az);
//conversione da radianti a gradi
pitch= (pitch*180)/PG; //PiGreco
roll= (roll*180)/PG;
```

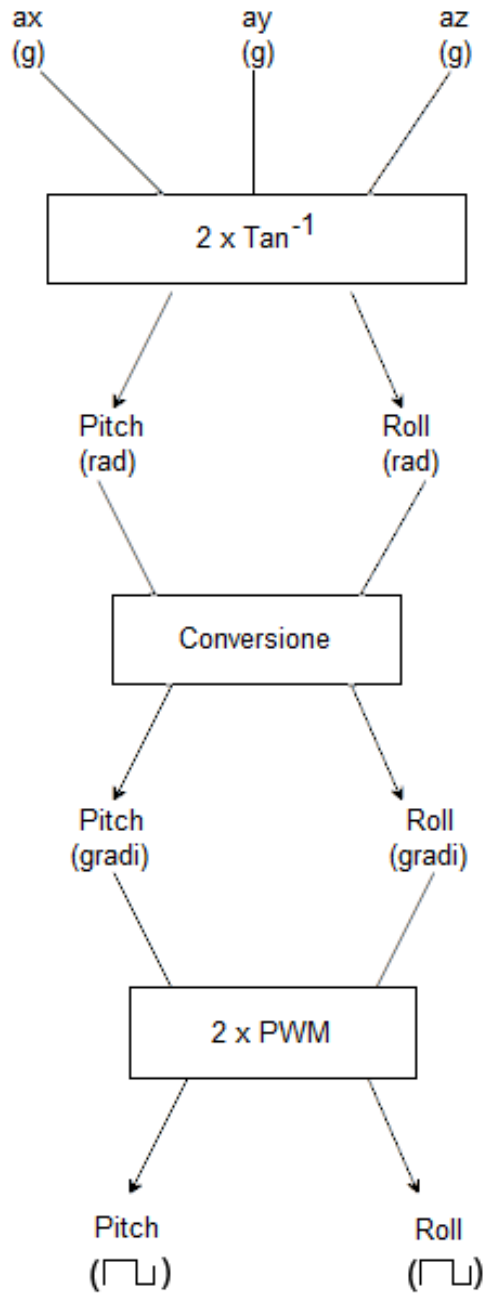

Figura 4.4: Schema funzionale della compensazione dell'errore di posizione angolare

Capitolo 5

Le componenti del progetto

Per architettura del sistema di intende l'insieme delle componenti del progetto, quindi, tutte quelle parti che sono state utili alla realizzazione dell'idea progettuale.
In questo capitolo ci si sofferma sulla descrizione nel dettaglio delle componenti tangibili (periferiche), dei moduli, delle interfacce e delle modalità di comunicazione di dati utilizzate.

5.1 I dispositivi usati

In questa sezione si descrivono le componenti elettroniche tangibili.

5.1.1 Microcontrollore PIC

Il modello usato nello specifico è il **dsPIC33FJ128MC802**[5].
La sigla PIC, presente nella nomenclatura (dsPIC33FJ128MC802) del modello del microcontrollore utilizzato, ne indica la famiglia di appartenenza, la cui produzione ebbe inizio nel 1975 ad opera della General Istrument che dieci anni dopo convertì la Microelectronics Division in Microchip Technology. Dal 1998 Microchip Technology continua a sviluppare nuovi e più potenti microcontrollori. Oggi PIC è un acronimo di "Programmable Intelligent Computer" (o anche "Peripheral Interface Controller").
La lettera "F" (dsPIC33**F**J128MC802) indica che è presente una memoria di

Figura 5.1: Microntrollore della famiglia PIC

tipo flash; inoltre, quest'ultima è di tipo riprogrammabile.
Di seguito sono riportate le caratteristiche principali.

