IBF

Istituto di Biofisica

CNR – Consiglio Nazionale delle Ricerche

Implementazione di un dispositivo universale, mobile, per la calibrazione di strumenti per Atomic Force Spectroscopy – AFM – su progetto BIOJERKER

Ing. Francesco Impallari

17/06/2016

INDICE

Elenco Figure

Elenco Tabelle

Introduzione

Il progetto, sviluppato presso l'Istituto di Biofisica (IBF) di Palermo del C.N.R., ha per oggetto quello di stabilire l'architettura software ottimale per ottenere uno strumento portatile, integrabile in sistemi di Microscopia a Forza Atomica (AFM, dall'inglese Atomic Force Microscope) esistenti come plugin esterno, adottando il Raspberry Pi.
L'AFM è un microscopio a scansione di sonda dove viene utilizzata una sonda microscopica per rivelare le proprietà di una superficie su scala nanometrica.
Il microscopio a forza atomica percepisce forze interatomiche che si instaurano tra la punta di una sonda e la superficie da esaminare. La punta è montata su una leva elastica che viene deflessa per effetto dell'interazione con la superficie. La deflessione viene misurata per mezzo di un dispositivo ottico composto da un laser e un fotodiodo a quattro quadranti.

Il sistema appena descritto ci permette di determinare le proprietà meccaniche della leva (cantilever) e quindi della costante elastica k, da una lettura diretta del segnale di deflessione.

L'implementazione software prevederà sia la progettazione di una interfaccia grafica user-friendly per la visualizzazione dei dati, sia la programmazione del sistema di acquisizione dati su hardware dedicato.
Al fine di rendere l'implementazione portatile e allo tempo stesso flessibile, verrà utilizzato il Raspberry Pi, un economico single board computer, messo in comunicazione con una scheda dedicata tramite protocollo seriale I^2C.
L'interfaccia, il calcolo numerico e la comunicazione verso l'utente e verso la scheda di acquisizione saranno gestiti dall'unità centrale (il Raspberry Pi)

sulla base di protocolli aperti. L'interfaccia grafica sarà sviluppata in javascript; la parte di calcolo numerico, di comunicazione verso l'utente e verso la scheda di acquisizione sarà codificata in Python. L'interfaccia utente dovrà girare su un client connesso via wi-fi in modo da non sovraccaricare il Raspberry Pi e rendere ancora più flessibile l'architettura dello strumento. L'utente userà l'interfaccia per trovare i parametri che adattano la curva teorica allo spettro di potenza del segnale proveniente dall'AFM. Per fare ciò egli potrà fornire o aggiustare i parametri iniziali di fitting e/o selezionare la parte di spettro da utilizzare per l'analisi. L'ausilio della visualizzazione grafica dei dati è fondamentale per potere giudicare la bontà dell'analisi effettuata e del parametro di calibrazione così ottenuto.
La comunità scientifica ha abbracciato con entusiasmo questo linguaggio interpretato arricchendolo di libreria veloci ed efficienti che facilitano notevolmente lo sviluppo di progetti di questo tipo.
Le potenzialità di calcolo del Raspberry permetteranno il calcolo e la visualizzazione dello spettro del segnale in tempo reale, nonché il calcolo dei parametri di calibrazione secondo i protocolli accettati dalla comunità scientifica.
Il seguente lavoro viene realizzato seguendo il ciclo di vita di un software, usando UML (Unified Modeling Language), un linguaggio di modellazione, semiformale e grafico, per descrivere il sistema e le interazioni richieste.

Nel *primo capitolo*, descriviamo la Microscopia a Forza Atomica (AFM), il suo funzionamento e come si determinano le proprietà meccaniche del cantilever ed in particolare della costante elastica k. Questo capitolo, puramente teorico, serve ad illustrare la tecnica AFM e le sue applicazioni ancor prima di progettare l'architettura ottimale del sistema da realizzare.

Nel *secondo capitolo*, vengono descritti i componenti hardware scelti per la realizzazione del sistema.

Il *terzo capitolo* tratta la scelta del sistema operativo per il Raspberry Pi e delle librerie adottate per il funzionamento del sistema.

Nel *quarto capitolo* si introducono gli strumenti di sviluppo e i linguaggi di programmazione del software: quelli utilizzati per il lato Client e per il lato Server.

Nel *quinto capitolo*, si descrive il caso di studio del sistema e quindi viene svolta un'analisi dei requisiti e di specifiche tecniche. Vengono introdotti *l'Use Case*, *l'Activity Diagram* e il *Sequence Diagram*, per descrivere meglio il sistema e le sue interazioni, sfruttando UML. Infine, viene anche descritta l'architettura del sistema/applicazione.

