Tecnologie Digitali – AA 2024/25

Scheda di laboratorio n.10

Introduzione ai sensori digitali

In questa esperienza concluderemo la transizione al digitale e utilizzeremo l'interfaccia digitale `I2C` di `Analog Discovery 2` per comunicare con alcuni dei sensori digitali ricevuti in dotazione, caratterizzarli ed usarli per eseguire qualche semplice esperimento. In particolare ad ogni tavolo sarà assegnato un sensore fra i seguenti: il barometro `LPS331AP`, l'igrometro `SHT3x` e il sensore inerziale `MPU6050`. Sebbene ricadano tutti entro gli standard `I2C`, ognuno ha le sue particolarità nella gestione della comunicazione, come indicato nel *datasheet*. Il livello di complessità del processo di comunicazione è comunque comparabile. Ad ogni tavolo sarà assegnato un sensore, inoltre verrà fornito un codice iniziale in grado di comunicare e leggere la temperatura: tutti i sensori infatti integrano un termometro, tipicamente per motivi di compensazione. La presentazione orale sarà una occasione per condividere fra i tavoli le informazioni su tutti i sensori.

1 Operazioni di base: configurare, leggere e scrivere una linea digitale

Come molti dispositivi digitali e `MCU`, `Analog Discovery 2` possiede diversi canali digitali che sono completamente riconfigurabili come canali di `input`, `output` o `clock`, in modo da mantenere una grande flessibilità nelle possibilità di collegare lo strumento ad interfacce digitali. Come esempio illustrativo, utilizzeremo `WaveForms`per visualizzare un semplice segnale costituito da due livelli discreti. Il "salto" da un livello all'altro avviene in modo casuale.

Task 1 Dopo aver cortocircuitato due dei canali digitali di `Analog Discovery 2` *a piacere*, utilizzare la modalità `Patterns` per creare un output digitate di tipo `Random` e con frequenza 50 Hz. Utilizzare la modalità `Logic` definendo un `bus` per il canale di input. Verificare la corretta conversioni dei livelli *alto* e *basso* nei valori binari **1** e **0**.

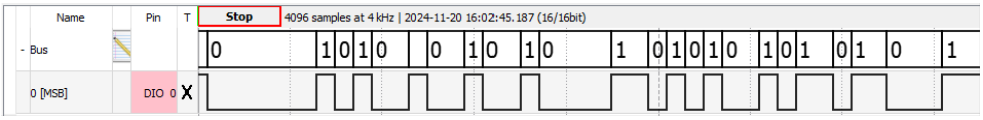

Figura 1.1: *Analisi di un segnale digitale utilizzando il modulo `Logica` di `WaveForms`.*

Mentre `WaveForms` può essere utile per visualizzare qualitativamente segnali digitali come in questo semplice esempio, per interfacciarci seriamente con i sensori ed effettuare piccoli esperimenti utilizzeremo script `python` dedicati e basati sul nostro modulo `tdwf`. Come visto a lezione, per cominciare la comunicazione con protocollo `I2C`, il Master manda un segnale di `Start` seguito dall'indirizzo dello Slave con cui si intende comunicare, `SDA - Slave ADdress`. Il `SAD` è di solito *hard-coded* all'interno del dispositivo e rappresentato con un numero a 7 bit (a cui aggiungeremo un bit di scrittura/lettura per comporre un intero *byte* - numero binario di 8 bit)[1]. L'indirizzo `SAD` può essere reperito nel *datasheet* del sensore, alternativamente può essere ottenuto collegando il sensore al controllore e passando in rassegna, *a forza bruta*, tutti i 126 possibili valori del `SAD`[2]. Quando viene interrogato con il giusto indirizzo, lo Slave risponderà quindi con un bit di "riconoscimento", `ACK` - dall'inglese *acknowledged*. Ricordiamo, per completezza, che nella nostra rete logica con resistenze di *pull-up*, il bit `ACK` sarà uno zero logico mentre il suo opposto, `NACK` - *not acknowledged* - sarà rappresentato da un uno logico, equivalente anche allo stato di riposo del sistema.

[1] anche se *hard-coded*, in molti dispositivi è prevista la possibilità di scegliere il valore tra quelli prsenti in una lista finita. Questa possibilità è utile qualora più elementi dello stesso tipo siano presenti nello stesso network `I2C`

[2] Non sono 128 come ci saremmo aspettati! Alcuni valori sono riservati per altre operazioni.