Nome Parametro	**Valore**
Stato commerciale	In produzione
Architettura	16 bit
Velocità CPU	40 MIPS
Tipo di memoria	Flash
Memoria Programmabile	128 KB
Pin di I/O	21
Numero totale di Pin	28
Periferiche di comunicazione digitale	2 UART; 2 SPI; 1 I2C
Range delle temperature	da -40 a 150 C
Oscillatore interno	7.32 MHz, 32.769 kHz
Timer	5x16-bit e 2x32-bit
Canali PWM	8x16 bits (risoluzione)

Approfondiamo qualche voce in particolare.
La velocità della CPU è espressa in MIPS (Million Instruction Per Second) acronimo che sta per "Milioni di Istruzione al Secondo"; 40 MIPS quindi sta ad indicare che il nostro microcontrollore, in particolare, sarà in grado di eseguire al massimo 40 milioni di istruzioni al secondo.

5.1.2 Oscillatore

L'oscillatore è un circuito elettronico che genera onde periodiche. Il modello usato in questo progetto è dotato di oscillometro interno la cui frequenza massima è di circa 80 Mhz.

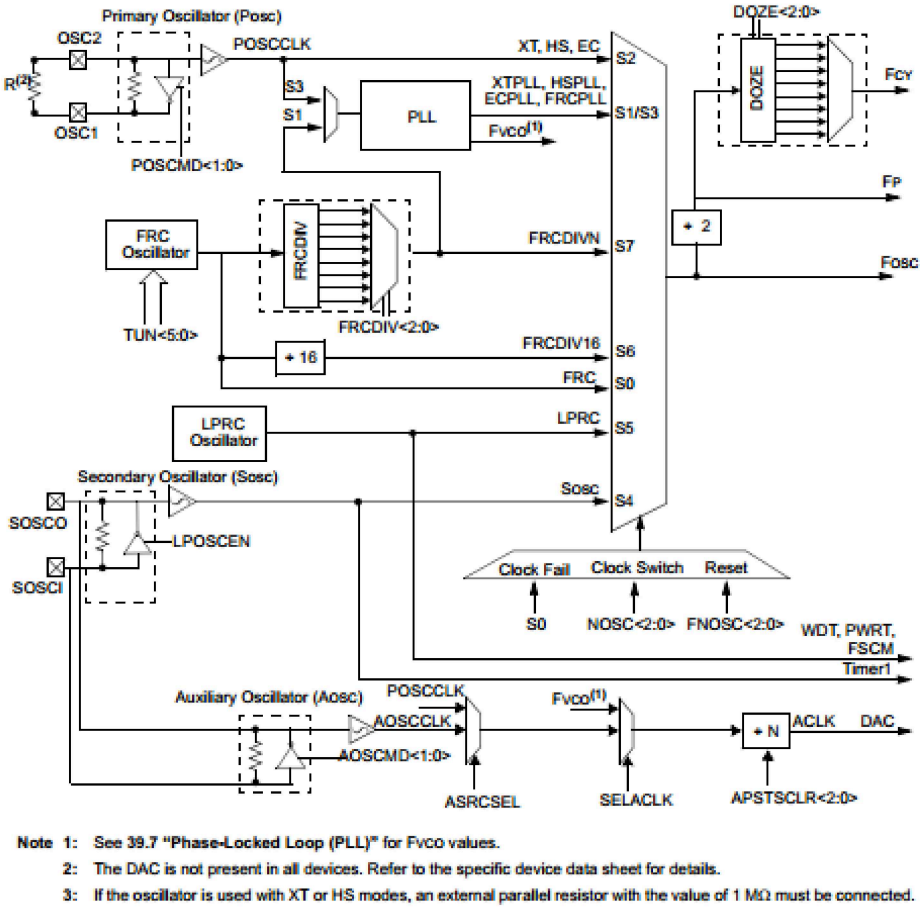

Figura 5.2: Schema oscillatore

Il segnale passa attraverso FRC Oscillator, esso può essere anche amplificato mediante la funzione di tuning, ma nel nostro caso non è stato necessario. Il segnale passa successivamente attraverso un divisore a cui si è dato il valore minimo, ovvero la divisione per 1.
A questo punto rimane da fare una scelta, relativa alla modalità secondo la

quale si vuole far funzionare l'oscillatore.