Il *sesto capitolo* è quello dell'implementazione, dove vengono illustrati e spiegati i passi più importanti del codice.

Il *settimo capitolo*, sarà dedicato interamente alla presentazione dei risultati ottenuti, mostrando con degli screenshot delle schermate dove la GUI si comporta esattamente come previsto nelle fasi di analisi e di progettazione.

1. Microscopia a Forza Atomica

1.1 Introduzione

Nel 1981 gli scienziati svizzeri Gerd Binning e Heinrich Rohrer [1] misero a punto un apparato sperimentale semplice ed efficace con l'obiettivo di analizzare la conduttività delle superfici. Questo dispositivo prese il nome di microscopio a scansione tunnel o semplicemente STM (Scanning Tunneling Microscope), considerato il capostipite di tutti i microscopi a scansione di sonda sviluppati in seguito. Il principio di funzionamento di questo dispositivo è puramente quantistico e consiste nell'attraversamento di una barriera di potenziale da parte di una particella. Considerando degli elettroni aventi energia totale E_0, racchiusi in un bacino di energia potenziale E_i, con $E_i > E_0$, appare impossibile che l'elettrone possa sfuggire da esso, in quanto il suo contenuto energetico è inferiore a quello della barriera. Dal punto di vista quantistico esiste però una probabilità finita, diversa da zero, che qualche particella attraversi la barriera (in disaccordo con la fisica classica) producendo una corrente denominata di *tunnel*. Considerando una punta acuminata e conduttrice in tungsteno, distante pochi nanometri dalla superficie del campione (anch'esso conduttore) e separati da uno strato isolante, ad esempio aria, l'energia potenziale posseduta dagli elettroni nella punta è minore che nell'isolante, quindi, affinché possano migrare da un conduttore ad un altro, occorre fornirgli energia cinetica (riscaldando il metallo) in accordo con la fisica classica. Secondo il modello quantistico se lo spessore dell'isolante è sufficientemente piccolo, si instaura un flusso di particelle attraverso la barriera di isolante e quindi una corrente di tunneling (la direzione verso la quale migrano gli elettroni dipende dal segno della tensione di bias

[1] G. Binnig, C.F. Quate, Ch. Gerber. Atomic Force Microscope. Physical Review Letters vol 56, n. 9, 3 Marzo 1986

$(1mV \div 1V)$). La probabilità che ciò avvenga si riduce all'aumentare della regione di spazio di attraversamento, secondo una legge del tipo:

$$I \propto \frac{V}{z} e^{\frac{z}{k}} \tag{1.1}$$

dove Iè la corrente espressa in [A], V è la differenza di potenziale applicata in [V], z è la larghezza della barriera in [nm] e k è un coefficiente dell'ordine di 10^{-10}m. La corrente dipende quindi linearmente dalla tensione ma esponenzialmente dalla distanza z. In principio, lo strumento non veniva adoperato in UHV (Ultra High Vacuum), eccetto in presenza di campioni biologici.

La presenza di contaminanti e di ossidanti sulle superfici dei campioni possono, in fase di misura, interferire con la corrente di tunneling e ciò può essere considerato un limite dei sistemi STM. Inoltre il loro uso è limitato alle superfici di campioni conduttivi. Un STM può essere adoperato per scansire un campione secondo due modalità di utilizzo: *altezza costante* e *corrente costante*. In altezza costante, la punta (o il campione) si muove lungo una direzione orizzontale mantenendo costante la sua altezza rispetto alla superficie del campione. La corrente misurata in ogni punto della superficie del campione, costituisce l'insieme di dati usati per realizzare l'immagine topografica dello stesso. In corrente costante, il sistema STM usa un sistema di feedback che permette di mantenere costante la corrente di tunnel regolando l'altezza da scansire. Per esempio, quando il sistema determina un incremento della corrente, esso regola la tensione da applicare allo scanner piezoelettrico, incrementando la distanza tra la punta e il campione. La variazione di quota prodotta dallo scanner per mantenere costante la corrente, costituisce l'insieme di dati usati per realizzare l'immagine topografica della superficie del campione. Dal confronto delle due modalità si evince che la tecnica ad altezza costante è la più rapida, in quanto non richiede la movimentazione dello scanner in direzione verticale

ma risulta performante solo per superfici relativamente lisce; la tecnica a corrente costante permette di misurare superfici irregolari con alta precisione pur richiedendo un maggior tempo di misura. In Figura 1 è mostrato uno schema delle parti che compongono la strumentazione STM.[2]

