

ALMA MATER STUDIORUM · UNIVERSITÀ DI BOLOGNA

---

---

SCUOLA DI SCIENZE  
Informatica (Laurea Magistrale)

PIATTAFORMA HARDWARE A BASSO COSTO  
A SUPPORTO DELLO SVILUPPO DI SOFTWARE PER  
LA MISSIONE ESA-EUCLID

Relatore:  
Chiar.mo Prof.  
**RENZO DAVOLI**

Presentata da:  
**FABIO PROIETTI**

Correlatore:  
Dott.  
**FRANCESCO GIACOMINI**

I Sessione  
2016/2017



*A coloro che hanno come passione  
la curiosità . . .*



# Indice

<b>Introduzione</b>	<b>ix</b>
<b>1 La missione Euclid dell'ESA</b>	<b>1</b>
1.1 Scopo della missione . . . . .	1
1.2 Studi riconducibili alla missione . . . . .	3
1.3 Composizione dello spacecraft . . . . .	5
1.4 Visual Instrument . . . . .	6
1.5 Near Infrared Spectrometer and Photometer . . . . .	7
1.5.1 Optical Mechanical Assembly . . . . .	8
1.5.2 Detection System . . . . .	9
1.5.3 Warm Electronics . . . . .	9
1.6 MIL-STD 1553 . . . . .	11
1.7 Analisi delle componenti della NI-ICU . . . . .	13
1.8 RTEMS . . . . .	15
1.9 NI-ICU Application Software . . . . .	17
1.10 La roadmap di Euclid . . . . .	20
<b>2 Progettazione e sviluppo del dimostratore</b>	<b>23</b>
2.1 Ciclo di vita di uno spacecraft . . . . .	23
2.2 Scopo della tesi . . . . .	25
2.3 NI-CDPU . . . . .	26
2.4 SecoiaSim . . . . .	27
2.5 Requisiti del dimostratore . . . . .	29
2.6 Scelte hardware . . . . .	30

2.6.1	Raspberry Pi . . . . .	30
2.6.2	Arduino Mega . . . . .	31
2.6.3	Stepper motor . . . . .	31
2.6.4	Celle foto-conduttrive . . . . .	33
2.6.5	Rilevatore di temperatura . . . . .	34
2.6.6	LED . . . . .	34
2.7	Protocollo SPI . . . . .	36
2.8	Voltage divider . . . . .	38
2.9	Pulse Width Modulation . . . . .	38
2.9.1	Implementazione di SPI in Raspberry Pi . . . . .	39
2.9.2	Implementazione di SPI in Arduino Mega . . . . .	40
2.10	Architettura hardware del dimostratore . . . . .	42
2.11	Raspberry Pi: specifiche software . . . . .	43
2.11.1	Controllo della Calibration Unit . . . . .	43
2.11.2	Gestione dei Riscaldatori . . . . .	44
2.11.3	Gestione dei motori passo-passo . . . . .	44
2.11.4	Calcolo delle telemetrie . . . . .	47
2.12	ATMEGA2560: Specifiche software . . . . .	48
2.12.1	Controllo della Calibration Unit . . . . .	49
2.12.2	Gestione dei riscaldatori . . . . .	50
2.12.3	Gestione dei motori passo-passo . . . . .	50
2.12.4	Gestione delle telemetrie . . . . .	51
2.13	Implementazione del protocollo SPI . . . . .	52
2.13.1	Flusso di comunicazione dei messaggi . . . . .	54
2.13.2	Modalità di spedizione dei messaggi . . . . .	54
<b>3</b>	<b>Analisi del dimostratore</b>	<b>59</b>
3.1	Vincoli Legali . . . . .	59
3.2	Obiettivi preposti . . . . .	60
3.3	Analisi delle specifiche . . . . .	61
3.4	Obiettivi raggiunti . . . . .	62
3.5	AstroPi . . . . .	64

3.6 Sviluppi futuri . . . . .	65
<b>Conclusioni</b>	<b>67</b>
<b>Bibliografia</b>	<b>69</b>



# Elenco delle figure

1.1	Lo Spacecraft e il suo posizionamento nello spazio . . . . .	3
1.2	Modello architetturale delle componenti presenti nella NI-ICU	12
1.3	Protocollo MIL-STD 1553 . . . . .	13
1.4	Architettura di RTEMS . . . . .	15
1.5	Livelli dell'Application Software della NI-ICU . . . . .	17
1.6	Diagramma delle classi dell'Application Software della NI-ICU	18
2.1	Ciclo di vita di uno spacecraft . . . . .	24
2.2	Diagramma a blocchi di SecoiaSim . . . . .	27
2.3	Stepper motor 28BYJ-48 . . . . .	32
2.4	Fotoresistore GL5528 . . . . .	33
2.5	Sensore di temperatura DHT11 . . . . .	34
2.6	Unità di Calibrazione . . . . .	35
2.7	LED che simulano il comportamento dei riscaldatori . . . . .	36
2.8	Schema elettrico del <i>resistor divider</i> . . . . .	38
2.9	Esempio di Pulse Width Modulation . . . . .	39
2.10	Schema SPI presente nel chip BCM2835 . . . . .	40
2.11	Schema SPI presente nel chip ATMEGA2560 . . . . .	41
2.12	Panoramica del dimostratore implementato . . . . .	42
2.13	Sequence diagram del protocollo SPI implementato . . . . .	55
3.1	Dispositivo AstroPi nella ISS . . . . .	64



# Elenco delle tavole

2.1	Sequenza di switch del motore 28BYJ-48 . . . . .	32
2.2	Modalità SPI per l'invio di dati . . . . .	37



# Introduzione

Negli ultimi anni molti dei progressi fatti nel mondo scientifico sono stati raggiunti grazie anche al forte sviluppo che la sfera informatica ha subìto, offrendo dei mezzi che fossero in grado di confermare o meno le teorie precedentemente elaborate.

Questo fenomeno trova fondamento anche per quel che riguarda la ricerca in ambito spaziale in cui la tecnologia, soprattutto quella embedded, sta sempre di più giocando un ruolo di vitale importanza, tantoché subito dopo la rivelazione della scoperta di Proxima Centauri b, l'esopianeta che ruota nel sistema solare più vicino rispetto al Sole, molta rilevanza mediatica è stata data su come potrebbe essere possibile raggiungerlo. La missione Breakthrough Starshot<sup>1</sup>, annunciata nel 2016 da Yuri Milner e Stephen Hawking, con lancio stimato nel 2036, prevede lo sviluppo di un nano-spacecraft di pochi grammi di peso e qualche centimetro di grandezza in grado di raggiungere il 20% circa della velocità della luce.

Oltre a questa, che attualmente è una delle maggiori che si stanno portando avanti, il connubio tra astrofisica e informatica è forte anche in tutti gli altri progetti attualmente in sviluppo. Tra questi vi è Euclid, una missione patrocinata dall'Agenzia Spaziale Europea (ESA), che si pone come obiettivo primario quello di ampliare l'attuale conoscenza rispetto all'espansione dell'universo e quindi scoprire la natura sia della materia che dell'energia oscura. Questa missione prevede il lancio entro il 2020 di uno spacecraft dotato di un telescopio che, attraverso le rilevazioni delle Baryonic Acoustic

---

<sup>1</sup>[https://en.wikipedia.org/wiki/Breakthrough\\_Initiatives](https://en.wikipedia.org/wiki/Breakthrough_Initiatives)

Oscillations (BAO) e del Weak gravitational lensing, è in grado di raccogliere informazioni utili allo studio dei problemi precedentemente citati.

Rispetto alle procedure di sviluppo di una missione, Euclid attualmente si trova nella Fase C, la fase in cui avviene l'implementazione ed il test del codice sorgente, su componenti con caratteristiche simili a quelle previste dalle specifiche dello spacecraft, ma che non sono necessariamente adatte al funzionamento nello spazio. L'implementazione della componente utile alla rilevazione del Weak gravitational lensing prevede l'utilizzo di una unità formata da due dispositivi principali riprodotti con due FPGA, una che contiene il processore su cui gira un sistema operativo RTOS e l'Application Software, l'altra che invece implementa i driver utili alla gestione di diversi sensori ed attuatori; le due board comunicano grazie al protocollo SPI.

Lo scopo principale di questa tesi è quindi quello di studiare se può essere possibile implementare un prototipo che sia in grado di emulare parte del comportamento delle due board utilizzando dispositivi hardware open-source e a basso costo. Questo potrebbe dare la possibilità agli sviluppatori di agevolare lo sviluppo e il test dell'Application Software dato che, per via dell'enorme costo, il numero delle unità previste non è tale da renderne agevole l'accesso da parte di più utenti contemporaneamente.

Questo elaborato è diviso in tre parti principali: nel primo capitolo si esplorerà l'aspetto più teorico della missione definendo quali sono gli obiettivi che si è prefissati di raggiungere, entrando anche più nel dettaglio delle componenti hardware e software che formano lo spacecraft. Dato che questa tesi si concentra sul modulo NISP, si parlerà quindi della componente NI-ICU, delle unità che lo compongono e di come l'Application Software ne regola il funzionamento. Nel secondo capitolo invece, avendo analizzato la questione, si cercherà di comprendere se il setup composto da un Raspberry Pi e un ATMEGA2560, a cui sono connessi vari sensori ed attuatori sia adatto ad integrare il ciclo di vita della missione e, nel caso, come si potrebbe interfacciare il sistema con una parte dell'Application Software originale. Infine nell'ultimo capitolo si analizza il lavoro descritto per capire se una soluzione

formata principalmente di hardware open-source è sufficiente per poter integrare quello che è l'attuale ciclo di vita dello sviluppo di una missione, o in caso contrario quali potrebbero essere i piani di sviluppo successivi che si potrebbero intraprendere.



# Capitolo 1

## La missione Euclid dell'ESA

In questo primo capitolo verranno discussi gli obiettivi della missione ESA-Euclid, le tecniche utili agli scopi della missione e gli strumenti utilizzati. Prima di porre l'attenzione sulla componente NISP, verrà fatto un rapido excursus sulla composizione generale dello spacecraft e dell'altra componente VIS. Sarà definito nel dettaglio il Near Infrared Instrument definendo dapprima il suo utilizzo per poi passare all'analisi delle sue principali componenti elettroniche. Infine, prima di parlare della roadmap prevista prima del lancio, sono analizzati l'Application Software e il sistema operativo su cui esso si basa.

### 1.1 Scopo della missione

Lo scopo fondamentale della missione Euclid è quello di capire come l'accelerazione cosmica modifica l'espansione dell'universo [1]. Per questo lo spacecraft misurerà le forme di miliardi di galassie e il *gravitational redshift* di altre decine di milioni attraverso una tecnica chiamata *Weak gravitational lensing*.

Per lensing gravitazionale si intende l'effetto che si crea quando una massa di materia si frappone tra l'osservatore e la fonte di luce sullo sfondo. In linea con la teoria della relatività di Einstein, questa distribuzione di massa

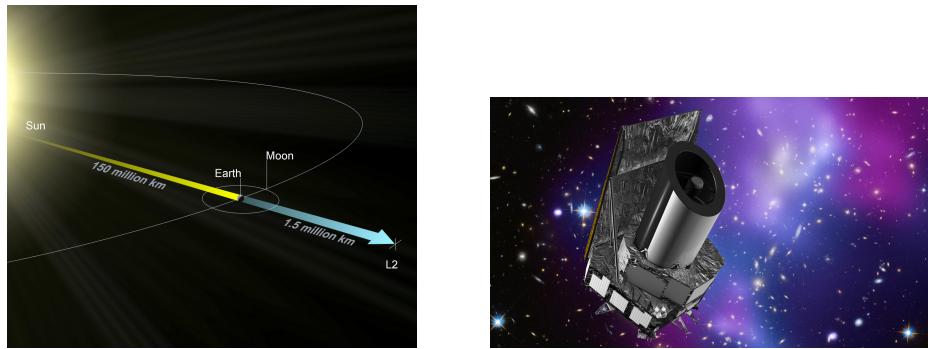
produce una distorsione spazio-tempo del raggio di luce emesso dalla componente sullo sfondo. Nel caso in cui la percentuale di curvatura del raggio è talmente piccola da non essere rivelata, si utilizza una tecnica chiamata *Weak gravitational lensing* che è in grado di calcolare la distorsione in modo statistico basandosi sull'orientamento e la forma di altre sorgenti poste sempre sullo sfondo. Se la radiazione elettromagnetica prodotta dalla fonte posta in un campo gravitazionale viene ridotta di frequenza, significa che la massa che produce l'effetto "lente" ha un campo gravitazionale con un potenziale maggiore rispetto all'oggetto sullo sfondo; in questo caso si manifesta il *gravitational time dilation*<sup>1</sup> che produce, nel caso in cui la lunghezza d'onda è maggiore, l'effetto di redshift gravitazionale.

La teoria che trova maggior fondamento per quanto riguarda la nascita dell'universo è quella del Big Bang, la quale stabilisce che l'attuale cosmologia deriva da una iniziale massa omogenea che si è rilassata nel tempo. Anche se l'effettiva forma iniziale dell'universo e la vera natura della gravità devono ancora essere confermate, diversi modelli teorici hanno ipotizzato che il meccanismo di crescita ed espansione dell'Universo è regolato da altri due elementi la cui natura è ancora sconosciuta. Circa il 76% della densità di energia viene definita *energia oscura*, mentre un altro 20% della densità dell'Universo, che è in grado di esercitare attrazione gravitazionale come normale materia ma senza emettere o assorbire luce è chiamata *materia oscura*.

Per dare risposte a questi problemi, Euclid effettuerà una mappatura in larga scala delle strutture cosmiche dell'Universo comprese in un periodo di circa 10 miliardi di anni. Con il lensing gravitazionale è possibile mappare sia la materia che l'energia oscura quantificando le immagini delle apparenti distorsioni delle galassie. Inoltre, attraverso l'uso delle *Baryonic Acoustic Oscillations*, una tecnica che permette di calcolare la regolare e periodica fluttuazione della materia normale, è possibile definire un misuratore standard

---

<sup>1</sup>E' la differenza di tempo che si ottiene nel momento in cui un osservatore effettua una misurazione del tempo in due eventi posti a distanza variabile rispetto alla stessa fonte di potenziale gravitazionale.



(a) [http://sci.esa.int/science-e-media/img/61/L2\\_1280w.jpg](http://sci.esa.int/science-e-media/img/61/L2_1280w.jpg) (accesso il 29/06/2017) (b) [http://sci.esa.int/science-e-media/img/bf/Euclid\\_spacecraft\\_illustration\\_1\\_1280.jpg](http://sci.esa.int/science-e-media/img/bf/Euclid_spacecraft_illustration_1_1280.jpg) (accesso il 29/06/2017)

Figura 1.1: Lo Spacecraft e il suo posizionamento nello spazio

utile a calcolare l’espansione dell’Universo utilizzando le variazioni delle distanze tra le galassie. Le rilevazioni sia della materia che dell’energia oscura, verranno effettuate attraverso il lancio di uno spacecraft (Fig. 1.1b) in quella che viene chiamata *L2 Lagrangian Zone* (Fig. 1.1a), ossia una porzione di spazio a circa 1,5 milioni dalla Terra. I punti *langrangiani* sono posizioni in una configurazione orbitale di due corpi dove un oggetto con massa minore ai precedenti due che è affetto solamente dalla gravità, può mantenere una posizione stabile rispetto ai due corpi. In particolare il punto L2, come anche L1 e L3, è posto in linea con la terra ed il sole, mentre L4 ed L5 sono posti in modo che essi siano due vertici di due triangoli equilateri che si formano rispettivamente con la terra ed il sole.

## 1.2 Studi riconducibili alla missione

La ricerca di esopianeti, ossia pianeti con caratteristiche simili a quelle della Terra, è un altro argomento su cui si sono concentrate molte missioni

spaziali; con i circa 4000 esopianeti candidati scoperti dal 2009, *Kepler*<sup>2</sup>, che basa la sua ricerca tramite osservazioni dallo spazio utilizzando la tecnica definita per “transito”, è la missione che ha riscosso il maggior successo nella ricerca di esopianeti [2]. I primi risultati mostrano che le caratteristiche dei pianeti con massa grande al più 1.3 volte rispetto a quella quella terra (*low-mass*) hanno caratteristiche comuni e che circa il 20% dei sistemi stellari hanno più di un pianeta di questa dimensione che gli ruota intorno.

Per quel che riguarda le osservazioni terrestri, i risultati di 8 anni di studi della missione HARPS (High Accuracy Radial Velocity Planet Searcher) mostrano che molti degli *host* di stelle simili al Sole hanno pianeti con periodi orbitali minori di 100 giorni. HARPS ha inoltre scoperto che le nane rosse con dimensione compresa tra 0.25 e 0.95 volte quella del sole comprendono un numero notevole di pianeti *low-mass* considerati abitabili. In questo caso la classificazione degli esopianeti avviene in base alla loro distanza dalla stella: sono definiti freddi tutti quei corpi che distano più di una unità astronomica<sup>3</sup> dalla stella di riferimento; essi sono molto importanti perché, data la loro lontana posizione da fonti che ne potrebbero alterare il comportamento in modo consistente, permettono la possibilità di provare teorie sulla loro formazione. Se la distanza invece è minore di 1 au, essi vengono definiti caldi ed i metodi fino ad ora utilizzati per la scoperta di nuovi esopianeti sono principalmente due: per transito fotometrico e calcolando la velocità radiale. Il transito fotometrico è in grado di stabilire il raggio di un pianeta nel momento in cui esso si frappone tra la stella cui appartiene e l'osservatore, mentre la velocità radiale permette di calcolare la massa del pianeta in base all'influenza che esso ha sul moto della stella stessa; grazie al colore che la luce assume rispetto allo spettro (rossa se si allonta dall'osservatore, blu se si avvicina), è possibile calcolare la frequenza di spostamento della stella e quindi la sua velocità radiale nei confronti di un punto centrale definito dall'osservatore. Anche se questi due metodi “tradizionali” sono largamente

---

<sup>2</sup>[https://www.nasa.gov/mission\\_pages/kepler/main/index.html](https://www.nasa.gov/mission_pages/kepler/main/index.html) (accesso: 26/06/2017)

<sup>3</sup>1 au = circa 150M di Km (ossia la distanza terra-sole)

usati, essi hanno permesso la scoperta di solo 8 esopianeti con massa inferiore a 30 volte quella della Terra (indicata con  $M_{\oplus}$ ) che si trovano le intervallo della zona abitabile.