In particolare il clock può essere impostato come:

- oscillatore primario (Posc)
- oscillatore secondario (Sosc)
- oscillatore Fast RC (FRC) con divisore clock interno
- oscillatore interno Low-Power RC (LPRC)
- Posc con Phase-Locked Loop (PLL)
- oscillatore interno Fast RC con PLL

La modalità da noi usata è l'ultima elencata, ovvero: Oscillatore interno Fast RC con PLL. Per selezionarla sarà necessario effettuare una configurazione dell'apposito registro (FOSC).
In base a quanto scelto, il segnale passa attraverso il PLL.
Nella figura sottostante (Figura 5.3) il segnale viene indicato come FIN e come si può notare, passando attraverso delle divisioni (pre e post scaled), impostando quindi in un certo modo le variabili in gioco, N1, N2 ed M, sarà possibile ottenere una frequenza d'uscita vicina (seppur inferiore) agli 80 Mhz.

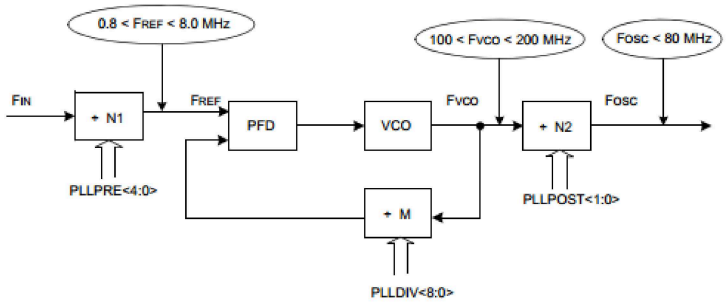

Figura 5.3: Schema PLL

5.1.3 L'accelerometro

Il modello in uso è denominato con la sigla *ADXL345*.

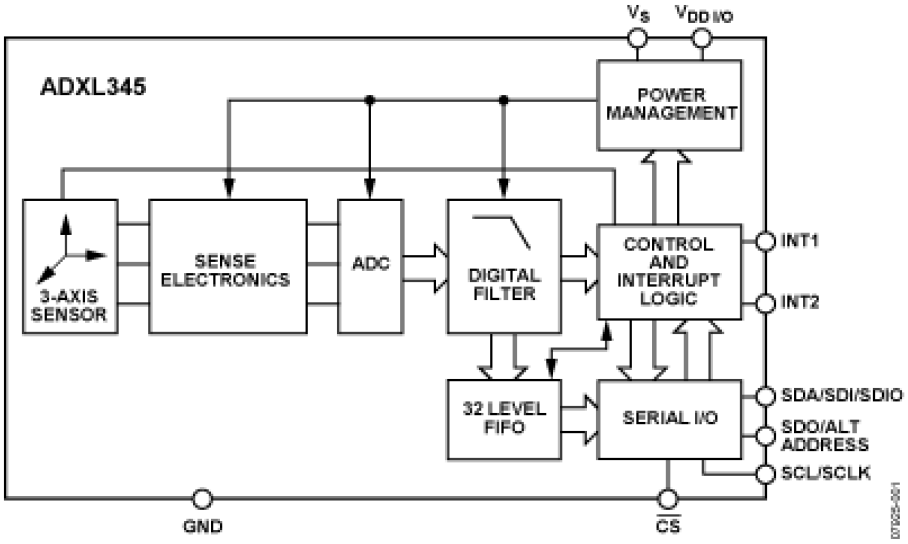

Figura 5.4: Accelerometro: ADXL345

Si tratta di un sensore a 3 assi (X, Y, Z) che presenta pin per interfacciarsi agevolmente con una scheda UART integrata nel microcontrollore.
Di dimensioni ridotte, sottile e con un'alta risoluzione pari a 13-bit. L'output, di tipo digitale, è formato da dati da 16-bit che sono accessibili mediante SPI oppure, come nel nostro caso, tramite interfaccia I2C.
Questo sensore, adatto per dispositivi mobili, misura l'accelerazione gravitazionale statica così, anche, come quella dinamica. Sono programmabili diverse funzioni speciali per rilevare la presenza di movimento e capire se l'accelerazione supera qualsiasi limite impostato dall'utente.