Dopo aver correttamente collegato il sensore ad `Analog Discovery 2`, solo dopo aver individuato i pin corretti secondo quanto indicato nella seconda parte della scheda, possiamo implementare un semplice codice di scansione e rivelazione del `SAD`:

```
import tdwf

ad2 = tdwf.AD2()

ad2.vdd = 3.3
ad2.power(True)

i2c = tdwf.I2Cbus(ad2.hdwf)

devs = i2c.scan()
for dev in devs:
    print(f"Found device @SAD: 0x{dev:02x}")

ad2.close()
```

A parte le definizioni di base che abbiamo imparato a conoscere, il codice attiva una linea di alimentazione a 3.3 V, che è compatibile con tutti i sensori che utilizziamo (verificatelo!). Questa è essenziale per impostare la logica di *pull-up* della nostra rete. Invece tra i comandi nuovi abbiamo la definizione del protocollo `I2C`, utilizzando la funzione `tdwf.I2Cbus(ad2.hdwf)`. All'interno del nuovo oggetto `I2C` abbiamo quindi accesso alla funzione `scan`: questa non fa altro che scorrere tutti i possibili valori di indirizzo e salvare tutti quelli per cui riceve risposta di riconoscimento, di nuovo rappresentata dal bit `ACK`.

Task 2 Solo dopo aver identificato i pin del sensore assegnatovi come illustrato nella seconda parte della scheda, testare il codice descritto sopra per ottenere il `SAD` del sensore. Verificare esplicitamente che il valore ottenuto è lo stesso riportato sul *datasheet*.

A questo punto siamo pronti a scendere nel dettaglio sul sensore che vi è stato assegnato. Come introdotto in precendeza studieremo un barometro, un igrometro e un accelerometro: nelle pagine successive verranno descritti in dettaglio con alcune specifiche sul funzionamento e possibili suggerimenti di esperimenti da effettuare. Per questa particolare esperienza sentitevi liberi di misurare o testare qualunque fenomeno che riteniate interessante.

2 Sensore #1 - Barometro LPS331AP

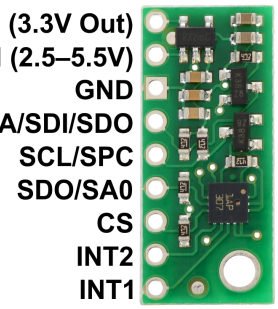

Figura 2.1: *Barometro LPS331AP.*

Il primo gruppo si focalizzerà sul sensore di pressione assoluto LPS331AP prodotto dalla STMicroelectronics e montato sul modulo riportato in Fig.2.1. Nell'esperienza useremo l'interfaccia I2C per comunicare con il sensore, per accenderlo, configurarlo e misurare le variabili di interesse. Come sarà chiaro dalla lettura del *datasheet*, il microchip non misura solo la pressione ma anche la temperatura. Il comportamento grezzo (*raw*) del sensore ha una non sorprendente dipendenza dalla temperatura che può dare luogo ad errori sistematici impossibili da rimuovere senza conoscere la temperatura. Per questo motivo, il termometro *on-chip* viene usato per compensare questi effetti; tutto questo avviene in maniera automatica a livello hardware e noi avremo solo accesso alla misura compensata.

Task 3 Partendo dal *datasheet* individuare ed annotare, per le quantità misurate dal sensore: (1) range dinamico del sensore; (2) sensibilità del sensore; (3) accuratezza del sensore; (4) precisione del sensore. Nell'ultimo caso, fare attenzione in particolare a come questi valori possano dipendere dalla configurazione del chip (velocità di acquisizione, numero di medie fatte dal chip stesso, eccetera).

2.1 Montaggio del sensore e verifiche di base

Task 4 Cercare nel *datasheet* il significato dei vari *pin* del sensore. Annotare come va alimentato e se l'alimentazione a nostra disposizione sia adatta. Individuare i *pin* di collegamento per l'interfaccia I2C e l'indirizzo SAD.

Task 5 Individuare sul datasheet la lista dei registri dell'integrato e prendere nota di quelli che sembrano essere più utili. Annotate anche le formule necessarie per convertire i valori di pressione e temperatura dai digit a delle unità fisiche.