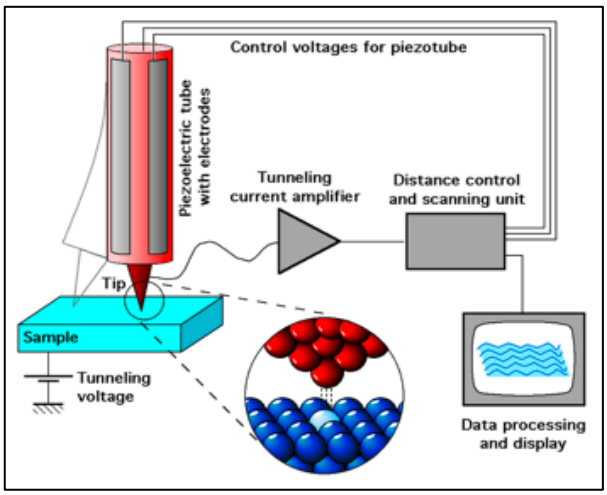

Figura 1: Schema di un STM

Nel 1986 Gerd Binning e la sua squadra ricevettero il premio Nobel per la fisica come riconoscimento per il contributo allo sviluppo di una nuova tecnica microscopica basata, come detto, sull'effetto tunnel. Questa tecnica ha ispirato la realizzazione di uno strumento denominato AFM il quale viene impiegato per lo studio, in scala atomica, delle superfici di varia natura: film sottili o spessi di materiale ceramico, materiali amorfi, vetri, membrane sintetiche o biologiche, metalli, polimeri, semiconduttori. Il microscopio a forza atomica o microscopio a scansione di forza SFM (Scanner Force Measurement) è in grado di operare in un mezzo quale aria o liquido e quindi in presenza di campioni biologici e in UHV (Ultra High Vacuum). Permette inoltre una risoluzione compresa in un range di 0.01 ÷ 1 nm lungo x e y mentre di 0.1 nm lungo z (indica la direzione ortogonale alla superficie).

[2] Microscopio effetto tunnel. Wikipedia.
https://it.wikipedia.org/wiki/Microscopio_a_effetto_tunnel

Il microscopio a forza atomica è un membro della famiglia dei microscopi generalmente indicata con l'acronimo di SPM (Scanning Probe Microscope) o microscopio a scansione di sonda, del quale fa parte il già noto STM e il microscopio ottico a scansione a *campo vicino* SNOM, etc. Ciò che accomuna tutti gli SPMs è l'utilizzo di una opportuna sonda appuntita, il *probe*. L'uso di una sonda permette di realizzare immagini topografiche del campione analizzato ad alta risoluzione (scala nano metrica), di sondare le forze di interazione tra superfici o molecole e di misurare l'interazione tra svariati tipi di materiale allorquando la sonda viene funzionalizzata chimicamente.

SPM	Segnale	Sonda	Risoluzione	Campioni	Ambiente
STM	corrente di tunneling	filo conduttore	1Å	conduttori solidi	vuoto, aria
AFM	forza	leva flessibile	1Å	ogni superficie	vuoto, aria, liquido
SNOM	flusso fotonico	guida d'onda affilata	100Å	ogni superficie	vuoto, aria, liquido
SICM	flusso ionico	micro-pipetta	200Å	ogni superficie	soluzione ionica
SECM	corrente faradica	micro elettrodo	1000Å	ogni superficie	soluzione ionica
SThM	flusso di calore	micro termocoppia	1000Å	ogni superficie	vuoto, aria
SMFM	forza magnetica	punta magnetizzata	100Å	superfici magnetiche	vuoto, aria

Tabella 1: Classificazione degli SPM

In Tabella 1 vengono mostrati i microscopi a scansione di sonda sviluppati nel corso degli anni, partendo dal capostipite ad effetto tunneling in funzione delle caratteristiche salienti di ogni dispositivo.