L'accelerazione di gravità è l'accelerazione che un corpo subisce quando è lasciato libero di muoversi in caduta libera in un campo di gravità.
Accelerazione Gravitazionale Statica
Tipicamente misurata dagli strumenti che hanno banda passante con caratteristica passa basso, in grado di rilevare accelerazioni continue e statiche con variazioni di frequenza basse, da 0 Hz a 500 Hz (normalmente).

Accelerazione Gravitazionale Dinamica
Gli strumenti che la misurano sono in grado di rilevare le accelerazioni che variano nel tempo, ma non di tipo statico: per esempio, quelle generate da un oggetto che vibra o da urti.
Attraverso la modalità basso consumo avremo una dispersione di potenza bassissima. Molto sottile, per un ingombro minimale.
Ideale per applicazioni relative alla protezione degli HDD, come equipaggiamento fitness, medico, industriale, ecc.

5.2 Interfacce e protocolli

In particolare si fa riferimento:

1. al protocollo relativo alla gestione del bus per la comunicazione microcontrollore-accelerometro e viceversa nella modalità descritta dal modello I2C;

2. all'uso del dispositivo UART come interfaccia per la raccolta dei dati dall'accelerometro in modo asincrono e seriale.

5.2.1 Bus I2C

La comunicazione tra uno o più dispositivi si può realizzare attraverso un insieme di linee chiamate bus. Un bus, in informatica, è definito come un canale che permette a periferiche e componenti del sistema elettronico lo scambio di dati e informazioni. Per un corretto uso del bus è necessario stabilire un insieme di regole da usare, ossia un protocollo.
I2C (Inter Integrated Circuit) è un sistema di comunicazione seriale bifilare utilizzato tra circuiti integrati, sviluppato da Philips nel 1982. La stessa azienda nel 1992 ha poi brevettato un protocollo di comunicazione adeguato all'uso del circuito.

Protocollo I2C

Il protocollo I2C specifica le regole di comunicazione tra i dispositivi collegati ad un bus formato da due linee bidirezionali chiamate rispettivamente SCL (Serial Clock Line) e SDA (Serial Data line) sincronizzate da un clock. Ciascuna (delle due linee) è collegata all'alimentazione attraverso una resistenza di pull-up (collegata a Vdd) in modo tale che sia presente un segnale di alto-basso, modificabile dal dispositivo.
Nella comunicazione seriale il master controllerà in modo esclusivo il segnale SCL.

Linea	**Descrizione**
Vdd	Tensione di riferimento
SCL	Clock per la sincronizzazione della comunicazione
SDA	Linea di transizione dei dati
GND	Massa di riferimento

La velocità standard varia da 10 Khz a 100 Khz, ma esistono varianti più veloci da 400 Khz a 3.4 Mhz. Non esistono limiti inferiori relativamente alle impostazioni di velocità.
Sono collegabili fino a 128 dispositivi e ciascuno avrà un indirizzo univoco da 7 o 10 bit e parteciperà alla comunicazione con il ruolo di **master** o *slave* ovviando di volta in volta al ruolo di trasmittente o ricevente.

Start (S)

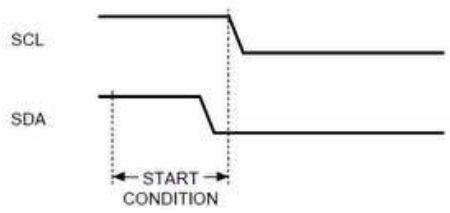

Figura 5.5: Start

Dopo essersi assicurato che il bus sia effettivamente libero, il master esegue

il segnale di Start.
Prima di inviare il comando, le linee SCL e SDA si trovano a livello logico alto.
Per inviare il comando: il segnale SCL rimane inalterato, mentre SDA viene posto al livello logico basso, così comando di start viene letto sul fronte di discesa del segnale SDA.
Facendo in questo modo segnala agli altri dispositivi che sta per iniziare una comunicazione.