Ogni sensore ha le sue particolarità, facciamo notare quelle meno evidenti di LPS331AP rispetto allo standard I2C, in modo che possiate procedere in maniera spedita con l'esperienza. Come potete rintracciare anche nel *datasheet* (dove? lo trovate nelle specifiche I2C ovviamente), in LPS331AP ogni registro può essere indirizzato in due modi alternativi. Per esempio, CTRL_REG1 è associato all'indirizzo SUB=0x20=32, ma è possibile raggiungerlo anche impostando SUB=0x20+0x80=0xA0=160, che differisce dal caso precedende solo per il bit più significativo (MSB = *Most Significant Bit*). Nel secondo caso, si attiva un meccanismo di **autoincremento** per si accede prima al registro 0xA0, poi 0xA1 (registro CTRL_REG2), poi 0xA2 (registro CTRL_REG3) e così via. Diversamente, se leggiamo molti *bytes* da 0x20 si otterranno solo (inutili) letture ripetute di CTRL_REG1 e non ci sarà nessun incremento nell'indirizzo fra una lettura e l'altra. Suggeriamo di focalizzare l'attenzione sul registro CTRL_REG1, che contiene un bit che *attiva* il sensore... altrimenti il dispositivo non farà praticamente quasi nulla, dato che rimane sostanzialmente spento. Un altro registro molto utile per fare pratica è chiamato WHO_AM_I e permette sostanzialmente di chiedere al chip di identificarsi. Si suggerisce di partire da questi due registri per fare pratica di comunicazione. Una volta certi di comunicare con il sensore, si può procedere a fare delle vere misure.

Task 6 Costruire uno *script* (modificando I2C_LSP3331_template che potete scaricare dal Team del corso), che legga i registri di misura, che calcoli i valori in unità fisiche e che riporti periodicamente il risultato sulla connessione seriale.

2.2 Studio del sensore e misure

Task 7 Effettuare uno studio statistico dei dati del sensore e confrontare con quanto riportato dal costruttore. Quale è la precisione/distribuzione della misura? Ci sono dei *drift*? Come approfondimento, considerare le opzioni di integrazione e *datarate* offerte dal dispositivo (non necessariamente tutte, potrebbero essere davvero molte: l'importante è farsi un'idea delle possibilità del sensore).

A questo punto, acquisita un minimo di dimestichezza con il sensore, lasceremo esplorare il suo uso per qualche misura di fisica o, alternativamente, qualche verifica più approfondita sul funzionamento del dispositivo. Riportia-

mo a seguire alcuni esempi possibili: vi lasciamo libertà, ma eseguite almeno uno degli esempi proposti (peraltro comunque abbastanza liberi).

- Come suggerito dal *datasheet*, il sensore è venduto anche come "altimetro". Data la variabilità della pressione atmosferica ovviamente non può che essere un altimetro relativo. Quanto precisamente/accuratamente può essere usato per misurare delle differenze di altezza o velocità verticali? Oppure, conoscendo le differenze di altezza fra diverse prese dati, potete dedurre qualcosa circa la fisica che determina la variazione di pressione con l'altezza?
- Il sensore integra un termometro perché evidentemente la temperatura è un parametro importante. In effetti, è importante e la sua misura serve a compensarne gli effetti sul meccanismo di trasduzione che permette al sensore di stimare la pressione. Provare a mettere sotto test la solidità della compensazione in temperatura del sensore: idealmente, la misura di pressione non dovrebbe dipendere per nulla dalla temperatura del sensore... ma è vero? quanto è vero?
- Una esperienza ovvia potrebbe consistere nel fare una misura di tipo prettamente atmosferico in funzione del tempo, per esempio loggando l'evoluzione di pressione e temperatura durante la giornata e/o al variare delle condizioni di misura nella stanza (riscaldamento acceso? spento? condizioni di insolazione, eccetera).
- Può un sensore del genere percepire lo sbalzo di pressione generato dall'apertura o chiusura, più o meno violenta, della porta di una stanza? Ha sufficiente precisione di lettura? Sufficiente risoluzione temporale? Oppure è possibile fare considerazioni simili per perturbazioni simili? Oppure ancora, risulta sensibile all'orientazione nello spazio? O alla luce? O a qualsiasi altra cosa vi venga in mente?

Homework 1 Come funziona questo sensore, esattamente? Come fa a tradurre una pressione in un segnale elettrico? Cercare qualsiasi fonte disponibile, *datasheet*, *brochure*, *application notes*, sito del produttore, eccetera che riuscite a trovare, e riassumetene gli aspetti chiave. Questo sarà in particolare richiesto a chi presenterà il sensore nella presentazione orale.