1.2 Introduzione all'AFM

Il microscopio a forza atomica (AFM) permette di produrre immagini 3D ad alta risoluzione delle superfici dei campioni analizzati, sia conduttori che isolanti su scala atomica. L'AFM misura forze molto piccole (meno di 1nN) che si instaurano tra la punta della sonda (usata per effettuare la misurazione) e la superficie di analisi. In figura 2 è mostrato uno schema semplificato, relativo al principio operativo di un AFM. La fase di misura può essere effettuata, in relazione al tipo di AFM, movimentando la sonda o il campione. La movimentazione della sonda può causare, in fase di misura, fenomeni indesiderati (rumore meccanico, armoniche meccaniche successive, ecc), mentre mantenendo fissa la sonda rispetto al campione, la determinazione dello spostamento relativo tra la punta del tastatore e la superficie di riferimento del campione non risente dei fenomeni citati in precedenza. Tuttavia, per effettuare misurazioni relative a grandi campioni, si preferisce movimentare la sonda mantenendo fisso il campione.

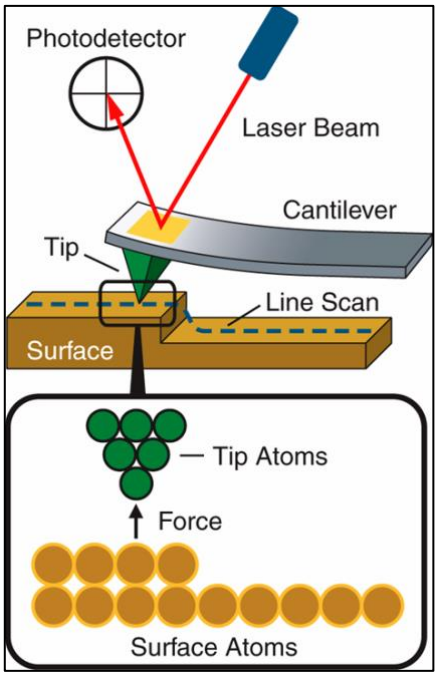

Figura 2: Schema semplificato di funzionamento dell'AFM

L'AFM può essere usato in modalità statica e in modalità dinamica. Nel primo caso, l'estremo libero della sonda, costituita da una punta acuminata, viene portato a contatto con la superficie del campione, si parla anche di *modalità per contatto*; durante il contatto iniziale, si instaura una forza repulsiva tra gli atomi della punta e del campione (causata dalla sovrapposizione degli orbitali elettronici) che provoca la deflessione della stessa; la deflessione dell'ordine del nanometro viene misurata da un sensore capacitore o da un rivelatore ottico. Ad esempio, nel caso di un sensore in silicio con costante elastica di 10N/m, a seguito di una forza di 0.2nN, si genererà una deflessione della leva di 0.02nm. Nella modalità di funzionamento definita *dinamica* o semplicemente *senza contatto*, la punta del sensore viene portata a pochi nanometri di distanza dalla superficie del campione, dunque il sensore viene posto in vibrazione. Anche in questo caso si generano delle forze di attrazione denominate di Van Der Waals, tra la sonda e il campione. Nella modalità dinamica il feedback è effettuato nell'ampiezza delle oscillazioni. Questa modalità viene preferita nel caso di

campioni *fragili* in quanto l'interazione con la punta è decisamente più contenuta.

Un AFM adoperato in modalità statica o per contatto, permette di ottenere immagini topografiche con una risoluzione verticale minore di 0.1nm e una risoluzione laterale di circa 0.2nm. Forze comprese tra 10nN÷1pN vengono misurate con una sensibilità allo spostamento di 0.01nm.

Il componente chiave di un dispositivo AFM è sicuramente la sonda. Per rilevare piccole forze (minori di 0.1nN) e ottenere alte risoluzioni verticali e laterali, è richiesta una costante elastica estremamente ridotta (1N/m) con un raggio di curvatura della punta dell'ordine dei 5÷50nm. Peraltro è desiderabile avere frequenze di risonanze comprese tra 10÷100kHz al fine di minimizzare la sensibilità alle frequenze strutturali che si attestano attorno ai 100Hz. La deflessione subita dalla leva, viene misurata da un fotorilevatore, sfruttando l'uso di una tecnica a deflessione del fascio laser incidente sulla sonda (figura 2).

1.2.1 Modalità operative

L'AFM può operare in varie configurazioni in base all'applicazione richiesta [3] (realizzazioni di immagini topografiche, rilevazione di proprietà locali del campione), in funzione delle condizioni operative, della tipologia e delle dimensioni del campione da esaminare. Ciò implica che l'interazione tra la punta della sonda con la superficie del campione può mutare, in relazione alle differenti configurazioni di utilizzo di un AFM. Pertanto si distinguono le seguenti modalità:

- modalità di funzionamento *statico* o in *contatto*;
- modalità di funzionamento *dinamico* o in *non contatto*;

[3] Bharat Bhusham. Handbook of nanotechnology. Springer, 2007. Pag. 591-608.