Le precedenti tecniche non danno sufficienti garanzie per la scoperta di questi tipi di pianeti; grazie al lensing gravitazionale invece si è in grado di rilevare un picco sensibile di *microlensing* nella cosiddetta “snow-line”<sup>4</sup>, aumentando quindi di molto la possibilità che quel corpo sia un esopianeta. A differenza delle altre tecniche è inoltre possibile rilevare esopianeti che non appartengono a nessun sistema stellare o quelli considerati caldi e con massa anche molto minore di 30  $M_{\oplus}$ . Oltre a determinare i meccanismi di origine e accelerazione di espansione dell’Universo, in modo complementare a *Kepler*, *Euclid* può avere anche lo scopo di effettuare una statistica su tutti quegli esopianeti che hanno dimensioni minori rispetto a quella di Marte e che si trovano sia nella zona libera (ossia a pianeti che non appartengono a nessun sistema stellare) che nella più famosa “zona abitabile”.

### 1.3 Composizione dello spacecraft

Per raggiungere i precedenti obiettivi lo spacecraft è provvisto di un telescopio (Korsh) che divide la luce così da far processare i campionamenti da un *Visual Instrument* (VIS) [3] e da un *Near Infrared Instrument* (NISP) [4]. Dato che la materia oscura si aggrega sotto l’influenza della gravità, il Visual Imager ha come principale scopo quello di effettuare le rilevazioni delle *gravitational weak-lens*, mentre la componente NISP determina l’immagine fotometrica e le misurazioni spettroscopiche di tipo *slitless* (rilevazioni fatte con filtri che permetto solo la diffrazione della luce che proviene da una regione lontana) applicando un *filter-wheel* (NI-FWA) ed un *grism-wheel* (NI-GWA), ossia due componenti ottiche in grado di rilevare il redshift fotometrico delle onde rilevate dal VIS. Al fianco delle componenti crio-meccaniche è presente

---

<sup>4</sup>Rileva se, oltre ai metalli, tra gli elementi che costituiscono il pianeta vi sono anche ghiaccio e idrogeno

l'intera struttura che comprende l'unità di calibrazione (NI-CU) e il *Focal Plane Assembly* (NI-FPA). Tutto il processo di acquisizione ed elaborazione dei dati è affidato alla Data Processing Unit (NI-DPU), mentre il controllo dello strumento è affidato alla Instrument Control Unit (NI-ICU). Dal punto di vista funzionale la NI-ICU gestisce tutte le operazioni necessarie all'interno del NISP e si interfaccia con lo *Spacecraft Control System* (S/C) per la gestione di vari task tra cui quelli inerenti alla gestione del tempo e della temperatura. Questo scambio, attraverso una interfaccia dedicata, offre il controllo delle componenti elettroniche per quanto riguarda il NI-FWA, il NI-GWA e per la Control Unit, ma anche per il monitoraggio sia dei sensori di temperatura che dei parametri per la gestione dei riscaldatori. Infine, ad eccezione delle due componenti ottiche, è prevista una componente ridondante per ogni elemento descritto; la connessione è di tipo parallelo e non è quindi possibile che una componente di tipo nominale sia connessa con una ridondante. Entrambe le sezioni, sia la nominale che la ridondante, sono installate in un singolo box in grado di offrire isolamento termico e strutturale dall'ambiente spaziale.

## 1.4 Visual Instrument

Lo scopo principale del Visual Instrument (VIS) [3] è effettuare misurazioni di tipo *weak lensing*; la materia oscura (e quella normale) viene aggregata con l'influenza della gravità. Nel momento in cui si rilevano fasci di luce però, questi appaiono distorti rispetto agli oggetti che vi sono nello sfondo ed è proprio effettuando delle misurazioni statistiche rispetto alle forme delle galassie nello sfondo che si può calcolare la massa della materia oscura. Inoltre utilizzando come base altre galassie ancora più lontane, si può determinare come questa materia si è formata. Questo processo è parte dell'espansione dell'Universo che è altresì guidata dall'energia oscura. Capire le caratteristiche e le modalità di formazione della materia oscura porta direttamente a capire le caratteristiche dell'energia oscura. Per misurare la tipica forma delle galassie

il VIS richiede un angolo di vista molto ampio; per coprire la maggior parte del cielo extra galattico nell'arco di una ragionevole durata della missione, lo strumento ha bisogno di un campo pari a circa  $0.5 \text{ deg}^2$ , più del doppio dell'area circolare coperta dalla luna piena ( $0.2^2$ ) nel cielo rispetto ad un osservatore posto sulla superficie terrestre. Per effettuare queste misurazioni lo strumento richiede un dispositivo di chiusura e una unità di calibrazione per il comparto dei rilevatori e delle unità elettroniche sia per elaborare i dati provenienti dal piano focale che per controllare lo strumento. L'interfaccia ottica è parte di un telescopio di 1.2 metri a tre specchi e i fasci da campionare sono divisi tra VIS e NISP attraverso un filtro di interferenza utile per il VIS dato che produce un passa banda tra 550 nm e 900 nm. In generale lo strumento è stato progettato per operare in modo semplice così da permettere la stabilità nei campionamenti. La maggior parte delle osservazioni sono portate a termine dopo un tempo di esposizione di circa 565 secondi con l'intero piano focale attivo; gli output, prima di effettuare di nuovo questa operazione, vengono quindi letti, digitalizzati, bufferizzati e compressi. Per coprire eventuali mancanze nella matrice di rilevazione, per permettere il recupero della risoluzione del campo spaziale e per minimizzare i danni sui dati dovuti dalle radiazioni, per ogni puntamento vengono effettuate 4 esposizioni.

Avendo la stessa durata, l'esposizione del VIS avviene durante l'esposizione spettroscopica del NISP. Nel momento in cui avvengono le rilevazioni fotometriche invece, dati i possibili disturbi causati dal *filter wheel*, le esposizioni del VIS vengono utilizzate con lo scopo di calibrare lo strumento.

## 1.5 Near Infrared Spectrometer and Photometer

Il Near Infrared Spectrometer and Photometer (NISP) [4] è uno strumento che opera in un intervallo tra 920 e 2000 nm ad una temperatura minore di 140 K. Le tipologie di osservazione dello strumento sono 2: in modalità

fotometrica l'acquisizione di immagini avviene con filtri *broad band* in grado di rilevare luce con diverse lunghezze d'onda, mentre in quella spettroscopica le immagini sono raccolte con dei rilevatori senza diffrazione. Le bande coperte per quanto riguarda la modalità fotometrica sono le Y, J e H poste rispettivamente tra 950-1192 nm, 1192-1544 nm e tra 1544-2000 nm, mentre per le rilevazioni spettroscopiche vi sono 4 bande senza diffrazione: rossa con dispersione a 0°, 90°, 180° che coprono un intervallo dello spettro che va da 1250 a 1850 nm e blu con 0° di dispersione che copre uno spettro tra 920-1300 nm. Il NISP è diviso in tre principali blocchi:

- il NI-OMA, che contiene il Mechanical Support Structure (NI-SA), il Termal Control (NI-TC), l'Optical Assembly (NI-OA), il Filter Wheel Assembly (NI-FWA) e Grism Wheel Assembly (NI-GWA) e la Calibration Unit (NI-CU);
- il Detector System (NI-DS) che è formato dal Focal Plane Assembly (NI-FPA) e dal Sensor Chip System (NI-SCS);
- il NI-WE (Warm Electronics) il quale è composto dalla Data Processing Unit (NI-DPU) e dalla Instrument Control Unit (NI-ICU).

### 1.5.1 Optical Mechanical Assembly

Mantenere la temperatura di regime ideale per ogni componente del NISP è di vitale importanza. Il progetto prevede che sia contenuta nel modulo di Payload con un tipo di incapsulamento multistrato per limitare al massimo il contatto con le radiazioni esterne e che operi ad una temperatura di circa 130 K con una stabilità maggiore di 0.3 K durante tutto l'arco della missione.

Il NI-OA è una parte composta da quattro lenti ed opera ad una temperatura compresa tra i 132 e 134 gradi Kelvin che varia in base alla forma delle lenti richiesta; esse infatti al variare della temperatura assumono una curvatura diversa.

Il NI-FWA invece è composto da un motore criogenico, tre filtri, una posizione aperta così che il flusso non subisce interferenze ed una posizione

chiusa che lo blocca. I tre filtri infrarossi (a banda Y, J e H) sono realizzati come dei doppi filtri di interferenza ricoperti con il processo PARMS che ricopre i filtri di uno strato di magnetroni (sottili film composti da materiali dielettrici o da metalli non magnetici) che sono in grado di aumentare le performance dei filtri stessi. Anche il NI-GWA possiede un motore criogenico, quattro *grisms*, ossia griglie di prismi per ogni *grism* in grado di catturare lo spettro relativo e una posizione aperta per evitare interferenze. Tre dei quattro filtri del grism sono rossi con una banda spettrale che va dai 1250nm ai 1850nm, mentre uno è blu con passa banda tra 920 e 1300 nm. Entrambe le componenti lavorano ad una temperatura di 135K.

All'interno dell'unità di calibrazione sono presenti cinque LED ad infrarossi che permettono una calibrazione del piano di rilevazione utilizzando cinque differenti lunghezze d'onda. Anche questa unità lavora in condizioni criogeniche.

### 1.5.2 Detection System

Il NI-DS ha lo scopo di acquisire le immagini dai campioni provenienti dal campo di vista, attraverso un array di  $4 \times 4$  sensori ad infrarossi riprodotti su dei multiplexer e letti dei Sidecar ASIC, ossia dei sistemi programmabili in grado di controllare e digitalizzare le rilevazioni analogiche dei sensori. La temperatura standard è minore di 100K.

### 1.5.3 Warm Electronics

#### Data Processing Unit

Sia la DPU nominale che quella ridondante sono montate intorno ad un bus Compact PCI condiviso usato, data la sua scalabilità, per connettere sistemi industriali oltre che aereospaziali. Esse al loro interno contengono:

- Otto Detector Control Unit (DCU) utilizzate sia per la gestione del clock che, grazie all'ausilio di FPGA, anche per il preprocessamento dei dati a bordo;

- una Central Processor Unit (CPU) anch’essa utile per la gestione e compressione dei dati che saranno poi spediti attraverso una connessione SpaceWire alla memoria centrale dello Spacecraft.

Ad eccezione della DCU, tutte le unità presenti nella DPU sono ridondanti. Il preprocessamento consiste nel raggruppare i dati provenienti dagli stessi campionamenti ma ad unità di tempo diverse, nell'estrazione delle telemetrie o di sottoinsiemi di informazioni utili a terra per il monitoraggio dei vari sistemi.

### **Instrument Control Unit**

Dato che lo scopo di questa tesi è cercare di ricreare un prototipo di una parte della NI-ICU utilizzando dispositivi a basso costo e facilmente accessibili, le successive sezioni spiegano più in modo dettagliato il funzionamento della componente sia dal punto di vista software che architetturale.

La struttura della NI-ICU è costituita da 4 board:

**LVPS** Il Low Voltage Power Supply è un elemento alimenta tutte le componenti della NI-ICU.

**CDPU** La Central Data Processing Unit è costituita da una FPGA con equipaggiato un processore LEON 2 con degli IP cores per quanto riguarda le interfacce MIL-STD 1553 connesse al Data Processing Unit e una interfaccia SPI per la comunicazione con il resto dei moduli.

**DAS** La Driver and Acquisition Support è una FPGA utile al controllo logico delle componenti ottiche del NISP, dei LED di calibrazione e dei riscaldatori. In particolare le componenti presenti nel DAS sono:

- 2 motori passo-passo, uno per il *filter-wheel* ed uno per il *grism-wheel*;
- una frizione per ogni motore;
- 2 sensori di posizione per rilevare se i motori si trovano nella *home position*;

- 2 riscaldatori, uno in NI-OMA dove sono collocati i motori e l’altro nel piano focale di rilevazione;
- 5 LED per l’unità di calibrazione controllati attraverso la Pulse Width Modulation (PWM).
- 12 sensori di temperatura per rilevare sia la componente interna che esterna.

**BACKPLANE** Una board trasversale necessaria a connettere tutte le altre. Sono quindi presenti le interfacce per i protocolli MIL-STD 1553, SPI e DC per il passaggio di corrente.

## 1.6 MIL-STD 1553

Il protocollo MIL-STD 1553 [5] è uno standard militare pubblicato dal Dipartimento di Difesa degli Stati Uniti che definisce le caratteristiche elettriche, meccaniche e funzionali di un bus seriale utile al trasferimento di dati. Nella sua versione più recente (la MIL-STD 1553B) il protocollo è composto da due bus paralleli, uno nominale ed uno ridondante a cui sono connesse quattro diverse componenti: un Bus Controller (BC), un Remote Terminal (RT), un Bus Monitor ed un Bus Backup Controller (questi ultimi due sono opzionali) (Fig. 1.3).

Il Bus Controller è l’unico dispositivo master presente ed ha il compito di inizializzare tutte le informazioni utili al trasferimento dei dati, che devono essere del formato comando/risposta. Il BC invia quindi dei comandi al Remote Terminal che li riceve attraverso il bus e li consegna al sottosistema a cui è interfacciato. Lo standard prevede che possano essere presenti fino a trentuno interfacce RT, che possono essere sia già integrate nei sottosistemi che separate da essi. Oltre alla comunicazione *BC to RT* il protocollo permette anche comunicazioni di tipo *RT to BC* e *RT to RT*.

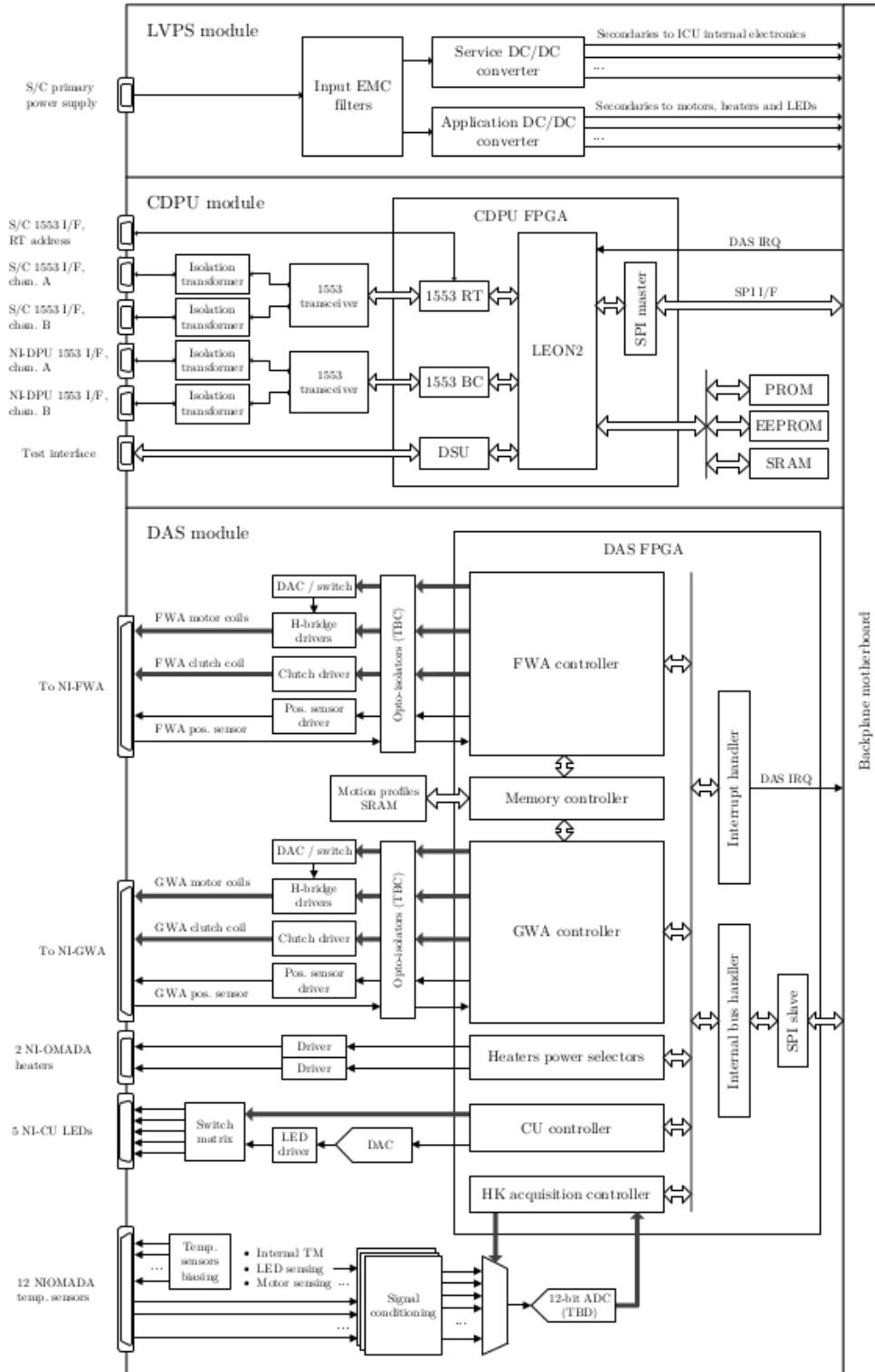


Figura 1.2: Modello architetturale delle componenti presenti nella NI-ICU

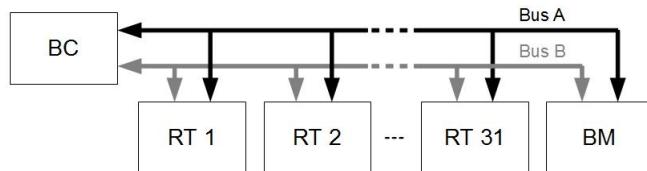


Figura 1.3: Protocollo MIL-STD 1553

Nel caso in cui fosse presente il Bus Monitor, esso è un modulo passivo che ha lo scopo di monitorare e salvare le transazioni avvenute nel bus tra il BC e uno dei RT.

Infine, nella configurazione del protocollo può essere presente un Backup Bus Controller, ossia una unità di backup del BC che viene usata nel caso di rilevazione di errori provenienti dal Bus Controller.

## 1.7 Analisi delle componenti della NI-ICU

Come si nota dalla Fig. 1.2 il modulo LVPS è composto da dei filtri utili all’isolamento elettromagnetico (EMC) connessi a due convertitori di voltaggio, uno utilizzato dalle componenti elettroniche interne della ICU e l’altro in grado di alimentare gli attuatori ed i sensori del modulo DAS. La FPGA presente nel modulo CDPU è in grado di comunicare con lo spacecraft attraverso due interfacce ridondanti di tipo MIL-STD 1553 con le componenti PROM, EEPROM e SRAM utilizzando una interfaccia di tipo RS-232. Infine la componente DSU permette di far comunicare il processore con l’interfaccia di test. Il connettore RS-232 permette di connettersi alla NI-ICU per sviluppare e gestire il software presente all’interno, offre pieno accesso a tutti gli spazi di memoria (EEPROM, PROM, SRAM) e a tutti i registri presenti su ogni singolo chip. Infine, oltre che comunicare con la DPU il processore ha anche il compito di controllare sia i moduli che i sottosistemi presenti nella DAS con l’invio di comandi predefiniti. Questa comunicazione tra CDPU FPGA e DAS FPGA avviene attraverso l’uso del protocollo SPI, dove la CDPU svolge il ruolo di master e la DAS quello di slave.