Stop (P)

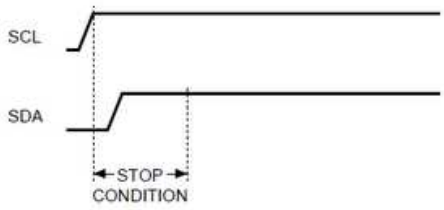

Figura 5.6: Stop

Prima di inviare il segnale di stop, si tenga presente che sono stati trasferiti dati e quindi che lo stato del bus si trova nelle condizioni per cui SDA e SCL sono a livello logico basso.
Per inviare il comando:
si pone alto SCL e successivamente anche SDA così che il segnale di stop venga letto sul fronte di salita del segnale SDA. Facendo così, si segnala allo slave che la comunicazione è terminata e che sarà scollegato dal bus.
Entrambe le linee SCL e SDA tornano a livello logico alto.

Invio e Ricezione di una sequenza di bit
Inizialmente, entrambe le linee sono a livello logico basso.
Per l'invio di una sequenza di bit(Figura 5.7)

Il master:

1. genera il segnale di start;

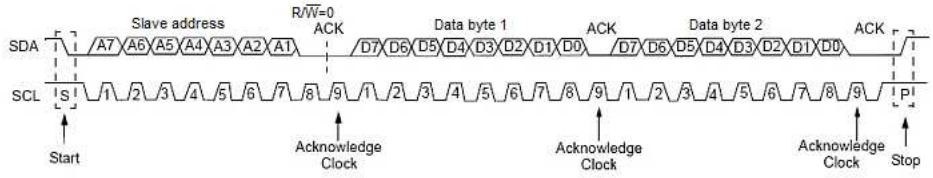

Figura 5.7: Invio di una sequenza di bit

2. imposta il valore (byte) che deve trasmettere su SDA. Il byte contiene l'indirizzo del dispositivo con cui si deve comunicare e un bit di comando (R/W) posto a 0;

3. manda in alto il segnale SCL e infine lo riporta basso;

4. attende una conferma di ricezione dallo slave (ACK) per ogni byte inviato;

5. alla fine della trasmissione genera la condizione di stop.

Ricezione di un bit(Figura 5.8)

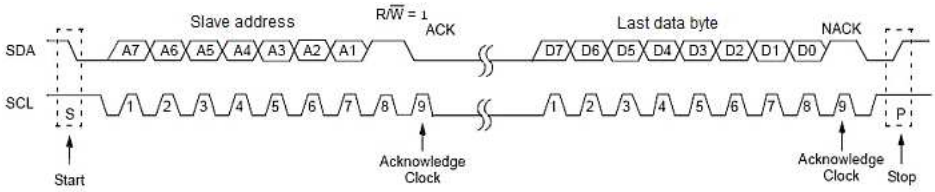

Figura 5.8: Ricezione di una sequenza di bit

Il master:

1. genera la condizione di start;

2. rilascia il bus riportando SDA a livello logico alto;

3. porta SCL alto e lo slave stabilizza SDA con il valore da comunicare;

4. legge questo valore e porta a livello basso SCL;

5. genera la condizione di stop;

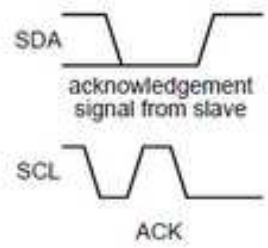

Figura 5.9: Invio bit

ACKNOWLEDGEMENT

È una funzione utile per comunicare al dispositivo che lo richiede l'avvenuta, corretta, ricezione di una sequenza di dati.
Il master rilascia la linea SDA e inizia a leggerne lo stato, mentre lo slave trasmetterà l'ack in corrispondenza con il ciclo SCL generato dal master.
Non ricevendo alcun ACK (vedi esempio Figura 5.8), il master potrà inviare un comando di STOP e inviare una nuova sequenza.

Check Address
In questo momento specifico della trasmissione ciascun dispositivo collegato al bus, che non sia il master, confronta il proprio indirizzo con quello che viene presentato sulla linea, mediante l'ausilio di un comparatore.