3 Sensore #2 - Igrometro SHT3x

Figura 3.1: *Igrometro SHT3x.*

Il secondo gruppo si focalizzerà sul sensore di umidità SHT3x prodotto dalla Sensirion e riportato in Fig.3.1.

È certamente chiaro a tutti che l'*umidità* quantifica in qualche modo il contenuto di vapore acqueo nell'aria, ma è utile ricordare alcuni concetti di base prima di usare questo sensore. In particolare è possibile definire una:

- *umidità assoluta,* che corrisponde al contenuto di acqua nell'aria misurato tipicamente in *grammi per metro cubo*;
- *umidità relativa,* detta anche RH (Relative Humidity), che misura il rapporto fra l'umidità misurata e quella di saturazione ad una data temperatura; in questo caso abbiamo quindi un numero adimensionale che viene tipicamente indicato con una *percentuale*.

Ricordiamo brevemente il processi fisici correlati con le quantità misurate da questo sensore. Per una data temperatura T l'equilibrio della transizione di fase liquido-vapore è definito da una pressione di vapore $P(T)$. Evaporazione e condensazione sono dei processi dinamici il cui bilanciamento dipende dalla pressione parziale (e quindi dalla densità) del vapore. Se la pressione parziale è superiore a $P(T)$ la condensazione prevale, mentre se è inferiore sarà l'evaporazione a prevalere. L'equilibrio si ottiene chiaramente ad una pressione di vapore uguale a $P(T)$, che possiamo per questo anche chiamare pressione di *vapore saturo*[3] e corrispondente a $RH = 100\%$. Quanto discusso vale chiaramente per il limite ideale di un sistema chiuso isotermo, che tenderà quindi sistematicamente a raggiungere $RH = 100\%$. Se consideriamo invece un sistema con oggetti e liquidi a diverse temperature (come nella stragrande parte delle situazioni reali) come potete facilmente immaginare la dinamica della evaporazione/condensazione si complica. Per andare un minimo sul quantitativo, nel grafico di sopra riportiamo la pressione di vapore $P(T)$ per un range di temperature per noi interessante, con anche una corrispondenza fra diversi valori di RH e diverse densità di vapore. Il grafico si ferma a 50 C ma, come noto, $P(T)$ sale fino a raggiungere una atmosfera a 100 C. Notate come un RH del 100% può corrispondere ad densità assolute di vapore *molto* diverse al variare della temperatura.

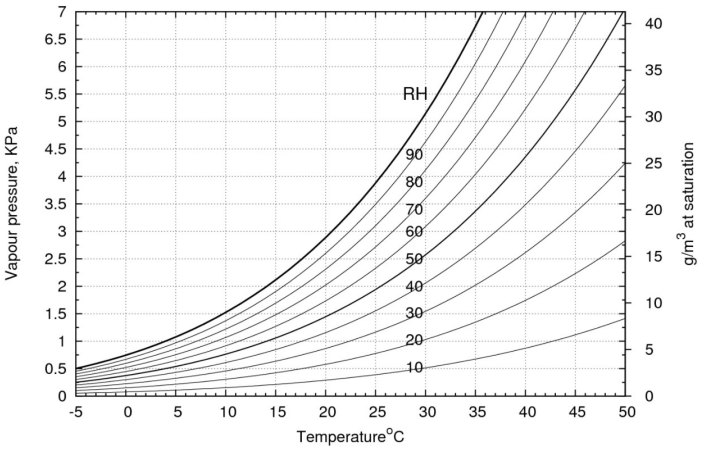

Figura 3.2: *Pressione di vapore a diversi valori di RH (umidità relativa).*

Task 8 Partendo dal *datasheet* individuare ed annotare, per le quantità misurate dal sensore: (1) range dinamico del sensore; (2) sensibilità del sensore; (3) accuratezza del sensore; (4) precisione del sensore. Nell'ultimo caso, fare attenzione in particolare a come questi valori possano dipendere dalla configurazione del chip (velocità di acquisizione, numero di medie fatte dal chip stesso, eccetera).

3.1 Montaggio del sensore e verifiche di base

Task 9 Cercare nel *datasheet* il significato dei vari *pin* del sensore. Annotare come va alimentato e se l'alimentazione a nostra disposizione sia adatta. Individuare i *pin* di collegamento per l'interfaccia I2C e l'indirizzo SAD.