Oltre all’interfaccia SPI la DAS FPGA è composta da un bus interno in grado di far comunicare tra loro i controller sia dei sensori che degli attuatori. Facendo sempre riferimento alla Fig. 1.2, dall’alto verso il basso vi sono i due moduli per la gestione dei motori utili al movimento delle unità ottiche: essi sono identici dal punto di vista architetturale, ma il *controller FWA* è connesso al NI-FWA mentre il NI-GWA al *GWA controller*. Le due sezioni controllano i due motori bipolarì in grado di effettuare una rivoluzione in 360 passi, una frizione per mantenere solido ogni singolo stato e un sensore di posizione che permette di stabilire se il motore si trova nella *home position* o meno (non presentano quindi la possibilità di avere dei feedback rispetto alla posizione attuale). Il movimento dei due motori è possibile grazie al caricamento di un *motion profile*; nella FPGA DAS è infatti presente un sottosistema connesso ad entrambi i moduli FWA e GWA in grado di caricare dei profili di movimento precedentemente salvati in una SRAM situata all’interno della DAS. Ogni profilo di movimento è composto da  $N$  micropassi ed ognuno di essi sono composti da:

- Durata del passo: Un valore di 16 bits che corrisponde all’ammontare di tempo nel quale i valori di energia delle due fasi devono essere applicati al micropasso corrente;
- Phase A: valore di 8 bit che corrisponde al valore da applicare alla fase A del motore passo-passo;
- Phase B: valore di 8 bit da applicare alla fase B del suddetto motore.

Complessivamente quindi ogni record di un profilo motore è formata da una parola di 32 bit data dalla composizione dei tre precedenti campi. Ogni movimento o concatenazione di movimenti è determinata dai meccanismi di calibrazione e da sequenze di osservazioni prestabilite. Durante l’inizializzazione, o quando richiesto, la NI-ICU può comandare il posizionamento dei motori in uno punto predefinito chiamato *home position*. Per questo ogni componente motore ha un sensore dedicato in grado di verificare se esso si

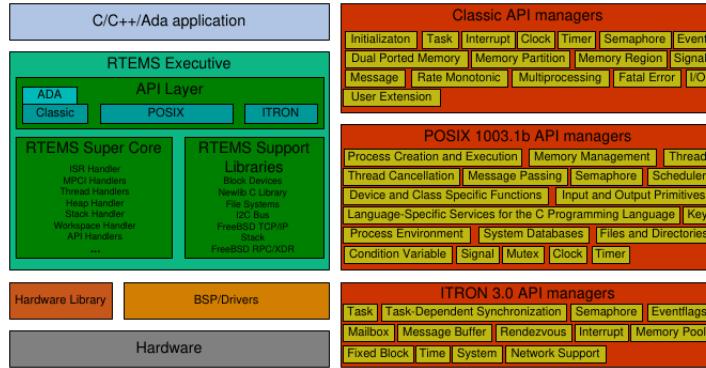


Figura 1.4: Architettura di RTEMS

trova o meno nella posizione di *home* e se richiesto, è possibile raggiungerla seguendo una apposita procedura.

Comunicando con l'unità di calibrazione la CDPU è in grado gestire 5 LED attraverso l'uso della PWM. Dato che la CU ha il compito di garantire una calibrazione di piccola scala delle unità di rilevazione, la NI-CU comporta la gestione di al più un LED alla volta; il *duty cycle* gestisce l'intensità di corrente che arriva ad ogni LED modificandone l'intensità di radiazione.

Anche se non c'è nessun controllo automatico della temperatura, all'interno del modulo DAS sono presenti due riscaldatori che possono essere comandati direttamente da terra. Dato che uno dei riscaldatori si trova all'interno del NI-FPA, questi, invece di essere comandati con l'utilizzo di PWM, usano un controllo di tipo DC.

Infine il driver per le telemetrie presente nella DAS FPGA, ha lo scopo di utilizzare dei sensori per monitorare il controllo delle temperature presenti nei vari compartimenti. Avvalendosi di una catena di multiplexer analogici, la comunicazione delle varie rilevazioni è svolta in modo sequenziale.

## 1.8 RTEMS

Real Time Executive for Multiprocessor System [6] è un Real Time Operating System (RTOS) open-source progettato per sistemi embedded. Questo

sistema operativo supporta diversi tipi di processori, inclusi quelli di tipo SPARC (come ERC32 o la famiglia LEON), supporta sia numerosi standard come POSIX e ISO C/C++ che alcune caratteristiche base del kernel. Per quanto riguarda le connessioni di rete permette l'utilizzo dello stack TCP/IP, UDP. Include inoltre la possibilità di effettuare debugging attraverso la porta ethernet o seriale; supporta diversi tipi di file system come IMFS e FAT. RTEMS è stato progettato per supportare applicazioni che hanno una minore flessibilità per quanto riguarda il tempo, lasciando quindi allo sviluppatore maggiore libertà nello sviluppo di queste funzionalità.

Tra le caratteristiche che il sistema operativo offre c'è il supporto al multitasking; esso implementa tecniche di scheduling come l'Event Driven e il Rate Monotonic. Il kernel inoltre dà la possibilità di selezionare i moduli che necessitano di essere caricati così da evitare ritardi non necessari o un eccessivo consumo di memoria.

Utilizzando BOOTP [7], RTEMS offre la possibilità di caricare attraverso la rete un modulo kernel o un eseguibile. Il file system possiede parte delle caratteristiche UNIX (permette ad esempio di montare device o avere una struttura di cartelle gerarchica) e POSIX con un set di routine per la manipolazione di file e cartelle.

Data la difficoltà di accesso alla board, RTEMS permette di effettuare debugging da remoto di tutti gli applicativi scritti in C/C++. Dal punto di vista strutturale (Fig. 1.4) può essere diviso in tre livelli: operazionale, organizzativo e concettuale. Il livello operazionale definisce la relazione tra il sistema operativo e l'applicazione utente; l'organizzativo invece è la parte in cui gli sviluppatori organizzano e strutturano il codice sorgente del sistema operativo; infine il livello concettuale è diviso a sua volta in altri tre livelli: uno per il supporto all'hardware, uno dove è contenuto il kernel e l'ultimo utile per lo sviluppo del codice utente.

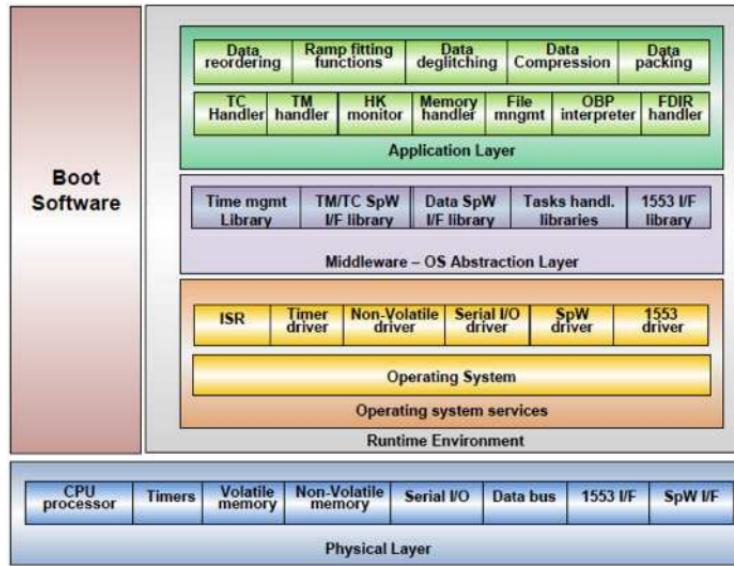


Figura 1.5: Livelli dell’Application Software della NI-ICU

## 1.9 NI-ICU Application Software

Il NI-ICU Application Software (NI-ASW) [8] è progettato e sviluppato per integrarsi con la versione qualificata per lo spazio del sistema operativo real-time RTEMS. Il suo scopo è quello di gestire l’interfacciamento con la DPU, il NI-OMA attraverso la DAS board e tutte le funzionalità utili al comando e monitoraggio del NISP. In particolare quindi in NI-ASW deve:

- Gestire le modalità operative del NISP;
- Comunicare con lo spacecraft attraverso il protocollo MIL-STD 1553 per ricevere TeleCommands e inviare Telemetrie;
- Controllare l’unità di calibrazione;
- Controllare il NI-FWA e il NI-GWA;
- Monitorare e gestire le telemetrie;
- Controllare la temperatura nel NI-SA e NI-FPA utilizzando i sensori ed i riscaldatori.

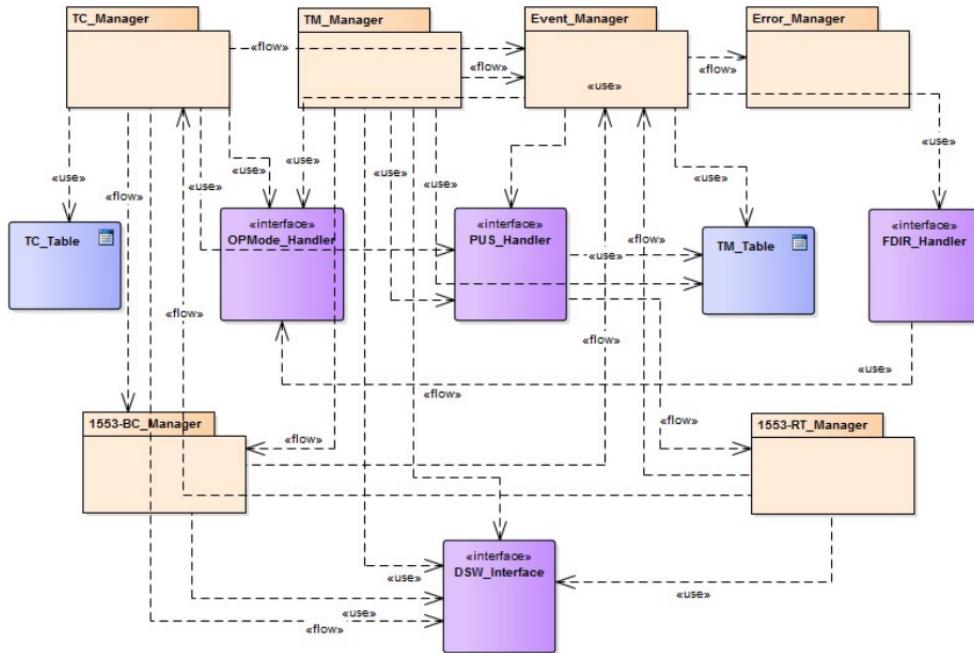


Figura 1.6: Diagramma delle classi dell'Application Software della NI-ICU

Date queste caratteristiche, l'applicativo può essere diviso in diversi livelli (Fig.1.5): a livello hardware sono presenti tutte quelle funzionalità necessarie per la gestione dei dispositivi contenuti nella ICU al di sotto di RTEMS; un livello intermedio necessario per lo sviluppo del software indipendentemente dalle scelte sottostanti; infine il livello dove è contenuto il NI-ASW.

Come si nota dalla Fig. 1.6 una volta che da terra è stata ricevuta la tabella contenente i comandi da impartire, essi vengono verificati invocando il servizio Packet Utilisation Standard (PUS), il quale è in grado di stabilire se i pacchetti ricevuti corrispondono allo standard [9]. Se accettati, vengono decodificati, distribuiti ai dispositivi ed eseguiti in base alla procedura stabilita. Il Time Manager invece riceve le informazioni di sincronizzazione dallo spaccraft attraverso l'interfaccia MIL-STD 1553, aggiorna il tempo di sistema, crea dei pacchetti e aggiunge come contenuto lo stato corrente aggiornato. Un'altra parte del Time manager ha inoltre il compito di ricevere attraverso polling sul protocollo SPI i pacchetti con i monitoraggi effettuati dai vari dispositivi; genera a questo punto dei nuovi pacchetti di tipo TM, li formatta

in tabelle e le spedisce allo spacecraft attraverso l’interfaccia MIL-STD 1553. Nel caso in cui una o più soglie siano superate, viene invocato il gestore degli eventi: esso invia gli allarmi all’interfaccia MIL-STD 1553 e gestisce in modo autonomo i fallimenti rilevati. Nel caso in questione, l’Event Manager interpella l’Error Manager che, oltre a gestire tutti gli errori del software, specificando il tipo di errore avvenuto, genera il relativo evento e lo consegna al gestore degli eventi. Vi sono infine le due interfacce MIL-STD 1553: la MIL-STD 1553RT fa polling sul buffer 1553 per rilevare se vi sono nuovi dati, invia nuovi TC all’interprete dei comandi. Legge le *TM tables* e le fornisce al S/C. MIL-STD 1553BC infine gestisce la comunicazione con la DPU, invia e riceve sia i pacchetti che le informazioni riguardo il tempo di sistema e la sincronizzazione degli eventi.

La ICU è la sola parte del modulo NISP ad interfacciarsi con il *Service Module* e quindi con il controllo da terra. Tutte le componenti meccaniche contenute nella DAS sono tutti moduli connessi e gestiti direttamente dalla ICU attraverso un controllo con accesso SPI, mentre la DPU è una unità intelligente in grado di ricevere comandi e produrre telemetrie; dato che le sue capacità necessitano di un controllo remoto, esse sono manipolate da terra attraverso la ICU.

Dato che l’ASW deve essere in grado di operare per diverse ore senza alcun intervento proveniente da terra è stato necessario stabilire delle politiche di recovery nel caso di rilevazione di errori. Sono stati perciò definiti un insieme di parametri di default e procedure per il controllo delle funzioni al fine di rilevare eventuali criticità. In generale, per quanto riguarda il NISP l’approccio previsto evita complicate procedure di recovery e, nel caso fosse necessario, effettua una transizione ad uno stato definito con “configurazione sicura”. Questo dà la possibilità di effettuare successivi interventi da terra e ridurre al minimo il tempo di inattività del sistema. Inoltre, dato che il controllo e i rilevamenti della temperatura interna sono di vitale importanza, ogni procedura attuata durante una criticità non prevede lo spegnimento del modulo. Lo stato del sistema di controllo delle temperature sarà perciò in

*idle*, tranne quando il fallimento avviene nel suo compartimento [10].

## 1.10 La roadmap di Euclid

Euclid è una missione approvata nel 2012 che ha visto sempre nello stesso anno l'inizio dell'implementazione del Payload Module (PLM) con l'Airbus Defence and Space di Tolosa. Nell'anno successivo sono stati posti sotto contratto l'implementazione dei vari sottosistemi con l'aiuto di altri partner e nel 2015 è stato approvato il design preliminare della missione presentato dall'Euclid-Consortium<sup>5</sup>. I strumenti VIS e NISP sono stati rispettivamente assegnati ai team di UCL-Mullard Space and Science Laboratory, al CNES e al laboratorio di astrofisica di Marsiglia. Il loro programma di studio è iniziato nel 2014 e ora si trovano entrambi alla fase D. A fine 2016 è stato svolto un Critical Design Review sugli strumenti. Per la metà del 2018 è prevista una review completa della missione. Tra il 2019 e il 2020 sono previsti l'inizio dei test completi e l'integrazione con il sistema di volo. Se a metà 2020 il piano di volo sarà accettato, il lancio è previsto per la fine dello stesso anno[11].

Per quanto riguarda l'Application Software, dopo che è stata superata la Critical Design Review (CDR) di settembre 2016 che riguardava il design architettonale, lo sviluppo del codice si sta soprattutto concentrando sull'implementazione dei servizi PUS, la gestione sia dei comandi da terra che dei dati provenienti dal calcolo delle telemetrie. L'immagine dell'ASW gira su un Elegant BreadBoard (EBB) che rappresenta la CDPU dell'unità di volo, mentre la DAS board è sostituita con un insieme di elementi che comprendono una FPGA che può emulare l'operazione del SECOIA ASIC (ossia la board che controlla le periferiche in base ai comandi ricevuti dalla NI-CDPU) previsto nel modello di volo finale. Questo setup, oltre che emulare la DAS, permette l'alimentazione della CDPU e la connessione dell'interfaccia debug a un connettore RS-232. Per quanto riguarda lo spacecraft, il protocollo MIL-

---

<sup>5</sup><https://www.euclid-ec.org/>

STD 1553 è attualmente simulato con un dispositivo Ballard 1553 USB, che può essere gestito ad alto livello tramite una GUI o con delle API C. Il simulatore dello S/C è costruito utilizzando delle API C fornite dalla Ballard Technology e, per ottenere delle simulazioni realistiche, offre la stessa gestione del tempo e della sincronizzazione del dispositivo finale. Lo scambio dei pacchetti contenenti sia TeleCommand (TC) che Telemetrie (TM) avviene attraverso il protocollo adottato dallo Spacecraft così da verificare il corretto controllo, generazione e gestione degli stessi.

Come per lo S/C anche la DPU è simulata attraverso le stesse API dello Spacecraft. Questo permette di testare il comportamento dell'Application Software in presenza di una o due DPU (nominale e ridondante).

Come detto, in questo momento il modulo DAS è stato sviluppato con un setup che prevede l'uso di una FPGA. Per costruire una piattaforma emulata flessibile ed in grado di supportare lo sviluppo e il test dell'ICU-ASW è stato progettato, congiuntamente da Italia e Spagna (INFN e UAH), un sistema più ampio composto da altre FPGA chiamato SecchiaSim [10].



# **Capitolo 2**

## **Progettazione e sviluppo del dimostratore**

Dopo aver definito gli scopi e gli elementi principali di Euclid, come introduzione a questo secondo capitolo sarà fatta una panoramica che spiega i passi necessari per la progettazione e lo sviluppo di una missione spaziale, così da poter inquadrare dove è previsto l'uso di SecoiaSim e stabilire se è possibile introdurre altre metodologie che possano essere un'aggiunta o una alternativa rispetto a quelle convenzionali. Dopo aver indentificato lo scopo della tesi, si passerà quindi a capire quali sono state le componenti utilizzate per lo sviluppo di questo sistema alternativo a SecoiaSim e come si è regolato il loro interfacciamento rispettando il più possibile le specifiche della missione.