N.B. Durante l'intera trasmissione viene sincronizzato il clock che permette ai due dispositivi, che non hanno la stessa velocità, di comunicare evitando la perdita di dati.
Un sistema di arbitraggio permetterà la trasmissione di dati quando sono possibili comunicazioni multi-master.
Il segnale SDA può cambiare solo ed esclusivamente quando il segnale SCL è basso.

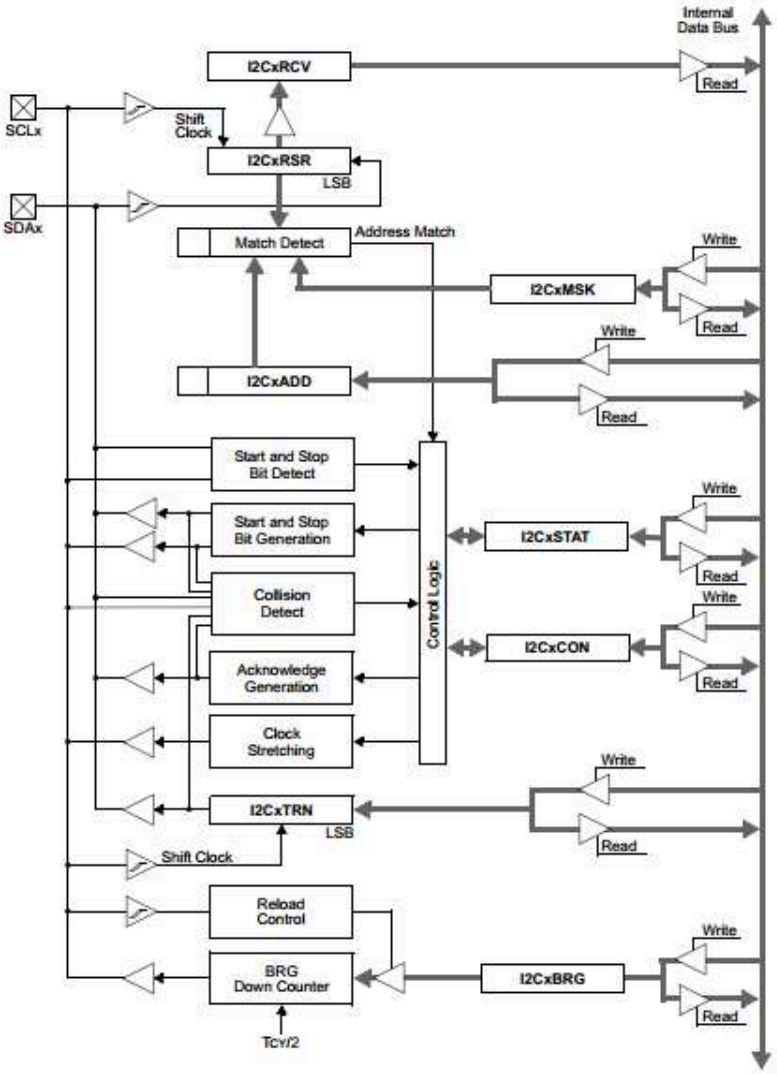

Figura 5.10: Schema a blocchi dell'interfaccia I2C del dsPIC33FJ128MC802

La figura (Figura 5.10) mette in mostra la modalità di funzionamento del protocollo dal punto di vista dei registri e delle procedure che sono usati nella routine di istruzioni del software.

5.2.2 Dispositivo UART

Acronimo per Universal Asynchronous Receiver Transmitter (ricevitore-trasmettitore asincrono universale), è un dispositivo hardware ad uso dedicato o anche ad uso generale. Il suo scopo è convertire flussi di bit di dati da un formato parallelo ad un formato seriale asincrono o viceversa.
È un modulo hardware molto diffuso e infatti, ormai, ogni famiglia di microcontrollori lo implementa consentendo ai progettisti il collegamento di dispositivi di tipologie diverse come Bluetooth, sensori di vario genere, display LCD e altro ancora.