Ogni sensore ha le sue particolarità e le sue "interpretazioni" del protocollo I2C. Qui sottolineiamo quelle più inattese rispetto allo standard che vi è stato illustrato, per permettervi di procedere in maniera efficiente e non farvi confondere.

In particolare, come ci si può rapidamente rendere contro iniziando a guardare il *datasheet*, non sembra esserci alcuna mappa di registri per SHT3x e la comunicazione sembra più che altro basata su dei "codici" a 16 *bit*, ossia 2

[3] Questa condizione è anche nota come *punto di rugiata* (se $T \geq 0$ C) o *punto di brina* (altrimenti), per motivi abbastanza ovvi: fissato un dato contenuto di vapore nell'aria, quando T scende sotto il punto di rugiada si ha formazione di rugiada, o brina se siamo sotto la transizione liquido-solido...

bytes. Per esempio, se mandiamo al chip due *bytes* `0x2737` (ossia mandiamo prima `0x27` e poi `0x37`) possiamo configurare il chip in un *Data Acquisition Mode* tale che il sensore effettua misure a ripetizione con una frequenza di 10 Hz. Non ci è dato sapere che succeda esattamente all'interno di questo chip, ma non sembra che questa operazione corrisponda a selezionare un registro alla locazione `0x27` per poi scriverci `0x37`, ma semmai sembra che `SHT3x` sia in grado di reagire ad un set di "comandi" binari, fra cui appunto `0x2737`. In fin dei conti, ai fini dell'operatività tutto questo è abbastanza irrilevante: l'unica cosa che conta è sapere quali codici binari è necessario mandare per ottenere le operazioni richieste.

Task 10 Individuare sul datasheet la lista dei comandi implementati dall'integrato e prendere nota di quelli che sembrano essere più utili. Il *datasheet* come vedrete include istruzioni dettagliate su come configurare il sensore e su come ottenere le misure di umidità. Annotate anche le formule necessarie per convertire i valori di pressione e temperatura dai digit a delle unità fisiche.

Task 11 Costruire uno *script* (modificando `I2C_SHT3x_template` che potete scaricare dal Team del corso), che legga i registri di misura, che calcoli i valori in unità fisiche e che riporti periodicamente il risultato sulla connessione seriale.

3.2 Studio del sensore e misure

Task 12 Effettuare uno studio statistico dei dati del sensore e confrontare con quanto riportato dal costruttore. Quale è la precisione/distribuzione della misura? Ci sono dei *drift*? Come approfondimento, considerare le opzioni di integrazione e *datarate* offerte dal dispositivo (non necessariamente tutte, potrebbero essere davvero molte: l'importante è farsi un'idea delle possibilità del sensore).

A questo punto, acquisita un minimo di dimestichezza con il sensore, lasceremo esplorare il suo uso per qualche misura di fisica o, alternativamente, qualche verifica più approfondita sul funzionamento del dispositivo. Riportiamo a seguire alcuni esempi possibili: vi lasciamo libertà, ma eseguite almeno uno degli esempi proposti (peraltro comunque abbastanza liberi). Mettiamo in guardia sul fatto che le misure di umidità non sono fra le più semplici, soprattutto per via dei tempi di risposta del sistema.

- Uno studio chiave è senza dubbio quello di determinare il tempo di riposta del sensore (lo chiariamo subito: non è istantaneo). Qui può essere utile confrontare dati reali con eventuali indicazioni nel *datasheet*.
- Un'altra possibilità consiste nel verificare il raggiungimento delle condizioni di saturazione, che ci aspettiamo siano raggiunte se il sensore è chiuso in un contenitore stagno con dell'acqua liquida. Col tempo il vapore saturerà lo spazio a disposizione portando a $RH = 100\%$. Facciamo notare che la densità di vapore ad un dato RH dipende fortemente dalla temperatura, quindi una condizione importante qui è che tutto sia isotermo.
- Come variante dell'idea precedente, è noto che la tensione di vapore dipende dal tipo di sostanze che sono disciolte nell'acqua. Per esempio una soluzione satura di NaCl avrà una pressione di vapore diversa da quella di una soluzione di acqua pura, e in teoria prevedibile.
- Può essere interessante verificare quanto il sensore sia in grado di rilevare l'umidità di diverse sorgenti: i dintorni della vostra pelle, sono più umidi dell'ambiente circostante? Cauteliamo sul fatto che è obiettivamente piuttosto difficile ottenere misure quantitative affidabili e in condizioni controllate e la correlazione con la temperatura (se per esempio toccate `SHT3x`) può dare risultati non sempre intuitivi.
- Come nel caso della pressione, un *datalogging* di lungo corso delle variabili ambientali può essere interessante da implementare e da correlare con eventi contingenti vari nella stanza.