### **2.1 Ciclo di vita di uno spacecraft**

Dato che ad ogni missione spaziale collaborano decine di organizzazioni, è molto importante la schedulazione delle fasi e del tempo che porta alla data del lancio. In particolare, per quanto riguarda l'implementazione di uno spacecraft sono previste 5 fasi di sviluppo [12]. La fase A (Fig. 2.1) consiste nello studio preliminare del problema; esso può essere svolto sia da delle in-

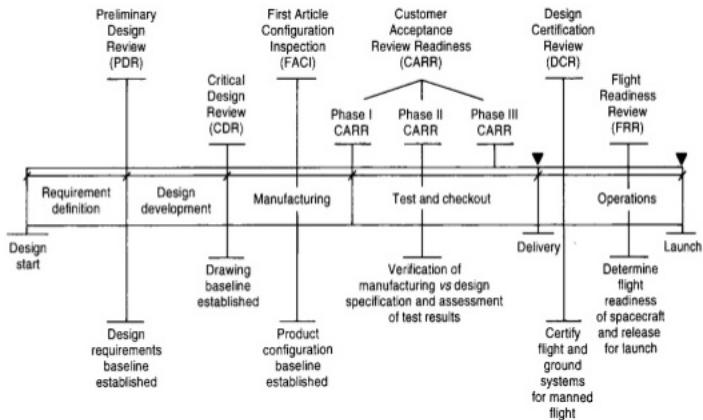


Figura 2.1: Ciclo di vita di uno spacecraft

dustrie che dai team di ricerca basandosi su studi preventivamente effettuati. In questa fase di analisi concettuale ci si chiede che tipo di configurazione debba avere lo spacecraft, quanto dovrebbe costare o quanto tempo dovrebbe rimanere in attività. La fase B invece prevede la pubblicazione di un bando per far sì che le industrie possano presentare i propri progetti di sviluppo e, in base alle ipotesi fatte nella fase iniziale, scegliere il migliore. Alla fine di questa fase viene quindi effettuata quella che si chiama una PDR (*Preliminary Design Review*) in cui si prende atto di ciò che sono le specifiche e di come sono cambiate rispetto alle previsioni iniziali. Durante le fasi C e D avviene lo sviluppo del progetto; dapprima, in quella che viene anche chiamata fase delta-B si studiano a fondo le specifiche offerte dal fornitore che ha vinto l'appalto, vengono effettuate eventuali modifiche al design architettonico e si inizia il vero sviluppo del codice. E' a questo punto della roadmap che le altre aziende incaricate della progettazione dei sottosistemi iniziano a creare i sistemi di test delle interfacce. La fase C si conclude con la CDR che permette al cliente di fare un confronto tra quelle che sono le specifiche e ciò che è stato prodotto. Nel momento in cui questa ultima *milestone* viene raggiunta, nella Fase D chiamata anche ATLO (*Assembly Test and Launch Operation*), inizia la fase di assemblaggio dello spacecraft con le unità che presentano le medesime caratteristiche di quelle utilizzate nella Fase C, ma

progettate per ambienti spaziali. Sono a questo punto effettuati i test di valutazione funzionali ad ogni fase della missione; sono svolte prove ambientali simulando sia l’ambiente di lancio che prove di robustezza nel caso di esposizione agli agenti spaziali. Si testano inoltre i canali di comunicazione e il giusto comportamento sia del codice che di tutte le unità interne. Se tutto è approvato con successo inizia la fase di lancio e tutte le componenti subiscono una revisione finale prima del lancio effettivo; quando lo spacecraft è partito per raggiungere la destinazione ha inizio la fase E che dura per tutta la permanenza dello spacecraft nello spazio.

## 2.2 Scopo della tesi

Come accennato nel ciclo di vita precedentemente descritto e come emerso nella roadmap di Euclid [11], vi è una collaborazione molto forte tra decine di società sia pubbliche che industrie private. Prendendo in esame lo sviluppo del modulo NISP, la progettazione del codice sorgente è assegnata alle sezioni dell’Istituto Nazionale di Fisica Nucleare e Istituto Nazionale di Astrofisica (INFN e INAF) di Torino, Bologna e Padova per quanto riguarda l’Italia e altri team in Spagna e Francia. Una parte della fase C della missione Euclid prevede lo sviluppo di una piattaforma hardware chiamata SecoiaSim utile alla progettazione e test del codice sorgente e con lo scopo di emulare nel funzionamento il modulo NISP che sarà presente nello spacecraft; lo sviluppo è stato in parte assegnato all’Italia ed in parte alla Spagna e per l’intera fase C sarà disponibile una sola unità del dispositivo. Questo potrebbe comportare problemi di natura organizzativa oltre che dell’aumento dei tempi necessari allo sviluppo e, come verrà analizzato successivamente, anche se questo task è assegnato a organizzazioni pubbliche, parte delle componenti di SecoiaSim provengono da aziende private.

Negli ultimi decenni lo sviluppo di codice open-source è enormemente aumentato [13] tanto che sia centri di ricerca pubblici che aziende sviluppano parte dei loro prodotti con codice libero; esempi di questo tipo possono essere

Canonical, Red Hat, Google o la stessa NAS [14]. Ancor più recentemente è iniziato a circolare in modo sempre più consistente il concetto di hardware open-source [15]. I suoi fondamenti si basano sulla possibilità per chiunque di “studiare, modificare, distribuire, creare e vendere il progetto o dell’hardware basato sul progetto”. Ciò non solo richiede che le procedure siano ben documentate, ma anche che i progetti siano chiari e possibilmente semplici così da poter essere modificati anche dai non esperti. In contrasto agli hardware proprietari che presentano difficoltà progettuali tali da richiedere molto spesso il conseguimento di appositi certificati, l’hardware libero riflette una maggiore possibilità di collaborazione sia nella risoluzione dei problemi che nel miglioramento delle prestazioni. Questo concetto è valido soprattutto nel campo scientifico che, data la vasta gamma di esigenze, necessita di una tale flessibilità e diversificazione sia del software che dell’hardware difficile da raggiungere utilizzando strumenti proprietari.

Lo scopo di questa tesi è quindi quello di verificare se e nel caso dove all’interno del ciclo di vita di una missione spaziale, sarebbe possibile utilizzare strumenti software ed hardware low-cost e open-source.

Prendendo in esame il caso Euclid è emerso che una possibile applicazione di questo concetto può avvenire durante la fase C della missione. Come detto, il task che riguarda lo sviluppo ed il test del codice sorgente per il modulo NISP avviene con l’utilizzo di SecoiaSim, uno strumento costoso, proprietario e disponibile in una sola unità. Anche se importante [16], al di là del fattore economico utilizzare strumenti liberi ed a basso costo darebbe la possibilità di produrre un numero maggiore di test boards, ma avendo allo stesso tempo un abbattimento dei costi ed un plausibile aumento della produttività.

## 2.3 NI-CDPU

Nella fase C di sviluppo dello spacecraft la Central Data Processing Unit è stata realizzata con una Elegant BreadBoard. Essa è quindi un dispositivo di

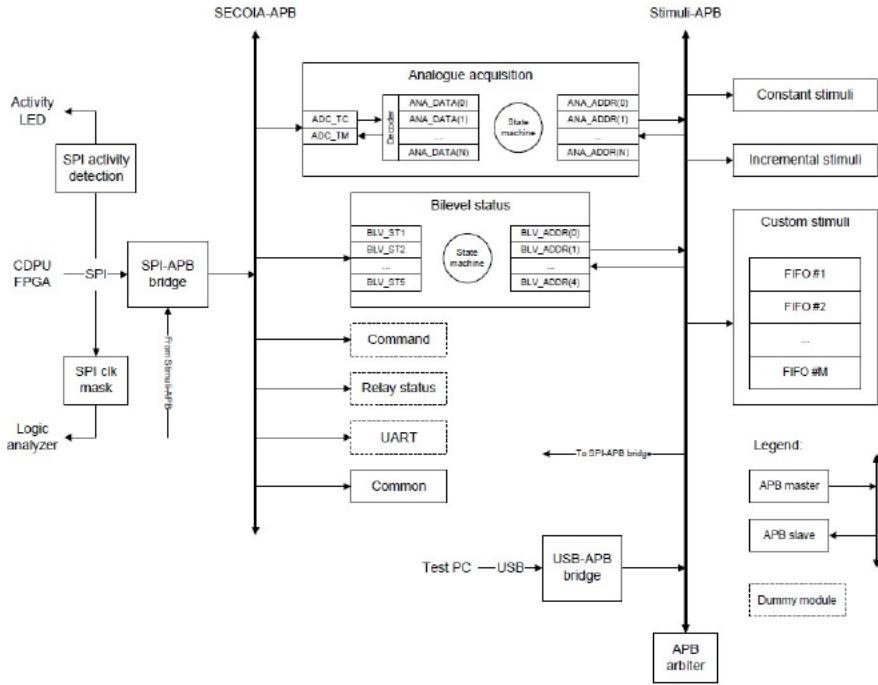


Figura 2.2: Diagramma a blocchi di SecoiaSim

livello TRL4 che ha superato con successo i test ambientali in laboratorio<sup>1</sup>. Dato che le specifiche non possono essere divulgate, non è possibile fare uno studio più dettagliato.

## 2.4 SecoiaSim

Al fine di riprodurre le funzionalità del dispositivo SECOIA ASIC, la componente posta nella NI-ICU che ha il compito di gestire i sensori e gli attuatori è stato progettato SecoiaSim [10, 17] (Fig. 2.2). Esso è composto da una FPGA (Spartan3-A) e permette di eseguire test riguardo l'Application Software ed il Driver Software presente nella DAS. Una delle proprietà di questo dispositivo è la modularità; tutti i controller e quindi tutte le risposte

<sup>1</sup>[http://space.epfl.ch/webdav/site/space/shared/industry\\_media/Annex\\_B.pdf](http://space.epfl.ch/webdav/site/space/shared/industry_media/Annex_B.pdf)

sia dei rilevatori che degli attuatori sono emulati attraverso una FPGA in base ai test da effettuare.

Dal punto di vista funzionale i compiti più importanti che SECOIA ASIC deve svolgere sono l'invio dei comandi agli attuatori e l'acquisizione dei dati dai sensori. Le funzioni che comportano l'invio di comandi avranno una influenza su quelle di acquisizione, ma nel caso di SecoiaSim tutti i dati acquisiti saranno deterministici e per niente influenzati dai comandi impartiti. A questo scopo esiste solamente un monitor SPI che rileva il passaggio di un segnale e quindi di un comando dalla CPDU a SecoiaSim, ma il comando in sé sarà ignorato. Le funzioni di acquisizione invece sono più elaborate ed è possibile acquisire i valori delle telemetrie sia analogiche che digitali. Più precisamente i dati telemetrici di risposta posso essere di tre tipi:

- Ad impulso costante: le telemetrie di alcune fonti possono essere configurate a priori;
- Ad impulso incrementale: in questo caso il primo valore di ritorno è configurato manualmente, mentre ad ogni richiesta la risposta sarà sempre maggiore alla precedente;
- Ad impulso variabile: ogni valore telemetrico di ritorno è preconfigurato indipendentemente ed inserito in una coda FIFO.

La Fig. 2.2 mostra ad alto livello l'architettura di SecoiaSim. Il dispositivo è costruito intorno a due Advanced Microcontroller Bus Architecture (AMBA-BUS): un SECOIA-APB con lo scopo di replicare lo stesso modulo interno che collega l'unità SPI con il resto dei moduli dello slave, in cui tutti i registri sono mappati su APB-bus di SECOIA ed accessibili dalla connessione dell'unità del *Bridge SPI*; ed un Stimuli-APB invece è usato per fruire i valori di risposta delle telemetrie. Il modulo Analog Acquisition simula l'acquisizione delle TMs interagendo con lo Stimuli-APB bus, mentre dato che i comandi provenienti dal master sono ignorati, il Command Module ha la sola funzione di offrire accesso ad una serie di registri che non hanno alcuna

funzionalità. Anche i moduli Relay e UART non hanno implementate funzioni importati e ritornano un errore nel caso si provi ad accedervi. Diverso invece il Common Module che come nel SECOIA reale offre l'accesso ad una serie di registri che contengono diverse funzioni globali.

## 2.5 Requisiti del dimostratore

Basandosi sia su quanto detto, sia su altre specifiche ora riservate [18, 19, 20], il sistema, al fine di riprodurre il comportamento di SecoiaSim deve rispettare dei requisiti specifici. Lato hardware sono necessarie delle board in grado di supportare il funzionamento dei moduli CDPU e DAS. L'unità di emulazione della CDPU dovrà quindi essere in grado di supportare un sistema operativo (meglio se di tipo RTOS), il processore quindi deve avere una architettura compatibile a quella del LEON2 ed è necessario il supporto alla comunicazione SPI. Per quanto riguarda la board di simulazione della DAS, anch'essa deve offrire la possibilità di comunicare in modo seriale con una velocità di clock sufficiente da permettere una comunicazione affidabile ed in linea con le specifiche ufficiali, permettendo anche di gestire l'impulso inviato alle unità esterne.

Per evitare valori di ritorno simlati, le specifiche necessarie sia dei rilevatori che degli attuatori devono essere quantomeno analoghi ai dispositivi reali. In particolare:

- I due motori devono essere bipolari di tipo *stepper* e il dispositivo che controlla la *home position* deve essere in grado di stabilire se il motore si trova nella posizione di default o meno;
- I LED di calibrazione che emulano la Control Unit devono essere in grado di modificare la loro intensità luminosa ed il dispositivo di selezione deve permettere che al più ve ne sia uno acceso;
- Le unità che simulano i riscaldatori devono riprodurre il concetto di “aumento o diminuzione dell'irrorazione di calore”;

- I sensori devono permettere la rilevazione della temperatura del comparto in scala Kelvin.

Dal punto di vista software per quel che riguarda la CDPU si deve avere la possibilità di seguire lo standard ISO C/C++, avere un modulo kernel che permetta la gestione e l’interfacciamento con il protocollo SPI. Per l’emulazione della DAS, la board deve avere un meccanismo che permetta sia la possibilità di fare polling sull’interfaccia seriale che di connettere e gestire le numerose periferiche necessarie.

## 2.6 Scelte hardware

### 2.6.1 Raspberry Pi

Il LEON2 [21] è un processore a 32-bit con una architettura SPARC V8, progettato per applicazioni embedded con le seguenti caratteristiche:

- ha due cache separate per istruzioni e dati;
- un interrupt controller;
- un supporto al debug attraverso un *trace buffer*;
- una porta I/O a 16-bit;
- un MAC ethernet ed una interfaccia PCI.

Permette inoltre comunicazioni di tipo SPI, I2C e ATA oltre che avere un controller SDRAM a 32-bit, si basa su una architettura AMBA AHB/APB così da permettere anche l’aggiunta di nuovi moduli. La sua frequenza di calcolo raggiunge i 125 MHz e la sua modularità ne permette l’installazione sia su architetture ASIC che FPGA.

Come dispositivo SoC per emulare la CDPU è stato utilizzato un Raspberry Pi versione B+ [22]; esso basa la sua tecnologia su un BCM2835 [23] che include un ARMv6 32-bit a 700 MHz, una RAM da 512 Mb, una cache di

livello 1 da 16 Kb e una di livello 2 da 128 Kb. Attraverso una GPIO a 40 pin supporta comunicazioni SPI sia master che slave, I2C, UART0, UART1 e permette la modulazione dell'onda con la tecnica della Pulse Width Modulation (PWM).

### 2.6.2 Arduino Mega

SecoiaSim emula la DAS board attraverso l'utilizzo di una FPGA che usa una interfaccia AMBA per connettere le sue componenti esterne. Per evitare di avere componenti emulate come in SecoiaSim, nel prototipo implementato si interfaccia quella che è la DAS board con componenti fisiche reali. A tal proposito si è scelto utilizzare un Arduino Mega equipaggiato con un ATMEGA2560 [24], che fa parte della famiglia AVR progettata da ATMEL. Questo microcontrollore ha una architettura RISC a 8-bit, con 32 registri connessi direttamente alla ALU; esso inoltre possiede una memoria flash di 256 Kb, una EEPROM di 4 kb e una SRAM da 8 Kb, 86 I/O *lines* connesse ad una GPIO che permette comunicazioni SPI, I2C, UART, 12 canali PWM a 16-bits, 12 ADC pin ed una velocità di clock di 16 MHz. La memoria flash è progettata in modo da essere riprogrammata attraverso una interfaccia seriale grazie ad un applicativo presente nella Flash boot che rimane in esecuzione anche quando il software presente nella Flash Application viene aggiornato. Richiede un voltaggio in input tra 6V e 20V.

### 2.6.3 Stepper motor

Per riprodurre le funzionalità dei motori presenti nel NI-FWA e NI-GWA sono stati utilizzati due stepper motor 28BYJ-48 [25] connessi a due driver ULN2003A [26]. Il 28BYJ-48 è un motore passo-passo con una *gear ratio* di 1:64 e un angolo di passo pari a  $5.625^\circ$  e richiede un voltaggio in input pari a 5V. Questo motore unipolare a 5 cavi può essere comandato attraverso un driver ULN2003A, composto da 7 coppie di Darlington transistor di tipo npn che caratterizza un più alto voltaggio in uscita ed ottimo per gestire dispo-

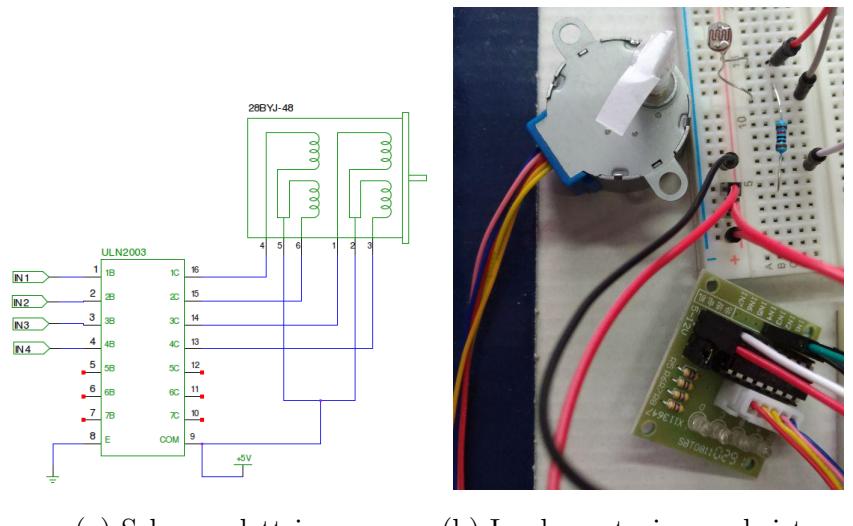
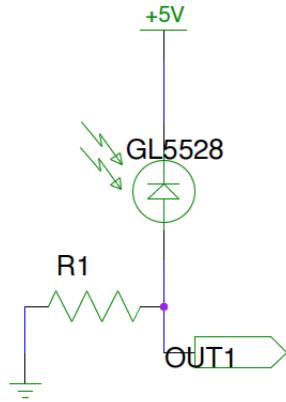


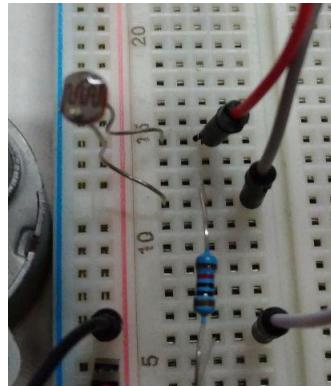
Figura 2.3: Stepper motor 28BYJ-48

	1	2	3	4	5	6	7	8
IN4	1	1	0	0	0	0	0	1
IN3	0	1	1	1	0	0	0	0
IN2	0	0	0	1	1	1	0	0
IN1	0	0	0	0	0	1	1	1

Tabella 2.1: Sequenza di switch del motore 28BYJ-48



(a) Schema elettrico



(b) Schema elettrico di GL5528

Figura 2.4: Fotoresistore GL5528

sitivi CMOS a 5V. Il driver che guida lo stepper, oltre al pin del voltaggio, richiede altri 5 pin di input: uno per la terra e altri quattro per le fasi del motore. Esso infatti è formato da 4 bobine che prendono in input 8 diversi segnali (tab. 2.1): questo viene detto anche metodo del mezzo-passo in cui nel primo passo si dà energia alla bobina 1, nel secondo alle bobine 4 e 3, nel terzo solo alla 3, nel quarto alla 3 e 2, poi solo alla 2 e così via (Fig. 2.3).