Un UART in pratica si interpone tra il bus di I/O e la periferica che ha la necessità di comunicare con il processore. Mediante l'ausilio di due canali, byte di dati verranno trasmessi in modo seriale, cioè un bit per volta, dal processore al dispositivo e viceversa.
Utilizzando un canale seriale il dispositivo UART riceverà dalla periferica un certo numero di bit (uno alla volta). Tali bit verranno assemblati e convertiti in byte e quindi a questo punto sarà possibile effettuare una transazione sul bus tra il dispositivo e il processore.
I registri più importanti sono denominati Rx e Tx, collegati a livello hardware due pin del dispositivo, a cui si connettono due cavi. Rx e Tx sono entrambi *shift register* (registri a scorrimento) da 8 bit. Rx riceve dati seriali in ingresso e li trasferisce in un pacchetto per vie parallele; Tx riceve dati in parallelo e provvede alla trasmissione in modo seriale verso il dispositivo; il converso viene effettuato per la trasmissione seriale alla periferica.
Sono presenti nei Personal Computer sotto il nome di USART (Universal Synchronous-Asynchronous Receiver/Transmitter) con il compito specifico di gestire le comunicazioni delle interfacce seriali RS-232. Gli USART costituiscono l'evoluzione degli UART in quanto aggiungono l'opportunità (come si evince dal nome) di effettuare comunicazioni seriali sincrone.

5.3 Calcolo del Baud Rate

Il Baud rappresenta il numero di dati che viene trasmesso in un secondo attraverso un sistema di trasmissione digitale ed è l'unità di misura del Baud Rate (o symbol rate).
Da non confondere con l'unità "bit-al-secondo" (bit/s o bps) che differisce perché ad un simbolo possono corrispondere più bit se si usa una modulazione non binaria (di ampiezza, frequenza o fase) e di conseguenza la velocità espressa in bps potrebbe risultare multipla rispetto a quella espressa in baud.

Per il corretto funzionamento dell'intero sistema, il calcolo del Baud rate è imprescindibile. Di seguito sono riportati i calcoli relativi alle velocità usate dall'oscillometro riadattate per il modulo UART e la comunicazione mediante I2C.

5.3.1 Frequenza dell'oscillatore

Frequenza nominale in ingresso (F_{IN}) = 7,37 MHz

$$F_{OSC} = F_{IN} * \frac{M}{N_1 * N_2} = 7,37 * \frac{43}{2*2} = 79,22MHz$$

$$F_{CY} = \frac{F_{OSC}}{2} = 39.61375MHz$$

Facendo riferimento alle formule impostate, il codice utile a tale configurazione è il seguente:

```
//configurazione oscillatore.
//FOSC = Fin * M/(N1*N2) = 7.37 *43/(2*2) =  79.22 Mhz
//FRC divided by 1 (default)
CLKDIVbits.FRCDIV  = 0b000;
//N1 =2  Input divided by 2 (default)
CLKDIVbits.PLLPRE  = 0b00000;
//N2 =2     Output divided by 2
```

```
CLKDIVbits.PLLPOST = 0b00;
//PLLDIV = 41;   M = PLLDIV + 2 = 43
PLLFBDbits.PLLDIV  = 0b000101001;
```

dove le variabili N1, N2 e M sono opportunamente inizializzate attraverso gli appositi registri, così come la funzione PLL.

5.4 Pulse-Width Modulation (PWM)

Il segnale PWM (modulazione a larghezza di impulsi) è un onda quadra di duty cycle variabile che permette di controllare l'assorbimento di un carico elettrico. In questo progetto, si ci riferisce al carico elettrico assorbito dai servomotori che si ottiene modulando opportunamente la variabile duty cycle. Tale variabile rappresenta il rapporto tra il tempo in cui l'onda assume valore alto e il periodo T (l'inverso della frequenza: T=1/f).
Per esempio un duty cycle dell'60 corrisponde ad un'onda quadra che assume valore alto per il 60% del tempo e valore basso per il 40% rimanente. Se si considera il duty cycle come un interruttore che indica la percentuale in cui un LED rimane acceso/spento, si potrebbe dire che ai valori:

- 0% , il LED risulta essere sempre spento;
- 100%, lo stesso LED risulta essere sempre acceso.