Homework 2 Come funziona questo sensore, esattamente? Come fa a tradurre una umidità in un segnale elettrico? Cercare qualsiasi fonte disponibile, *datasheet*, *brochure*, *application notes*, sito del produttore, eccetera che riuscite a trovare, e riassumetene gli aspetti chiave. Questo sarà in particolare richiesto a chi presenterà il sensore nella presentazione orale.

4 Sensore #3 - Sensore inerziale MPU6050

Figura 4.1: *Sensore inerziale a sei assi MPU6050.*

Il terzo gruppo si focalizzerà sul sensore inerziale MPU6050 prodotto dalla InvenSense e montato sul modulo riportato in Fig.4.1. Il dispositivo è un sensore monolitico inerziale a sei assi, nel senso che è in grado di fare una misura vettoriale dell'accelerazione e della velocità angolare. Diversamente dai due sensori precedenti, MPU6050 ha un comportamento completamente standard da un punto di vista dell'interfaccia I2C, tuttavia è (potenzialmente) più complesso per quanto riguarda la gestione dei dati. Tipicamente, infatti le misure di questo tipo di sensori vengono combinate con quelle provenienti da un GPS o magari anche da una bussola per ricostruire la configurazione nello spazio 3D di un dispositivo. Questa procedura generica va sotto il nome di *sensor fusion*, ed è in grado di combinare una misura assoluta ma magari non precisissima (come quella del GPS) con una più precisa ma affetta da forti *drift* e *offset* (come quella di MPU6050). Qui suggeriremo al più di integrare i dati di uno/due assi, ma ovviamente siete liberi di esplorare questi concetti più avanzati.

Task 13 Partendo dal *datasheet* individuare ed annotare, per le quantità misurate dal sensore: (1) range dinamico del sensore; (2) sensibilità del sensore; (3) accuratezza del sensore; (4) precisione del sensore. Nell'ultimo caso, fare attenzione in particolare a come questi valori possano dipendere dalla configurazione del chip (velocità di acquisizione, numero di medie fatte dal chip stesso, eccetera).

4.1 Montaggio del sensore e verifiche di base

Task 14 Cercare nel *datasheet* il significato dei vari *pin* del sensore. Annotare come va alimentato e se l'alimentazione a nostra disposizione sia adatta. Individuare i *pin* di collegamento per l'interfaccia I2C e l'indirizzo SAD.

Ogni sensore ha le sue particolarità, MPU6050 implementa in modo relativamente standard il protocollo I2C ma ha una mappa dei registri piuttosto articolata e anche delle funzionalità meno ovvie come il buffer FIFO (*First In First Out*) per le misure veloci. Facciamo notare però che *non è assolutamente necessario* passare per il registro FIFO per fare le misure di accelerazione e velocità angolare: basta leggere i registri corrispondenti dedicati ai tre assi di accelerazione, alla temperatura del sensore, ai tre assi di rotazione. Ognuna di queste misure ha vari registri di configurazione che controllano medie, fondoscala e quant'altro, ma è sempre possibile usare le impostazioni di *default* (a patto che siano note... e tutto è scritto nel *datasheet*). Se interessati (è opzionale) suggeriamo di dedicarsi al FIFO in un secondo momento. Come ulteriore suggerimento non facilmente rintracciabile nei *datasheet*, facciamo notare che dopo l'accensione il chip si trova di *default* in uno stato di *stand-by* e la lettura di qualsiasi registro di dati fornirà solo una lunga serie di 0x00. Per attivare veramente il chip è necessario scrivere qualcosa nel registro PWR_MGMT_1, scegliendo la configurazione desiderata.

Task 15 Individuare sul datasheet la lista dei comandi implementati dall'integrato e prendere nota di quelli che sembrano essere più utili. Il *datasheet* come vedrete include istruzioni dettagliate su come configurare il sensore e su come ottenere le misure di temperatura, accelerazione e rotazione lungo i 3 assi. Annotate anche le formule necessarie per convertire i valori forniti a delle unità fisiche.