#### 2.6.4 Celle foto-conduttrive

Come detto i due motori passo-passo connessi alla DAS non presentano nessun meccanismo di feedback della posizione. In questo caso il comportamento del rilevatore che verifica se essi si trovano nella *home position* viene svolto da due foto resistori. Queste componenti ritornano al microcontrollore un valore che indica la variazione di luce rilevata. Il dispositivo [27], che richiede voltaggio minimo di 3.3V, non è altro che un conduttore di corrente con una resistenza che varia in base al grado di esposizione alla luce. Il modello GL5528 (Fig. 2.4) calcola un valore corrispondente ad 1 K $\Omega$  di resistenza nel caso di forte luce e 10 K $\Omega$  nel caso di un ambiente poco luminoso. Essendo un rilevatore analogico esso può essere connesso ad uno dei 12 pin analogici presenti nel ATMEGA2560.

### 2.6.5 Rilevatore di temperatura

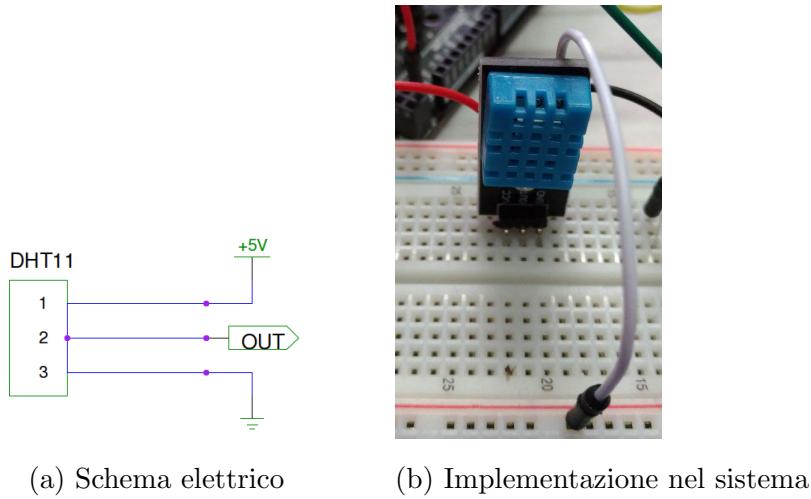
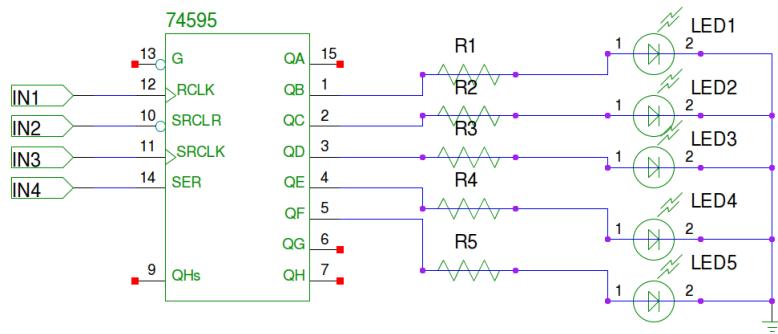


Figura 2.5: Sensore di temperatura DHT11

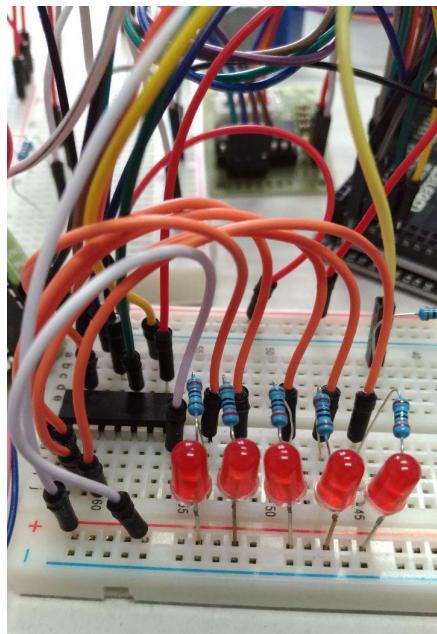
Alcuni valori telemetrici richiesti riguardano la temperatura all'interno dei vari compartimenti della NI-ICU. Come sensore di misurazione è stato utilizzato un DHT11 (Fig. 2.5) [28], un rilevatore di temperatura e umidità digitale con una risoluzione a 16-bit con un range di accuratezza pari a  $\pm 2$  gradi Celsius e come input un voltaggio tra 3.3V e 5V circa. I pin necessari al funzionamento sono un  $V_{cc}$  da 5V, un pin per la terra ed uno digitale connesso all'ATMEGA2560.

### 2.6.6 LED

La Calibration Unit è l'unità di calibrazione presente nella DAS che utilizza dei LED ad infrarossi. L'emulazione di questa componente è stata realizzata implementando un circuito contenente LED (Fig. 2.6) di colorazione rosso ed uno shift register SN74HC595 [29]. Il modello SN74HC595 è uno shift register di tipo ad 8-bit Serial input Parallel Output (SIPO) che opera ad un voltaggio compreso tra 2V e 6V. Il suo scopo è quindi quello di prendere come input fino ad 8 bit in seriale e parallelizzarli ognuno in uno



(a) Schema elettrico



(b) Implementazione nel sistema

Figura 2.6: Unità di Calibrazione

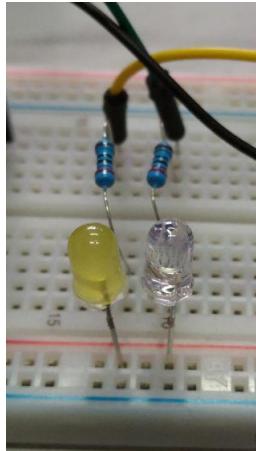


Figura 2.7: LED che simulano il comportamento dei riscaldatori

degli 8 output presenti; a tal scopo la sua connessione con il microcontrollore necessita di 4 pin: per la sincronizzazione del clock, per l'invio della sequenza di bit, per abilitare o meno l'output del dato appena ricevuto e per determinare la modulazione dell'impulso (PWM). Inoltre, oltre al GND sono presenti altri due pin per aumentare il voltaggio in ingresso ( $V_{cc}$ ) e per connettere più shift register in serie.

La tecnica dei LED (Fig. 2.7) gestiti con la PWM è stata usata anche per riprodurre l'unità che controlla il riscaldamento dei comparti della NI-ICU. Quindi nel caso in cui l'Application Software invii un comando per aumentare la temperatura, l'intensità luminosa dei LED varierà in proporzione al valore ricevuto.

## 2.7 Protocollo SPI

Il Serial Peripheral Interface [30] bus è un protocollo seriale sincrono con architettura master-slave. E' stato progettato da Motorola all'inizio degli anni '80 e utilizzato soprattutto per comunicazioni embedded tra microcontrollori a breve distanza o per connettere un microcontrollore con un dispositivo periferico come schermi, chip di memoria, sensori, CMOS, ecc. Le sue caratteristiche principali sono: è di tipo full-duplex con una banda che varia tra i

Modalità SPI	CPOL	CPHA	INVIO DATO
0	0	0	basso → alto
1	0	1	alto → basso
2	1	0	alto → basso
3	1	1	basso → alto

Tabella 2.2: Modalità SPI per l’invio di dati

20 e i 100 Mb/s. Dato che è sincrono utilizza un solo canale per il clock. Dal punto di vista architettonale per una connessione full-duplex master-slave oltre al clock il protocollo richiede un collegamento di output chiamato MISO (Master Input Slave Output) ed uno di input che prende il nome di MOSI (Master Output Slave Input). E’ inoltre presente una connessione di Slave Select (SS) per “attivare” l’ascolto dello slave. La configurazione del protocollo, oltre alla frequenza di clock, necessita di stabilire il *timing diagram*; esso stabilisce il comportamento della Clock Polarity (CPOL) e Clock Phase (CPHA). Nel primo caso se la polarità è impostata a 0 lo stato del protocollo è *idle*, se 1 vi è comunicazione tra master e slave. Se la trasmissione è attiva deve essere stabilita la fase della frequenza, ossia determinare in quale periodo del clock inviare il dato e in quale leggere la risposta (tab. 2.2).

Quando il master vuole stabilire una connessione con uno slave esso prima configura il protocollo, poi invia un segnale attraverso il canale SS per “attivare” il dispositivo slave. Dato che si tratta di una comunicazione full-duplex, ad ogni invio di una sequenza di dati tramite il canale MOSI, corrisponde sempre una ricezione nella linea MISO.

Per essere conformi alle specifiche rilasciate dalla missione Euclid, il Raspberry Pi avrà il ruolo di master, mentre l’ATMEGA2560 quello di slave.

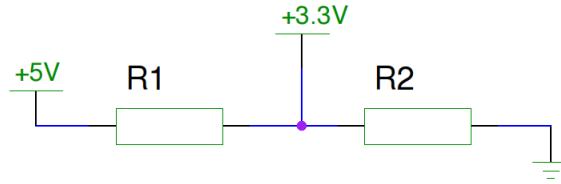


Figura 2.8: Schema elettrico del *resistor divider*

## 2.8 Voltage divider

In elettronica con *Voltage Divider* si intende un circuito passivo lineare che produce una corrente in uscita ( $V_{out}$ ) minore di quella in entrata ( $V_{in}$ ) [31]. In particolare nel sistema prodotto è stato realizzato un *Resistor Divider* (Fig. 2.8), ossia il più comune tipo di Voltage divider formato da due resistenze collegate in serie dove la resistenza  $R_1$  ha come input la corrente in entrata e come output sia il  $V_{out}$  che la resistenza  $R_2$  la quale poi è connessa alla terra. Questo metodo è stato utilizzato per il canale MISO del protocollo SPI dato che il master ha come tensione richiesta 3.3V, mentre lo slave ha un output di 5V.

$$V_{out} = V_{in} \frac{R_2}{R_1 + R_2} \quad (2.1)$$

Dalla formula 2.1 è possibile definire la tensione in uscita sapendo il valore di quella in entrata e delle due resistenze. Applicando la formula inversa e sapendo che le due resistenze sono direttamente proporzionali l'una dall'altra, si ha che la  $V_{in}$  assume come valore 5V, la  $V_{out}$  3.3V e assunto che la resistenza  $R_1$  sia  $120\Omega$ ,  $R_2$  sarà uguale a  $220\Omega$ .

## 2.9 Pulse Width Modulation

La Pulse Width Modulation (PWM) [32] è una tecnica usata per codificare un segnale digitale ed il suo principale uso riguarda il controllo dei dispositivi elettrici. L'idea di base di questa tecnica è quella di ottenere una tensione media (*duty cycle*) che deriva dal rapporto tra la durata dell'impulso positivo e quello negativo. Il duty cycle infatti è il risultato del

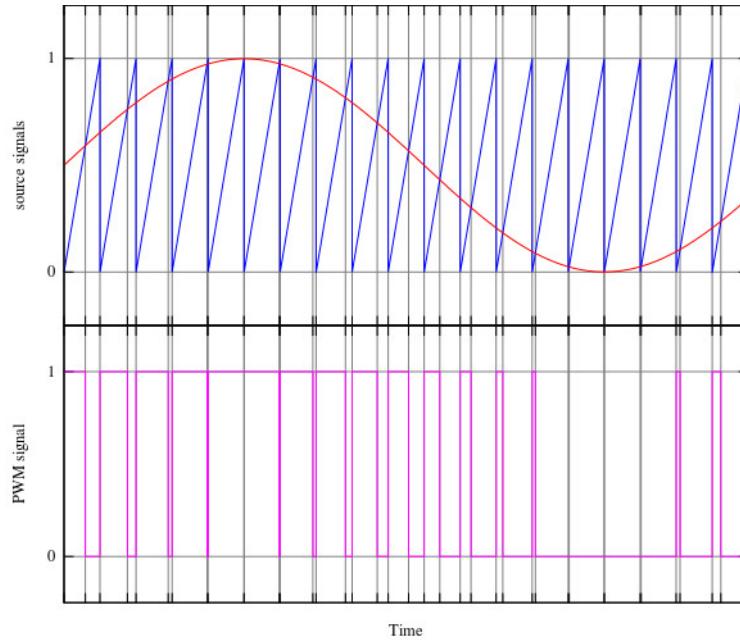


Figura 2.9: Esempio di Pulse Width Modulation

periodo di tempo che il segnale necessita per effettuare un ciclo on-off o viceversa; esso può essere espresso in percentuale o in base al rapporto di potenza (Fig. 2.9) che nel caso di un ATMEGA2560 corrisponde ad un valore compreso nell’intervallo [0, 255].

### 2.9.1 Implementazione di SPI in Raspberry Pi

BCM2835 [23] ha una sola interfaccia SPI (Fig. 2.10) connessa con il resto delle componenti del chip attraverso una *APB interface*. Esso è in grado di inviare e ricevere sequenze di dati lunghe fino a 16 bit. L’interfaccia prevede due buffer FIFO uno per i dati in entrata e l’altro per quelli in uscita, connesse all’interfaccia APB, un protocollo low-power che permette di accedere ai registri programmabili situati nei sub moduli periferici del dispositivo. Grazie al Clock Divider register inoltre è possibile determinare la frequenza di clock, mentre con il modulo FSM i dati vengono serializzati e trattati in base alla modalità SPI scelta (tab. 2.2).

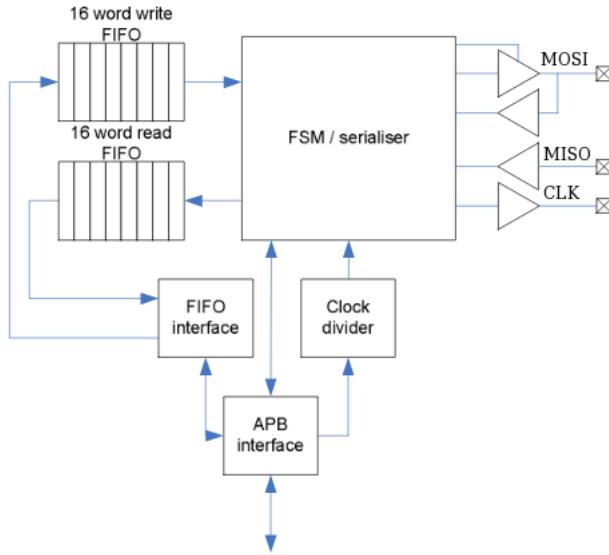


Figura 2.10: Schema SPI presente nel chip BCM2835

Nel momento in cui si instaura una trasmissione sempre attraverso l'APB si configurano il Clock Divider register e il modulo Serializer il quale attiva il canale di clock ed invia un bit 0 tramite il canale SS allo slave con cui vuole comunicare. A questo punto l'Advanced Process Bus riceve la sequenza di dati che devono essere spediti, li trasmette all'interfaccia FIFO che li serializza nel buffer prima di venire spediti sul canale MOSI. Contemporaneamente all'invio la connessione MISO riceve i dati provenienti dallo slave, questi passano per il rispettivo buffer di ricezione e sono gestiti dall'interfaccia FIFO prima di essere spediti dall'APB ad un'altra unità del chip.

### 2.9.2 Implementazione di SPI in Arduino Mega

ATMEGA2560 [24] comprende una interfaccia SPI con un buffer di 8-bit che può funzionare sia da master che da slave. Nel caso in cui il dispositivo si comporta come slave la comunicazione SPI (Fig. 2.11) può avere inizio solo quando il master lo abilita attraverso il canale SS. Quando un byte è ricevuto e quindi la comunicazione viene chiusa, il dato che prima è ricevuto su uno shift register viene spostato nell'SPDR, ossia un registro che comunica

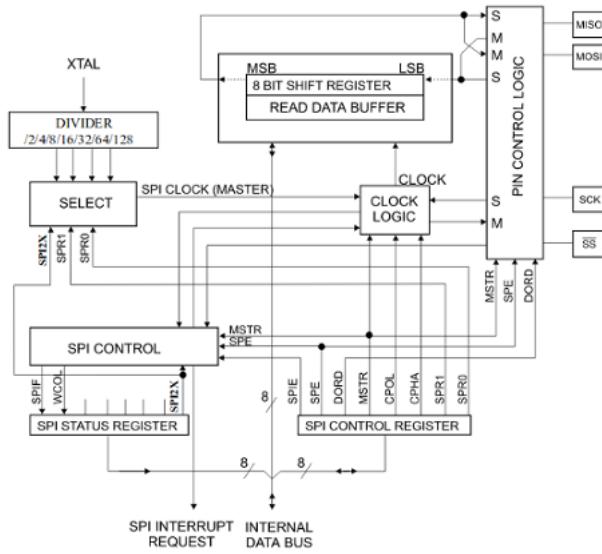


Figura 2.11: Schema SPI presente nel chip ATMEGA2560

direttamente con un bus interno; completata la ricezione, l'hardware imposta un flag (SPIF) nello Status Register che stabilisce la fine della trasmissione. Se il dispositivo viene configurato come slave impostando il flag SPE presente nell'SPCR (SPI Control Register), affinché un dato sia spedito deve essere salvato direttamente nello shift register, ma non viene effettivamente inviato. La spedizione avviene solamente quando è il master ad inviare un messaggio allo slave. Il dato proveniente dalla linea MOSI è salvato in seriale all'interno dello shift register e nello stesso momento, con uno *shift out* nel canale MISO, il dato precedentemente salvato viene spedito.