Utilizzando dispositivi PIC (come in questo progetto), il PWM potrebbe essere generato:

- mediante emulazione software;
- oppure tramite hardware.

Nella generazione per vie software la frequenza e il duty cyle sono assegnati ad un certo piedino in uscita che provvede alla generazione dell'onda. Mentre, per la generazione dell'onda con metodo hardware si utilizza il modulo Capture Compare PWM (CCP).

Capitolo 6

Collaudo

> «*Il mio software non ha mai bugs. Include soltanto funzionalità casuali*». (cit)

Man mano che veniva scritto il codice con le configurazioni del sistema e delle componenti, queste venivano collaudate per capire dove si verificavano eventuali interruzioni e incongruenze e le motivazioni che portavano ad averle.
Il primo passo è stato fatto sulla configurazione dell'oscillatore. Per verificare che le impostazioni fossero adeguate al corretto funzionamento del sistema si è provveduto ad aggiungere un comando di prova inerente l'accensione di un pin-LED presente sulla piastra.
Come secondo passo, si è pensato di configurare il modulo UART, che, come detto nel capitolo precedente, è stato utile ad avere i primi risultati inerenti i dati forniti dall'accelerometro. Inizialmente si è provveduto a verificare che fosse in atto una comunicazione: un computer è stato collegato via Bluetooth al microcontrollore che attraverso l'interfaccia UART ha mandato in stampa una stringa di prova. Successivamente, con il medesimo metodo sono stati visualizzati su schermo i dati relativi alle variabili x, y, z e poi ulteriormente i valori di pitch e di roll.
La parte importante in termini di risultati concreti è stata apportata dalla configurazione del protocollo I2C attraverso cui si è realizzata la reale connessione tra il microcontrollore e l'accelerometro. I test hanno riguardato le procedure di cui il protocollo si compone e in particolare hanno consistito

nell'impostare punti di pausa e verifiche mediante l'accensione o lo spegnimento del LED (di cui prima) in risposta alle procedure/funzioni.
Finite le configurazioni, visti i primi risultati, è stato possibile constatare visivamente la corretta risposta dei servomotori rispetto ai dati raccolti e opportunamente rielaborati.

Capitolo 7

Conclusioni

É tempo di tirare le fila del discorso. Al termine del progetto si è ottenuta una sospensione cardanica o Gimbal (che dir si voglia) che reagisce ai movimenti della sospensione compensando le sollecitazioni sul piano orizzontale.
L'uso del microcontrollore rappresenta il cuore della soluzione, ciò che permette di attuare la soluzione al problema posto.
Il firmware contenuto all'interno della memoria permanente del microcontrollore è stato scritto in linguaggio C e viene rilasciato in allegato alla tesi secondo i principi della metodologia open hardware (Capitolo 3).
Per degli sviluppi successivi non si esclude la possibilità di realizzare una nuova soluzione al medesimo problema usando stavolta un approccio di tipo hardware: cioè una soluzione integrata nel sistema. I vantaggi che si possono ricavare da una realizzazione ti tipo hardware sono: innanzitutto una maggiore accuratezza nel calcolo degli angoli di pitch e di roll; secondariamente, una miglior velocità di attuazione dei movimenti.

Bibliografia

[1] *Codice sorgente del software*
arslab.dmi.unict.it/it/content/progetti-di-studenti

[2] *Open Source Hardware Association*
www.oshwa.org/definition/italian/

[3] *Gnu Operating System*
www.gnu.org

[4] *Ministero dello Sviluppo Economico*
www.uibm.gov.it/index.php/brevetti

[5] *Componente della famiglia dei microcontrollori Microchip*
www.microchip.com/wwwproducts/Devices.aspx?dDocName=en532302

[6] *Introduzione all'architettura dei calcolatori* 3/ed
Carl Hamacher, Zvonko Vranesic, Safwat Zaky, Naraig Manjikian
2013 - McGraw-Hill