Task 16 Costruire uno *script* (modificando I2C_MPU6050_template che potete scaricare dal Team del corso), che legga i registri di misura, che calcoli i valori in unità fisiche e che riporti periodicamente il risultato sulla connessione seriale.

4.2 Studio del sensore e misure

Task 17 Effettuare uno studio statistico dei dati del sensore e confrontare con quanto riportato dal costruttore. Quale è la precisione/distribuzione della misura? Ci sono dei *drift*? Come approfondimento, considerare le opzioni di integrazione e *datarate* offerte dal dispositivo (non necessariamente tutte, potrebbero essere davvero molte: l'importante è farsi un'idea delle possibilità del sensore).

A questo punto, acquisita un minimo di dimestichezza con il sensore, lasceremo esplorare il suo uso per qualche misura di fisica o, alternativamente, qualche verifica più approfondita sul funzionamento del dispositivo. Riportiamo a seguire alcuni esempi possibili: vi lasciamo libertà, ma eseguite almeno uno degli esempi proposti (peraltro comunque abbastanza liberi).

- Un esperimento piuttosto ovvio consiste nel realizzare un rivelatore di *tilt*, guardando l'orientazione del vettore di accelerazione gravitazionale. Come sempre, con questo sensore è fondamentale essere ben coscienti della presenza di *offset* in qualsiasi misura.
- Anche senza andare nel limite della fusione dei (dati dei) sensori, è abbastanza facile fare delle misure di rotazione su un singolo asse (suggerito quello perpendicolare alla breadboard) in funzione del tempo. Dato che il sensore è affetto da un *offset* non trascurabile, una banale integrazione numerica della velocità angolare verticale sarà certamente afflitta da un grosso *drift*. Tuttavia, se l'esperimento viene fatto cominciare con una misura di *nulling* a sensore fermo (ossia fittando i dati per trovare l'*offset* da rimuovere), è possibile fare delle misure relativamente accurate anche sulla scala di decine di secondi. A questo punto in teoria ci aspetteremmo solo una dinamica di *diffusione* data dall'integrazione di un dato rumoroso. Sfortunatamente l'*offset*, dipendendo da vari parametri quali per esempio la temperatura esatta, non è nemmeno stabile e tende tipicamente a driftare nel tempo, quindi dopo un poco certamente l'accuratezza delle stima peggiorerà fino a rendere la stima dell'orientazione angolare completamente inaffidabile.
- L'esperimento precedente è in teoria replicabile per le coordinate x e y (l'inclusione di z presuppone verosimilmente che `Nucleo144` non sia su un piano, il che richiede una combinazione affatto banale dell'*input* di tutti i sei assi dell'unità inerziale). Nella pratica, è necessario fare una doppia integrazione dell'accelerazione, e la procedura è ancora più suscettibile alla presenza di *offset* e *drift*.
- Non è banale da realizzare dato l'ingombro di `Nucleo144` e la necessità di usare dei cavi di collegamento, ma in linea di principio ovviamente l'accelerometro può essere utile a misurare la dinamica di un oscillatore armonico smorzato, per esempio un pendolo. Nel caso vi ispiraste a questo esempio, le rotazioni non saranno un effetto gradito quindi suggeriamo di costruire un pendolo quadrifilare. In generale, gli accelerometri come `MPU6050` son particolarmente adatti a misurare accelerazioni/rotazioni ripetute, come per esempio nei misuratori di passi.
- Alcune fonti[4] sostengono che il giroscopio di questo sensore sia in grado di percepire se la Terra sia in rotazione o meno: possibile? Lasciamo a voi giudicare ma anticipiamo che in ogni caso la misura è davvero al limite di quello che è possibile fare con un sensore come questo.

Homework 3 Come funziona questo sensore, esattamente? Come fa a tradurre accelerazioni e rotazioni in un segnali elettrici? Cercare qualsiasi fonte disponibile, *datasheet*, *brochure*, *application notes*, sito del produttore, eccetera che riuscite a trovare, e riassumetene gli aspetti chiave. Questo sarà in particolare richiesto a chi presenterà il sensore nella presentazione orale.

[4] Per esempio https://stkwans.blogspot.com/2012/11/detection-of-rotation-of-earth.html