La configurazione del protocollo avviene modificando i bit dell'SPI Control Register: esso è un registro a 8 bit dove i primi due bit, nel caso in cui l'interfaccia operi da master, definiscono il *Clock rate*; i bit 2 e 3 configurano la polarità e la fase della trasmissione; i bit 4, 5 e 6 invece definiscono rispettivamente se l'interfaccia opera da master o slave, se l'ordine del byte è MSB o LSB e se l'interfaccia SPI è abilitata o meno; infine il bit 7 è utilizzato per stabilire se gli interrupt sono attivi.

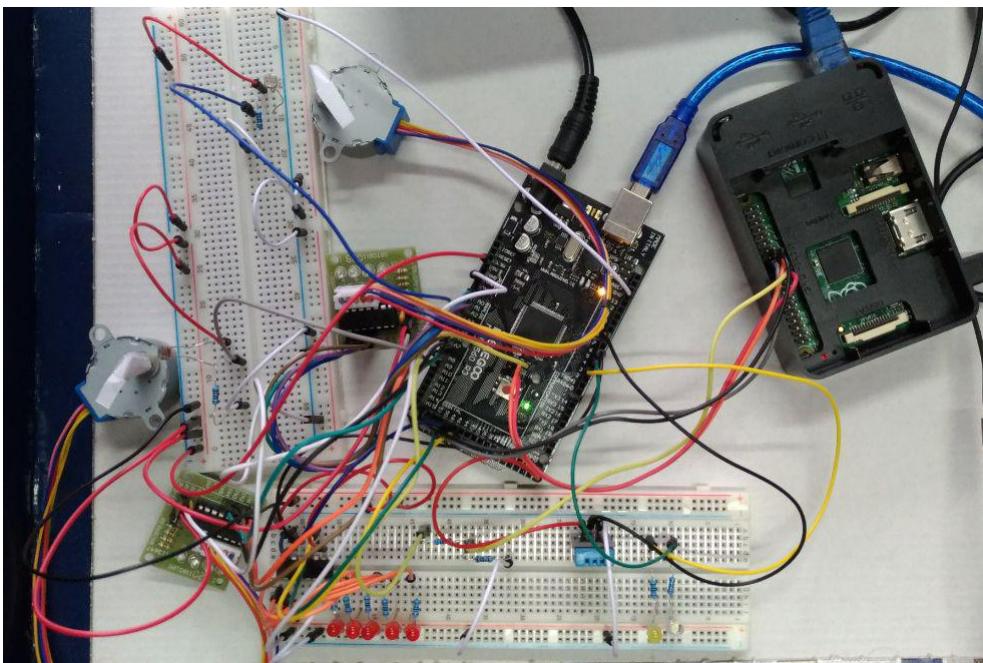


Figura 2.12: Panoramica del dimostratore implementato

## 2.10 Architettura hardware del dimostratore

Avendo come riferimento la struttura architettonica della NI-ICU (Fig. 1.2), il sistema è stato progettato per emulare il comportamento della NI-CDPU con l'uso del Raspberry Pi B+ e della DAS con un ATMEGA2560; queste due unità comunicano attraverso l'interfaccia seriale SPI dove il modulo con sopra il BCM2835 funge da master, mentre l'ATMEGA2560 da slave (Fig. 2.12).

Inoltre all'ATMEGA2560 sono connesse come unità periferiche i seguenti dispositivi:

- 2 motori passo-passo 28BYJ-48 per emulare il comportamento del NI-FWA e NI-GWA, oltre a due foto resistori GL5528 per indicare la *home position*. Tutte queste periferiche sono connesse alla board attraverso pin analogici.
- Il funzionamento della Calibration Unit è stato riproposto con 5 LED

di colore rosso gestiti da uno shift-register SN74HC595. L'intensità di colore è gestita attraverso un canale PWM connesso allo shift register.

- Nelle unità NI-SA e NI-FPA sono presenti due riscaldatori per mantenere la temperatura delle unità stabili; a tal proposito il loro comportamento è stato simulato da due LED (di colore giallo e bianco) connessi a due canali PWM. Definita una scala, l'intensità luminosa sarà proporzionale alla potenza che sarebbe irrorata nei riscaldatori.
- Parte dei valori delle telemetrie calcolabili fanno riferimento alla temperatura presente nel NI-FPA e NI-SA. E' stato usato un sensore di temperatura digitale DHT11 al fine di emulare entrambe le rilevazioni.

## 2.11 Raspberry Pi: specifiche software

La libreria implementata nel Raspberry Pi può essere richiamata dalle API dell'Application Software [20] così da potersi interfacciare con l'intero sistema hardware. Queste API scritte in C++14 sono contenute in un wrapper collegato alle API C dell'ASW così da poterlo testare importando semplicemente la libreria.

Lo sviluppo e il test delle funzionalità della libreria sono avvenute in ambiente Linux con la versione kernel 4.4. La libreria quindi può essere divisa in due livelli: una parte che lavora in kernel space e una in user space; tutte le funzionalità che permettono la comunicazione SPI avvengono attraverso il modulo kernel *spidev*.

### 2.11.1 Controllo della Calibration Unit

L'unità di calibrazione connessa all'ATMEGA2560 è composta da 5 LED collegati ad una matrice che permette la calibrazione di al più un LED alla volta. L'Application Software quindi tramite l'interfaccia SPI invia messaggi al driver della CU con lo scopo di [20]:

- Inizializzare il driver con una delle configurazioni predefinite e impostare il lock del registro per evitare nuove inizializzazioni;
- Avviare la procedura di calibrazione impostando il valore riguardante l'attuale livello di corrente e inviando due messaggi: un messaggio per controllare il LED che si vuole calibrare e l'altro per definire il *duty cycle* compreso tra [0, 1024]. Ad esempio con un duty cycle pari a 512, il LED verrà calibrato al 50% della sua potenza massima;
- Stop della procedura di calibrazione e disconnectione del LED connesso;
- Reset completo del driver.

### 2.11.2 Gestione dei Riscaldatori

Il controllo dei due riscaldatori (NI-FPA e NI-SA) è necessario per mantenere stabile la temperatura nel comparto motore e in quello del Focal Plane Assembly all'interno del NISP. A questo scopo l'inizializzazione del relativo driver prevede la configurazione dell'*heater FPA* inviando un comando SPI che imposta il suo *duty cycle* iniziale a 0, mentre l'*heater SA* prevede una configurazione iniziale con un *duty cycle* pari alla metà della potenza. Come per l'unità di calibrazione il *duty cycle* assume un valore compreso nell'intervallo [0, 1024]. Allo stesso modo è inoltre possibile spegnere l'attuatore semplicemente impostando la sua potenza a 0.

### 2.11.3 Gestione dei motori passo-passo

Come detto la DAS è composta da due stepper motor, il NI-FWA e il NI-GWA ed il loro comportamento è regolato dall'invio tramite SPI di un profilo motore. Ogni profilo inviato corrisponde ad un determinato numero di *step* ed ogni step al suo interno è definito da un *micro-step* composto da 3 campi: un *micro-step time*, ossia un valore a 16 bit che esprime il tempo in microsecondi; il livello di corrente della fase A ed il livello di corrente della fase B. Questi ultimi due campi corrispondono ad un valore intero a 8 bit che varia

nell'intervallo [-128,127]. Rispetto all'hardware implementato, è il valore della fase A che è necessario al motore per eseguire un singolo microstep; in questo caso se il valore della fase A è pari a zero il passo del microstep è nullo, invece se la fase assume un valore negativo il motore effettuerà un numero di passi proporzionale alla grandezza assegnata in senso antiorario, altrimenti se positivo il numero di passi sarà in senso orario. L'angolo di rotazione del motore, come per le due precedenti componenti varia in base al valore della fase: ad esempio se a fase 0 non corrisponde nessun movimento a fase +64 il movimento sarà in senso orario di 180°.

Dal punto di vista funzionale quindi il software deve essere in grado di: inizializzare il driver, effettuare la procedura di reset, eseguire gli step che permettono l'esecuzione di un profilo motore e quelli per l'invio di un singolo micro-step; quando una delle due precedenti esecuzioni è avvenuta, si deve avviare la procedura detta di *de-energization*. In certi casi è necessario verificare la posizione di uno dei motori ed in questo caso il software, dato che i motori non sono in grado di ritornare nessun valore rispetto alla loro posizione, svolge la procedura di lettura del sensore che rileva se e quando il motore si trova nella *home position*.

La fase di inizializzazione come prima cosa prevede che il controller del motore sia configurato con uno dei settaggi predefiniti, poi imposta lo stato del driver come *safe*. Ciò significa che il driver motore è inizializzato e pronto ad eseguire un comando; gli altri due stati possibili sono *disable* e *on*, i quali rispettivamente corrispondono allo stato successivo alla procedura di reset e al momento di esecuzione di uno step. Lo stato *safe* può essere raggiunto anche a seguito della procedura di *de-energization*.

Il caricamento e l'esecuzione di un profilo invece si svolge prendendo come input una struttura creata nell'Application Software contenente tutti gli step presenti all'interno di un determinato profilo ed invia ogni microstep al controller che provvede ad eseguirlo. Per eseguire ogni singolo microstep si utilizza la procedura di *energization*; quest'ultima, per evitare eventuali danni al motore, esegue i seguenti passi:

- Verifica se lo stato del driver è *on*, in quel caso esegue la procedura di *de-energization*;
- Abilita il *bridge* attraverso l'apposito comando SPI, permettendo così di comunicare con il driver del motore;
- Invia prima il comando con l'ID del motore a cui si vuole riferire il profilo ed una volta connesso imposta le sue fasi a 0;
- A questo punto lo stato del driver passa da *safe* ad *on*;
- Vengono infine spediti due messaggi SPI contenenti il valore delle fasi del micro-step da eseguire.

Quando viene richiamata la procedura di *de-energization*, essa svolge i seguenti compiti: imposta i parametri delle fasi del motore ancora connesso a 0; disconnette il motore; disabilita il *bridge* ed imposta lo stato del motore come *safe*.

Per evitare che i motori vengano disconnessi quando vi è un livello di corrente maggiore di zero, il sistema offre la possibilità di abilitare una componente del driver che vigila questa operazione. Quindi se la componente è abilitata e viene effettuata una procedura non consona, allora essa verrà bloccata dal modulo che vigila lo status.

Il software attraverso il controller è in grado di valutare se un motore si trova nella posizione predefinita o meno. Questa operazione prevede che venga letto il sensore di luce relativo al motore ritornando come valore l'attuale livello di corrente. Se la corrente, data una certa soglia, corrisponde al valore rilevato nel momento in cui il motore si trovava nella *home position*, allora non viene effettuata nessuna operazione; altrimenti è possibile comandare al driver di far ricercare la posizione di default al motore. Quest'ultimo quindi, ruoterà fin quando non viene rilevata la *home position*. Al fine di svolgere questa procedura in sicurezza, sono necessari i seguenti passi: si connette il sensore del motore cui si vuole rilevare la posizione e si inizia la procedura di

movimento che porta il motore sulla posizione di default; a questo punto viene prima misurato l'effettivo livello di luce per poi disattivare e disconnettere il sensore dal motore.

Infine la procedura di reset ripristina tutti i valori del software: disconnette entrambi i motori, disconnette il modulo di bridge, abilita la protezione per scongiurare tensioni indesiderate e imposta lo stato del driver come *disable*.

#### 2.11.4 Calcolo delle telemetrie

L'Application Software ha lo scopo di gestire e controllare le misurazioni di tutti gli elementi presenti nella ICU. Per far questo all'interno dell'ASW sono presenti tre differenti buffer. Il primo, chiamato *TM\_BUFFER* ha lo scopo di immagazzinare tutti i comandi che verranno poi spediti al relativo driver presente nella DAS (*TM Manager*); questo buffer può contenere fino a 256 TeleCommand (TC) ed ogni record è lungo 4 byte. Il *TM\_DATA\_1* e il *TM\_DATA\_2* vengono utilizzati per immagazzinare le risposte provenienti dallo slave; a differenza del *TM\_BUFFER* questi possono contenere al più 256 valori di 16 bit di lunghezza.

Il modulo di acquisizione delle telemetrie acquisisce in modo digitale sia le rilevazioni effettuate con sensori analogici che digitali. I segnali che riceve corrispondono al voltaggio dei LED, le misurazioni della temperatura sia interna alla NI-ICU che dell'ambiente esterno, lo stato dei sensori di rilevazione della posizione dei motori e il valore relativo alla potenza irrorata dai riscaldatori.

La configurazione del modulo consiste nel determinare quali rilevazioni si vogliono effettuare riempiendo il buffer con i relativi comandi; inoltre si definisce il tipo di acquisizione: nel caso *one-shot* i comandi presenti nel buffer verrano spediti solo una volta, altrimenti se la modalità scelta è quella ciclica le stesse misurazioni verranno effettuate periodicamente. Sono infine impostati 4 valori che determinano il tempo (*settling\_time*) di attesa in base al tipo di misurazione (se analogica o digitale) che si vuole effettuare.

Definiti questi parametri, prima di inviare i comandi delle relative telemetrie da calcolare, si avvisa la DAS tramite un messaggio di inizializzazione. La procedura prevede che prima di mettersi in attesa delle risposte, vengano inviati tutti i comandi delle telemetrie che si intende calcolare seguiti da un messaggio di tipo EOT (*END OF TABLE*); ricevuto l'EOT il *TM Manager* elabora la richiesta e spedisce indietro i valori calcolati.

Per permettere la lettura di un precedente ciclo di rilevazioni (che sia *one-shot* o ciclico) ogni volta che il master riceve un EOT da parte dello slave, il buffer in cui vengono salvati i dati viene scambiato. Quindi se *TM\_DATA\_1* è il buffer attivo per salvare il corrente ciclo di misurazioni, esso verrà scambiato con *TM\_DATA\_2* prima che il master invia i nuovi comandi per il calcolo delle nuove telemetrie. La procedura di lettura delle telemetrie prevede infatti che il buffer corrente sia impostato come *in uso* così che qualsiasi tentativo di accesso viene bloccato. Analogamente, mentre il software decide di leggere le TM ricevute, la struttura che contiene il buffer viene bloccata per tutto il periodo di lettura evitando così che vengano fatte operazioni indesiderate.

Infine anche il modulo di acquisizione delle telemetrie prevede una procedura di reset dello stesso; quando eseguita, essa svuota tutti i buffer sia dalle vecchie rilevazioni che dai TC inseriti, inoltre imposta a *false* i flag che riguardano il blocco e l'uso dei buffer per la ricezione delle telemetrie.

## 2.12 ATMEGA2560: Specifiche software

Come anticipato, l'ATMEGA2560 è l'unità del sistema che contiene i driver in grado di gestire le unità periferiche ad esso connesse in base ai comandi pervenuti dall'Application Software. Il firmware presente in questo microcontrollore è interamente sviluppato in C++11 con l'ausilio della libreria Arduino.h.

### 2.12.1 Controllo della Calibration Unit

Il driver presente nell'unità di calibrazione ha lo scopo di interfacciarsi con il driver SPI dello slave e gestire i LED in base al comando ricevuto. Per far questo, prima ancora che il driver venga inizializzato dal master, appena dopo l'upload del firmware si avvia la configurazione automatica che permette la gestione della periferica. Dato che l'unità prevede l'uso di uno shift register [29] i pin necessari per abilitare la comunicazione con esso sono: un pin che gestisce il clock ed uno per l'invio dei dati in seriale, un *latch* pin che abilita l'output solo dopo che tutta la sequenza di bit è stata ricevuta ed un pin chiamato *Output Enable* (OE) che, se collegato ad un pin di tipo PWM, permette la gestione della luminosità dei LED.

Dopo che è stata eseguita l'inizializzazione a seguito del comando pervenuto dall'ASW, l'unità si prepara ad effettuare la calibrazione di uno dei 5 LED periferici collegati. Questa operazione detta *excitation* prevede che vengano ricevuti due messaggi: il primo stabilisce la sequenza di bit da inviare allo shift register relativa al LED da connettere, mentre il secondo contiene il *duty cycle* relativo al valore di calibrazione. L'ammontare del *duty cycle* è convertito rispetto al massimo valore che il microcontrollore può assegnare alla PWM e, per eseguire l'azione, viene attivato il latch pin. Dato che la calibrazione è mutuamente esclusiva, quindi è possibile avere al più un LED acceso e ogni qual volta che avviene la procedura di calibrazione lo shift register viene ripristinato. Inoltre quando il valore del *duty cycle* è scritto nel registro dedicato, viene stabilito il valore di corrente in mA da assegnare al LED; questo valore dato un ammontare di default è poi proporzionale rispetto al *duty cycle* ricevuto.

Infine nel caso in cui venga ricevuto il messaggio di reset, tutti i valori impostati saranno impostati a quelli di default. Il *duty cycle*, l'amperaggio di ogni LED e l'eventuale LED connesso saranno impostati a 0.

### 2.12.2 Gestione dei riscaldatori

Il controllo dei due riscaldatori presenti, il NI-FPA e il NI-SA, è compito specifico dell'*heater driver*. Esso è connesso al driver SPI ed è in grado di ricevere comandi provenienti dall'Application Software. Per simulare il loro comportamento il driver ha lo scopo di gestire due LED connessi a pin che ne permette la PWM.

Dopo che i due riscaldatori sono stati inizializzati è quindi possibile eseguire la procedura per impostare la potenza da irrorare. Come nel caso dell'unità di calibrazione saranno quindi ricevuti due messaggi: uno che seleziona il LED e l'altro che ne definisce l'intensità dell'impulso convertendo il valore ricevuto rispetto alla potenza massima del PWM presente nel microcontrollore. La procedura di reset infine imposta a zero la potenza dell'impulso ricevuto dal riscaldatore e non ne permette nessuna operazione di controllo prima di una nuova inizializzazione.

### 2.12.3 Gestione dei motori passo-passo

I due driver FWA e GWA presenti nel microcontrollore hanno lo scopo di controllare rispettivamente il NI-FWA ed il NI-GWA ed anche se divisi presentano le stesse caratteristiche di funzionamento; tutto ciò che sarà detto successivamente è applicato ad entrambi i motori passo-passo.

Il controllo di questi motori avviene attraverso l'utilizzo di quattro pin che possono essere connessi sia a canali digitali che analogici; questo perché per far eseguire uno step al motore è necessario passare sequenzialmente in input una delle 8 configurazioni presenti nella tab. 2.1. Quindi per poter far muovere il motore in senso orario l'ordine di invio delle sequenze sarà quello crescente, altrimenti nel caso si voglia effettuare uno spostamento in senso antiorario l'ordine di invio sarà decrescente. Per questi tipi di dispositivi quindi, a differenza di un normale DC motor, la velocità di rotazione varia in base all'intervallo di tempo che intercorre tra l'esecuzione di un passo e quello successivo.

Il comportamento dei driver relativi ai due motori sono regolati dalle procedure implementate nell'ASW; per questo motivo il loro scopo è quello di eseguire i comandi pervenuti dal master SPI. In particolare, dopo che il driver è stato inizializzato con una delle configurazioni prestabilite, esso è pronto a svolgere una rotazione o il reset del motore; eseguita la procedura di connessione, è compito del master inviare i parametri relativi ad ogni step che il driver deve svolgere. Le ultime due fasi pervenute sono inoltre mantenute dal driver così da essere fruibili dal *TM Manager*.

Come spiegato precedentemente, dato che non è possibile stabilire l'angolo di rotazione del motore rispetto ad una posizione di default, il sistema ne permette la rilevazione utilizzando un sensore di luminosità. Nel momento in cui il master richiede la lettura del sensore di posizione, esso ritorna un valore che, data una certa soglia, stabilisce se il motore si trova o meno nella *home position* e, nel caso si trovi fuori posizione, si esegue la procedura di reset: come termine di paragone, viene calcolato l'attuale stato di luce e ad ogni passo eseguito dallo stepper viene effettuata una nuova rilevazione; se il risultato dell'ultima misurazione risulta essere molto minore rispetto al valore rilevato prima dell'inizio della procedura, allora significa che il motore ha raggiunto la sua posizione di default e il processo termina.

#### 2.12.4 Gestione delle telemetrie

Il driver per il controllo delle telemetrie presente nell'ATMEGA2560 ha lo scopo di effettuare le misurazioni richieste dal master. La procedura prevede che, prima di soddisfare ogni singola richiesta da parte del master, esso deve ricevere l'intera tabella, che indica quali telemetrie calcolare, che verrà salvata nel *TM\_buffer*. Le misurazioni hanno inizio solo quando l'ultimo messaggio in entrata corrisponde a EOT (End Of Table). La lettura dei dati copre tutte le unità descritte precedentemente: il driver presente nello slave va a leggere il valore corrispondente alla fase A e B dei due motori, il livello di corrente di ogni LED e richiede una nuova rilevazione dell'*home sensor*.

Per quanto riguarda la rilevazione delle temperature sia nei vari compartimenti che all'esterno, è stato implementato un sensore di temperatura che emula il comportamento di quelli presenti nel NI-SA e nei comparti dove sono presenti i motori NI-FWA e NI-GWA. Da specifiche, ognuna di queste unità deve avere una temperatura che può variare in un intervallo prestabilito che è molto minore rispetto alla temperatura media presente sulla Terra. Per questo, ogni volta che nel dimostratore si effettua una rilevazione della temperatura esterna, essa viene prima convertita da gradi Celsius a gradi Kelvin, poi il dato espresso in Kelvin viene sommato ad un valore random che dà come risultato un valore della temperatura che cade nell'intervallo definito dalle specifiche. Ad esempio se un certo comparto della NI-ICU richiede una temperatura tra 100 e 140 K e il sensore implementato nel dimostratore calcola una temperatura esterna di 293 gradi Kelvin (ossia 20°C), per avere un dato telemetrico all'interno del precedente intervallo, verrà calcolato un valore random che, se sommato a 293 K, avrà come risultato un valore che cade tra 100 K e 140 K. Questo tipo di operazione è stata importante in fase di test del dispositivo implementato per verificare la corretta trasmissione di tutte le telemetrie dallo slave al master.

## 2.13 Implementazione del protocollo SPI

L'interfaccia seriale presente nel Backplane all'interno della ICU ha lo scopo di connettere i moduli CDPU e DAS. La comunicazione è necessaria affinché la CDPU possa impartire comandi che possono essere svolti dalla DAS board.

A tal fine nel sistema implementato è presente una interfaccia seriale che connette il Raspberry Pi e l'ATMEGA2560. In generale questa comunicazione permette al master di comandare le operazioni rispetto ai due motori, i sensori di posizione, i LED di calibrazione e i riscaldatori, ma anche di richiedere il calcolo delle telemetrie e di ottenere le rilevazioni come risposta da parte dello slave.

Anche se il bus di comunicazione, la frequenza di clock e la struttura del messaggio sono diversi rispetto al paradigma presente nella NI-ICU, è stato scelto di mantenere la stessa lunghezza del messaggio. Ogni volta che verrà spedito un comando quindi, esso avrà una lunghezza di 32 bit di cui: i primi 8 bit corrispondono all'ID dello slave, i secondi 8 bit corrispondono all'indirizzo del driver nello slave ed i restanti 16 bit contengono il dato corrispondente al comando relativo all'operazione che il master chiede di svolgere allo slave.

Il paradigma implementato nel dimostratore prevede che i messaggi possano essere di tipo standard o variabili. I messaggi standard hanno lo scopo di impartire dei comandi predefiniti e sono usati sia per configurare i driver presenti nello slave che per il controllo delle periferiche. Nei comandi variabili il dato di 16 bit che viene spedito può assumere un valore variabile compreso in un intervallo che dipende dallo scopo del driver a cui è rivolto. I messaggi variabili sono utilizzati per impostare i parametri dei driver che gestiscono i riscaldatori e l'unità di calibrazione.

Esistono inoltre altri tre tipi di messaggi:

- per comunicare i valori delle fasi A e B durante l'esecuzione di un profilo motore;
- per inviare le telemetrie richieste;
- per richiederne i dati come risposta.

I messaggi utilizzati per eseguire il profilo motore sono dello stesso tipo dei messaggi variabili, ma questi hanno la differenza di poter avere interi anche negativi nel campo da 16 bit dove è contenuto il dato.

Anche i messaggi inviati nel *TM\_buffer* per il calcolo delle telemetrie hanno una lunghezza di 32 bit di cui: 3 bit selezionano il tipo di rilevazione (analogica o digitale), 2 bit definiscono il *settling\_time*, ossia un valore prefissato che indica quanto attendere prima di effettuare una certa rilevazione, 4 bit definiscono l'identificativo del device hardware (DAS-ID), 7 bit sono l'indirizzo del registro SPI e gli ultimi 16 bit compongono il dato.

Per quanto riguarda infine i messaggi spediti dallo slave al master contenenti le risposte, essi hanno una lunghezza di 16 bit tutti che compongono il valore del dato richiesto.

### 2.13.1 Flusso di comunicazione dei messaggi

Come descritto precedentemente, i buffer SPI implementati a livello hardware in Raspberry Pi e ATMEGA2560 sono rispettivamente di 16 bit e 8 bit; inoltre se il BCM2835 offre due buffer FIFO per inviare e ricevere i messaggi, ATMEGA2560 utilizza uno shift register di tipo SIPO il quale, dopo aver ricevuto tutti i bit relativi al messaggio, sposta il dato in un altro registro per poi poterlo leggere. Nel caso della scrittura lo slave invia sequenzialmente i bit nell'SPDR, che saranno inviati (*shift-out*) solo quando perverrà un nuovo dato dal master.

Avendo il vincolo della dimensione del registro SPDR presente nello slave, i messaggi spediti e ricevuti devono essere partizionati prima di poterli spedire. Tutti i messaggi da 32 bit sono quindi divisi in 4 parti da 8 bit inviate sequenzialmente; lo slave implementa un buffer di ricezione in cui salva tutte le parti ricevute per poter poi ricostruire l'intero messaggio.

### 2.13.2 Modalità di spedizione dei messaggi

Avendo diversi tipi di messaggi, per mantenere la sincronizzazione tra master e slave richiesta dal protocollo, il sistema implementa quattro modalità di invio e ricezione dei messaggi: *Atomic*, *TM\_receive*, *TM\_send* e *Stepper* (Fig. 2.13).

La *Atomic* è la modalità principale che gestisce tutto il paradigma SPI implementato nel sistema. Essa infatti ha il compito di selezionare le altre modalità di gestione dei messaggi, ma anche di eseguire i comandi di tipo standard e variabile pervenuti dal master. Ciò comporta che questa modalità ha il compito di gestire tutti i comandi utili al controllo sia della Calibration Unit che dell'Heater Driver, ma anche la maggior parte dei comandi relativi

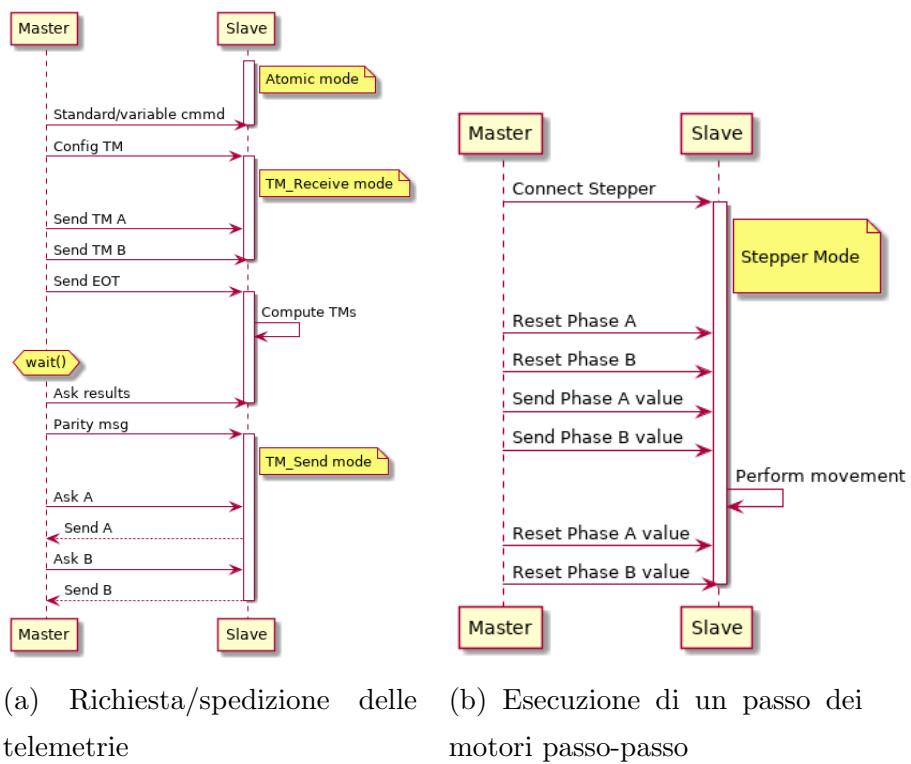


Figura 2.13: Sequence diagram del protocollo SPI implementato

al calcolo delle telemetrie e alla gestione dei motori. Per quanto riguarda i controller dei motori passo-passo, nel momento in cui avviene la connessione del motore, la procedura prevede la ricezione delle informazioni riguardo i micro-step da eseguire. La modalità *Stepper* si basa sullo stesso principio della precedente, ma si aspetta che la successiva coppia di messaggi contenga i parametri da assegnare alla fase A e B del motore selezionato; solamente quando sia la fase A che la fase B assumano come valore 0, allora viene riattivata la modalità atomica.

Per accedere alle modalità di gestione delle telemetrie, il driver SPI deve ricevere il messaggio che comunica l'intenzione di effettuare un ciclo di rilevazioni mentre si trova in *Atomic mode*; si passa quindi alla *TM\_receive*, la quale ha il compito di leggere e salvare nel *TM\_buffer* tutte le richieste di rilevazione. Solo quando il messaggio contiene un EOT il driver inizia la rilevazione e il riempimento del buffer di risposta ed imposta la modalità atomica in attesa che inizi la procedura di invio. Il master, dopo aver spedito l'ultimo messaggio, attende un certo tempo per far sì che lo slave svolga le precedenti operazioni. Il master attraverso un apposito messaggio SPI comunica la sua intenzione di ricevere le risposte attivando la modalità *TM\_send* nello slave.

Questa modalità prevede l'invio da parte dello slave di messaggi da 16 bit, ma questo può avvenire solo a seguito della ricezione di un nuovo messaggio dal master; in altre parole nella *TM\_send mode* si passa da una comunicazione considerata “half-duplex” a 32 bit (perché tutti i dati di risposta per gli altri comandi vengano ignorati dal master) a un “full-duplex” a 16 bit. Ciò comporta due problemi. Il primo riguarda la divisione del messaggio: il master per ricevere tutti i dati richiesti deve necessariamente inviare un numero di messaggi da 8 bit pari al doppio della dimensione del buffer appena spedito. L'altro problema riguarda la sincronia: se, subito dopo che lo slave riceve il messaggio che gli ordina di attivare la modalità *TM\_send*, il master richiede la prima parte di rilevazione che doveva essere calcolata, il risultato sarebbe un messaggio che non corrisponde all'esatto valore misurato, poiché lo slave non ha avuto ancora la possibilità di spedire all'SPDR il dato richiesto. Per

questo motivo il paradigma di comunicazione implementa un meccanismo per non perdere la sincronia chiamato *parity message*. Prima di richiedere la prima parte della prima rilevazione, il master invia un messaggio di 8 bit di cui ignorerà la risposta; alla ricezione però lo slave invia all'SPDR i primi 8 bit del primo record presente nel buffer di risposta. A questo punto dato che la sincronia è stata stabilita, al successivo invio di un messaggio dal master, esso può ricevere i dati richiesti.

Quando anche l'ultima telemetria è stata spedita al master, è ristabilita la modalità atomica. A causa del *parity message* la modalità atomica è fuori sincronia; per questo altri tre messaggi da 8 bit vuoti sono spediti dal master allo slave.



# Capitolo 3

## Analisi del dimostratore

Dopo aver mostrato le caratteristiche del prototipo che è stato sviluppato, in questo ultimo capitolo sarà analizzato ciò che è stato svolto anche in base a quelli che erano gli obiettivi preposti; saranno quindi messe in luce le limitazioni sostanziali di cui il sistema soffre, ma anche i risultati ottenuti, che potrebbero costituire la base per ulteriori sviluppi. A supporto della tesi iniziale, prima di capire quali potrebbero essere gli sviluppo futuri, si parlerà infine di un progetto che vede per la prima volta un Raspberry Pi nello spazio.

### 3.1 Vincoli Legali

Tutto ciò che è stato sviluppato in questa tesi sarà rilasciato sotto licenza GPLv3 per quel che riguarda il codice e Creative Commons per la documentazione degli esperimenti relativi al sistema. E' però importante notare che lo sviluppo di questo progetto si basa interamente su specifiche hardware e software [18, 19, 17, 20] relative all'implementazione delle componenti utilizzate all'interno della missione ESA-Euclid e quindi coperte da livelli di riservatezza più o meno stringenti. L'accesso a tali documenti è stato possibile solamente dopo che l'associazione con l'Istituto Nazionale di Fisica Nucleare

è stata approvata e accettando quindi di non divulgare questi materiali a terzi.

Per questi motivi, al fine di pubblicare il lavoro svolto sotto licenza libera, tutto ciò che è stato rilasciato trova fondamento per quel che riguarda le reali funzionalità delle componenti, ma si discosta rispetto a quelle che sono le caratteristiche specifiche.

### 3.2 Obiettivi preposti

La progettazione e lo sviluppo di questo sistema sono nati con l'idea di capire se fosse possibile introdurre nuove metodologie che possano sostituire o integrare quelle che ora costituiscono il corrente ciclo di vita di una missione; è infatti emerso che alcuni meccanismi adottati in certe situazioni possono risultare problematici o addirittura scomodi durante lo sviluppo del codice sorgente.

Prendendo in esame la missione ESA-Euclid è stato analizzato il problema dell'implementazione dell'Application Software per quel che riguarda il modulo NISP durante la Fase C. Lo scopo di questa fase è quello di sviluppare e testare dei prototipi che seguono le stesse specifiche dello spacecraft finale, ma che non sono adatte a resistere in un ambiente spaziale. L'implementazione di queste componenti, o il reperimento dell'hardware necessario, fa spesso riferimento a soluzioni che sono sì stabili, ma che necessitano spesso di un importante impegno economico; per questi motivi è stata acquistata una sola unità hardware con lo scopo di simulare il comportamento della NI-ICU che sarà presente nello spacecraft. Dato che i team di sviluppo del codice sorgente provengono da diverse aree geografiche, l'utilizzo del prototipo hardware presenta alcune criticità organizzative.

Questo lavoro di tesi mira a far riflettere sulla possibilità di sostituire, o quantomeno integrare, il ciclo di vita di una missione con l'introduzione di dispositivi che rispecchiano le specifiche proposte, ma che siano costituiti di componenti hardware e software che siano low-cost ed open-source. Questa

soluzione infatti permetterebbe ai team di sviluppare diverse unità su cui effettuare test pratici verosimili in attesa che il modulo certificato sia a disposizione ottenendo dei possibili vantaggi soprattutto per quel che riguarda le tempistiche preposte.

### 3.3 Analisi delle specifiche

Dato che lo stato dell'arte non offriva soluzioni da cui poter iniziare, vista gli obiettivi preposti si è scelto di prendere in considerazione hardware low-cost, possibilmente open-source, di largo consumo quindi sia facilmente reperibile che con una documentazione e con un supporto completo. Inoltre date le specifiche era necessario che i dispositivi dovessero possedere una potenza di calcolo almeno pari a quella originale e la possibilità di interagire attraverso il protocollo SPI; dal punto di vista economico invece, il costo totale delle componenti utilizzate per lo sviluppo è tale da permettere la possibile implementazione di numerose copie del sistema. Le componenti usate inoltre offrono tutte una documentazione completa e una enorme comunità a supporto degli sviluppatori. ATMEGA2560 e la board Raspberry Pi al di fuori del processore sono dispositivi OSH; Broadcom infatti ha rilasciato una vasta documentazione a riguardo ma non è possibile accedere a determinate informazioni se non dopo la firma di un Non-Disclosure Agreement (NDA). Infine tutte le unità che riguardano sensori ed attuatori non sono interessati dal concetto di Open Source Hardware (OSH) ma sono dispositivi facilmente reperibili e largamente utilizzati per moltissimi scopi.

Le funzionalità che questo dimostrativo simula corrispondono all'emulazione del comportamento della DAS rispetto ai comandi spediti dalla CDPU, quindi la gestione delle altre interfacce presenti nella NI-ICU, come il MIL-STD-1553 o l'RS232, non sono previsti. Analizzando nel dettaglio ciò che il sistema offre, esso presenta indubbiamente delle differenze, a volte anche notevoli, rispetto alle specifiche originali. Principalmente il fatto di utilizzare hardware molto diversi rispetto a quelli definiti nelle specifiche comporta una

sostanziale differenza soprattutto nei registri a disposizione per quel che riguarda l'implementazione dei driver utili alla gestione delle unità periferiche. Questa diversità si ripercuote necessariamente anche per quel che riguarda le modalità di comunicazione del protocollo SPI; sia la velocità di clock che la modalità di spedizione di tutti i tipi di messaggi potrebbe comportare problemi di sincronizzazione software. Allo stesso modo ogni volta che arriva un messaggio contenente un *duty cycle* o uno dei parametri per l'esecuzione di un *microstep* del motore, esso deve necessariamente essere convertito. Proprio a riguardo dei motori, essendo i 28BYJ-48 degli stepper unipolari, in base ai valori ricevuti dal master è stato necessario implementare un paradigma di esecuzione del passo analogo a quello specificato.

Oltre a quanto detto precedentemente, è altresì vero che l'implementazione di un sistema di questo tipo mostra che sarebbe possibile andare oltre a quelli che sono i limiti di SecoiaSim; il fatto che gli attuatori nel dispositivo reale siano simulati e pre-caricati nei registri o che i comandi impartiti dall'ASW siano solamente rilevati ma non gestiti adeguatamente, possono essere vincoli strutturali altrettanto forti. Inoltre nel dimostrativo è implementata una libreria in grado perfettamente di interagire attraverso un wrapper con le API dell'Application Software che regolano il funzionamento dell'hardware; ciò comporta che le limitazioni hardware sono state superate con l'implementazione di classi C++ che ne simulano il comportamento. Infine la facilità di sviluppo e la possibilità di ottenere risultati reali permettono che le fasi di test assumano dei connotati analoghi rispetto a dei valori che altrimenti sono completamente emulati.

### 3.4 Obiettivi raggiunti

Allo stato attuale il dimostrativo non è sicuramente in grado di prendere parte ad un iter con vincoli così restrittivi come quelli usati per la costruzione di uno spacecraft. La non possibilità di includere la libreria nell'ASW che attualmente è in sviluppo, non ha permesso di svolgere test accurati riguardo

aspetti fondamentali come potrebbero essere la velocità di clock, le scelte su come spedire un messaggio dal master allo slave o rispetto al tempo necessario per computare le azioni da parte dell'ATMEGA2560.

Ciò che è stato mostrato però è sicuramente sintomo che con l'enorme diffusione dei dispositivi System on Chip (SoC)<sup>1</sup> quelle che fino ad ora sono state procedure di sviluppo sicuramente funzionali, possono essere quanto meno integrate da dispositivi a basso costo che permetterebbero un ciclo di sviluppo meno problematico. Prendendo in esame il caso ESA-Euclid ad esempio, il fatto di avere un solo simulatore del modulo NI-ICU potrebbe risultare scomodo sia per la coordinazione tra team che per le fasi di sviluppo e test anche solo di parti di codice. Il sistema implementato ha però gettato le basi per rispondere ad interrogativi che fino ad ora, a causa anche delle limitazioni tecnologiche, non potevano essere posti; il solo fatto che, partendo dallo studio delle specifiche, è sorta l'idea su una possibile fattibilità di un metodo del genere, è sicuramente la prova che qualcosa di questo tipo è possibile. Analizzando le specifiche di SecoiaSim emerge che il tipo di hardware utilizzato, è importante che abbia dei requisiti alla base simili a quelli richiesti, ma esso non deve necessariamente essere analogo al modello originale; infatti neanche SecoiaSim possiede periferiche reali tantoché alla ricezione dei comandi non corrisponde nessuna azione se non la verifica che l'impulso sia pervenuto. Realizzare un prototipo funzionante, low-cost e open-source che segue, anche se con le dovute modifiche date anche dai vincoli legali, le specifiche di una missione spaziale è sicuramente il raggiungimento di un primo obiettivo che può essere un punto di partenza per continuare o modificare il percorso di sviluppo attuale. La scelta delle componenti sia hardware che software offre la possibilità di modulare il sistema ad-hoc rispetto le esigenze: l'ATMEGA2560 infatti è retro-compatibile in molte delle sue funzionalità con microcontrollori della sua stessa famiglia. Se si volesse quindi ad esempio ricreare una porzione del sistema che serva solo per implementare

---

<sup>1</sup>Un circuito integrato che presenta tutte le componenti di un computer o di altri sistemi elettronici.



Figura 3.1: Dispositivo AstroPi nella ISS

il comportamento dei motori, è possibile anche utilizzare un ATMEGA328p ed il porting del codice non necessita praticamente di nessuna modifica.

### 3.5 AstroPi

AstroPi [33] è un programma educativo che ha lo scopo di far avvicinare i giovani di una età compresa tra i 10 e i 18 anni, alla programmazione e alla computazione dei dati nell'ambito fisico ed in particolare nel contesto dell'esplorazione spaziale. Questo progetto è patrocinato principalmente dalla Raspberry Pi Foundation e dall'Agenzia Spaziale Europea (ESA) e prevede sfide di programmazione su Raspberry Pi B+. I vincitori hanno avuto la possibilità di vedere il codice da loro implementato funzionare all'interno di due Raspberry Pi B+ progettati per ambienti spaziali 3.1 e che quindi hanno quindi superato i test ambientali ESA, posti nella ISS. Il programma è diviso in 8 diversi obiettivi divisi per fascia di età che comprendono: la programmazione di un sensore di umidità in grado di rilevare se nelle sue prossimità vi è presenza umana; svolgere dei campionamenti all'interno della stazione spaziale così da poterla riprodurre sulla terra con il popolare gioco Minecraft; tracciare la posizione dell'ISS, rilevare sopra quale Stato esso si trova e disegnare la bandiera sulla matrice di pixel (Fig. 3.1); un dispositivo

di watchdog che misura temperatura, pressione e umidità ambientale ed è in grado di lanciare degli allarmi nel caso i valori siano fuori dall'intervallo atteso; implementare un dispositivo che svolga delle rilevazioni *near-infrared* su ampi range di foreste sulla terra per studiarne lo stato; progettare dei *reaction games* per testare i riflessi degli astronauti nel corso della loro missione; ed infine implementare un dispositivo che fosse in grado di rilevare le radiazioni presenti nella stazione spaziale.

Senza entrare nel dettaglio tecnico della questione, i risultati che sono emersi dai vari esperimenti sono stati pressoché tutti soddisfacenti soprattutto per quel che riguarda la prova di calcolare lo stato di salute delle foreste, tanto che è stato offerto, come supporto all'analisi, un set di dati dall'esperimento *Earth Observation Detective* della missione Principia [34], di cui anche AstroPi fa parte.

Al di là degli scopi educativi a cui questo programma è rivolto, è importante evidenziare questo progetto come ulteriore prova di come le tecnologie SoC low-cost e open-source stiano lentamente sviluppando una potenza ed una stabilità tale da poter funzionare ed essere utilizzati anche per fini minori nell'arco di missioni spaziali.

## 3.6 Sviluppi futuri

Il prototipo del sistema realizzato ha dimostrato la fattibilità di implementazione di un dispositivo che possa supportare il processo di sviluppo e test del codice sorgente utilizzando componenti low-cost modulari ed open-source.

A questo punto le strade da intraprendere sono due: continuare la ricerca utilizzando il setup esistente oppure cambiare approccio utilizzando altre unità che potrebbero avere delle performance superiori. Nel caso in cui sia la prima strada quella che si vuole seguire, il passo successivo potrebbe essere quello di testare le effettive capacità del sistema includendo la libreria sviluppata all'interno del codice sorgente dell'ASW e farlo girare nella distro

Linux (Raspbian Jessie) presente nel Raspberry Pi B+. Questo potrebbe essere il primo vero banco di prova per verificare se e come reagisce la libreria agli input dell'Application Software. A questo punto si potrebbe pensare di implementare funzionalità come la gestione degli errori, degli interrupt, un sistema di Watchdog o l'On Board Time, tutte funzionalità che l'hardware messo a disposizione supporta. Considerando inoltre questo setup, si può pensare di effettuare il porting del sistema operativo RTEMS su Raspberry Pi e proprio questo specifico topic è stato proposto alla Google summer school del 2016 ma necessita ancora di essere migliorato [35]; lo stato attuale di sviluppo è in grado di supportare i modelli A e B di Raspberry Pi e funzionalità come l'overclock, l'uso del clock interno del chip ARM e l'utilizzo della console attraverso la periferica UART0. Ciò che dovrebbe essere implementato a supporto del Board Support Package (BSP) sono l'accesso alla GPIO, l'implementazione del protocollo SPI e i driver relativi alle entrate USB. Ovviamente è possibile effettuare anche modifiche alle attuali periferiche utilizzate ed un esempio potrebbe essere quello di utilizzare motori passo-passo bipolari anziché unipolari.

Nel caso invece si decida di implementare un sistema completamente diverso ci sarebbero da porsi diversi quesiti a riguardo; si dovrebbe capire cosa cambiare nel setup attuale, cosa mantenere e soprattutto quali dispositivi adottare. Una possibilità a riguardo potrebbe essere mantenere una compatibilità tra il dispositivo che simula la DAS e ciò che dovrebbe emulare la CDPU; a tal proposito mantenendo invariato per quel che riguarda il setup dell'ATMEGA2560, si potrebbe pensare di implementare le librerie che interfacciano l'Application Software, o meglio ancora effettuare il porting di RTEMS, come firmware su un microcontrollore ATMEL AVR a 32-bit. Questa ultima soluzione inoltre sposerebbe appieno anche l'etica di poter implementare codice open-source su dispositivi open-source hardware.

# Conclusioni

L'implementazione del primo stadio di un sistema principalmente composto da dispositivi open-source ha mostrato che questo tipo di soluzione non è attualmente in grado di sostituire quelle procedure consolidate in uso. Come nel caso della missione Euclid, implementazione di sistemi di questo tipo potrebbe essere sicuramente di grande aiuto come supporto allo sviluppo nel caso in cui diversi team dislocati geograficamente, debbano condividere l'unica unità del modulo implementata secondo le specifiche. Anche se, per varie motivazioni che non sono solo di carattere tecnologico, lo stato dell'arte è totalmente privo di soluzioni di questo genere, un piccolo passo in questa direzione è stato fatto grazie al progetto a base educativa AstroPi.

In conclusione quindi, un prototipo che vede rispettivamente connessi attraverso un paradigma master/slave come SPI un Raspberry Pi ed un AT-MEGA2560, potrebbe non essere la soluzione ideale che potrebbe richiedere un mondo complesso come l'astrofisica, ma questo primo studio è sicuramente importante per stabilire le future strade da intraprendere. Sarebbe sicuramente interessante capire che tipo di comportamento potrebbe avere l'Application Software nel caso venisse connesso alla libreria che gestisce l'hardware sottostante, sia nel caso si utilizzi Raspbian sia dopo aver effettuato il porting di RTEMS su Raspberry Pi. Nel caso invece si pensasse di modificare lo stato fisico del sistema, sarebbe interessante capire, utilizzando ad esempio microcontrollori a 32-bit della famiglia AVR di ATMEL (effettuando magari anche in questo caso il porting del sistema operativo), come potrebbero cambiare le prestazioni del prototipo stesso.

L'avanzamento tecnologico nel campo embedded sta offrendo soluzioni che, soprattutto in altri ambiti di ricerca, stanno dando dei risultati molto soddisfacenti e come si è visto per la missione Breakthrough Starshot, anche l'astrofisica è coinvolta in questo progetto e si sta muovendo a piccoli passi in questa direzione.

# Bibliografia

- [1] Y Mellier. Euclid: Mapping the geometry of the dark universe. In *Science from the Next Generation Imaging and Spectroscopic Surveys*, volume 1, page 3, 2012.
- [2] MT Penny, Eamonn Kerins, N Rattenbury, J-P Beaulieu, AC Robin, S Mao, V Batista, S Calchi Novati, A Cassan, P Fouqué, et al. Exels: an exoplanet legacy science proposal for the esa euclid mission—i. cold exoplanets. *Monthly Notices of the Royal Astronomical Society*, page stt927, 2013.
- [3] Mark Cropper, S Pottinger, S-M Niemi, J Denniston, R Cole, M Szafrańiec, Y Mellier, M Berthé, J Martignac, C Cara, et al. Vis: the visible imager for euclid. In *SPIE Astronomical Telescopes+ Instrumentation*, pages 91430J–91430J. International Society for Optics and Photonics, 2014.
- [4] Thierry Maciaszek, Anne Ealet, Knud Jahnke, Eric Prieto, Rémi Barbier, Yannick Mellier, Florent Beaumont, William Bon, Anne Bonnefoi, Michael Carle, et al. Euclid near infrared spectrometer and photometer instrument concept and first test results obtained for different breadboards models at the end of phase c. In *SPIE Astronomical Telescopes+ Instrumentation*, pages 99040T–99040T. International Society for Optics and Photonics, 2016.

- [5] US Standard MIL STD. 1553b “aircraft internal time division command/response multiplex data bus”. *US Department of Defense, Washington DC*, 1978.
- [6] Helder Silva, José Sousa, Daniel Freitas, Sergio Faustino, Alexandre Constantino, and Manuel Coutinho. Rtems improvement-space qualification of rtems executive. *1st Simpósio de Informática-INFORUM, University of Lisbon*, 2009.
- [7] WJ Croft and John Gilmore. Bootstrap protocol. Technical report, 1985.
- [8] Sebastiano Ligori, Leonardo Corcione, Vito Capobianco, and Luca Valenziano. Design of the on-board application software for the instrument control unit of euclid-nisp. In *SPIE Astronomical Telescopes+ Instrumentation*, pages 914330–914330. International Society for Optics and Photonics, 2014.
- [9] Mario Merri, Bryan Melton, Serge Valera, and Andrew Parkes. The ecss packet utilization standard and its support tool. In *SpaceOps 2002 Conference*, page 06, 2002.
- [10] S Ligori, L Corcione, V Capobianco, D Bonino, G Sirri, F Fornari, F Giacomini, Laura Patrizii, Luca Valenziano, R Travaglini, et al. Detailed design and first tests of the application software for the instrument control unit of euclid-nisp. In *SPIE Astronomical Telescopes+ Instrumentation*, pages 99042Q–99042Q. International Society for Optics and Photonics, 2016.
- [11] Giuseppe D Racca, René Laureijs, Luca Stagnaro, Jean-Christophe Salvignol, José Lorenzo Alvarez, Gonzalo Saavedra Criado, Luis Gaspar Venancio, Alex Short, Paolo Strada, Tobias Bönke, et al. The euclid mission design. In *SPIE Astronomical Telescopes+ Instrumentation*, pages 99040O–99040O. International Society for Optics and Photonics, 2016.

- [12] Charles D Brown. *Elements of spacecraft design*. Aiaa, 2002.
- [13] Andrea Bonaccorsi and Cristina Rossi. Why open source software can succeed. *Research policy*, 32(7):1243–1258, 2003.
- [14] Adil Anis. Nasa open source softwares. <https://opensource.gsfc.nasa.gov/>. accesso il 08/06/2017.
- [15] Jonathan Lock. Open source hardware. Technical report, Technical Report, Department of Technology Management and Economics, Chalmers University of Technology, 2013.
- [16] Martin Elvis. The crisis in astrophysics and planetary science: How commercial space and program design principles will let us escape. *arXiv preprint arXiv:1609.09428*, 2016.
- [17] Rafael Toledo. *SecoiaSim requirements specification document*.
- [18] Jose M Carnicer. *Serial Control Interface ASIC (SECOIA) Datasheet*.
- [19] CayetanoRaichs. *EUCLID NISP Instrument Control Unit (ICU), ICU HW-SW Interface Control Document (ICD)*.
- [20] Santiago Carmona Meco Imanol Viana. *NISP-ICU DSW Requirement/Architectural Design Document*.
- [21] Jiri Gaisler. The leon processor user's manual. *Gaisler research*, 2001.
- [22] Adafruit Learning System. Introducing the raspberry pi model b+. <https://cdn-learn.adafruit.com/downloads/pdf/introducing-the-raspberry-pi-model-b-plus-plus-differences-vs-model-b.pdf>. accesso il 09/06/2017.
- [23] *BCM2835 ARM Peripherals*. Broadcom Europe Ltd., 2012.
- [24] *Atmel ATmega640/V-1280/V-1281/V-2560/V-2561/V*. Atmel.

- [25] *BYJ series Geared Step Motor 28BYJ48-12-300-01*. Changzhou Fulling Motor Co.
- [26] *HIGH-VOLTAGE, HIGH-CURRENT, DARLINGTON TRANSISTOR ARRAYS*. Texas Instruments, .
- [27] *CdS Photoconductive cells GL5528*. SparkFun Electronics.
- [28] *DHT11 Product Manual*. AOSONG.
- [29] *SNx4HC595 8-Bit Shift Registers With 3-State Output Registers*. Texas Instruments, .
- [30] Louis E Frenzel. *Handbook of serial communications interfaces: a comprehensive compendium of serial digital input/output (I/O) standards*. Newnes, 2015.
- [31] Wikipedia. Voltage divider. [https://en.wikipedia.org/wiki/Voltage\\_divider](https://en.wikipedia.org/wiki/Voltage_divider). accesso il 18/06/2017.
- [32] Jonathan Valvano. *Embedded Systems (Introduction to Armxae Cortexu2122-M Microcontrollers), Fifth edition*. Jonathan Valvano, 2014.
- [33] David Honess and Oliver Quinlan. Astro pi: Running your code aboard the international space station. *Acta Astronautica*, 2017.
- [34] BH See, R Morris, S Gorard, and N Griffiths. Uk space agency principia education programme report: the reach and spread of its projects., 2017.
- [35] Sebastian Huber. Pagina github di rtems per raspberrypi. <https://github.com/RTEMS/rtems/tree/master/c/src/lib/libbsp/arm/raspberrypi>. accesso il 20/06/2017.

# Ringraziamenti

Ai miei genitori, che con i loro valori ed il loro supporto sono sempre stati dalla mia parte permettendomi di raggiungere questo traguardo. Anche se scontato, *grazie*.

Ai miei nonni, chi c'è ancora e chi non c'è più, *grazie*, perché con i vostri insegnamenti risiederete sempre dentro di me.

Ai due pilastri, il prof. Renzo Davoli e i Dott. Francesco Giacomini, che mi hanno sempre supportato in questo lavoro di tesi. *Grazie*.

A Sheri, che è stata la chiave che ha aperto il mio essere. *Grazie*.

Ad Ilenia, che mi ha donato l'immensa gioia della semplicità. *Grazie*.

Ad Antonio, Federico, Gianluca, Giovanni Modica, Giovanni Pisana, Simone e Stefano, coloro con cui ho condiviso e continuerò a condividere molti momenti importanti della mia vita. *Grazie*.

Agli amici di una vita, coloro che, anche se il destino ha diviso le nostre strade, ci sono sempre. *Grazie*.

Al Crohn, che mi ha fatto capire che l'importanza di combattere. *Grazie*.

Alla vita, che mi fa essere sempre positivo. *Namasté*.