- modalità operativa detta *tapping*;

La modalità per contatto, figura 3, risulta la tecnica più utilizzata in AFM. La punta della sonda è portata a contatto con la superficie del campione e la loro distanza risulta inferiore alla dimensione media di un raggio atomico.

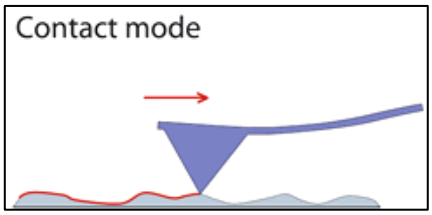

Figura 3: Modalità per contatto

Volendo acquisire un'immagine topografica in 3D, occorrerebbe movimentare il campione sottostante alla sonda (ammesso che la sonda sia ferma rispetto al campione) lungo le direzioni x e y. Se la distanza tra le due entità venisse mantenuta constante, la punta della sonda potrebbe non rilevare perfettamente il profilo del campione oppure potrebbe spezzarsi a seguito di una asperità superficiale elevata. La scansione ad *altezza costante* (figura 4a), quindi, viene usata qualora il campione abbia una rugosità superficiale molto piccola dell'ordine degli Angstrom. La tecnica di utilizzo ad altezza costante viene impiegata senza l'ausilio di un controllo di feedback, quindi vengono registrati in fase di misura le variazioni di altezza della sonda deflessa, proporzionali alla forza di interazione sonda-campione agenti punto per punto.

Questa tecnica viene impiegata laddove è essenziale possedere una elevata velocità di scansione, al fine di registrare in tempo reale il mutamento superficiale di un campione.

La modalità per contatto detta a *forza costante* (figura 4b) , prevede l'ausilio di un controllo di feedback agente lungo l'asse z (ortogonale alla

superficie), mediante il quale viene regolata la posizione del campione rispetto alla punta della sonda, mantenendo così costante la deflessione della sonda e la forza agente tra i due. La velocità di scansione, diversamente dal caso precedente, risulta limitata dal tempo di risposta del dispositivo di controllo.

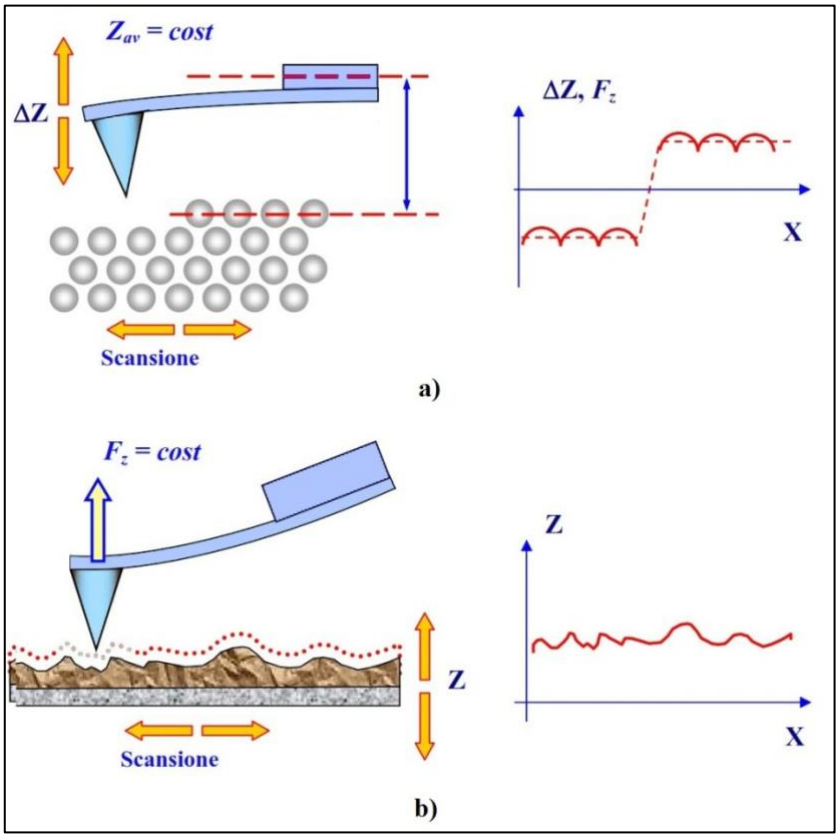

Figura 4: a) Modalità per contatto ad altezza costante; b) Modalità per contatto a forza costante.

La modalità di utilizzo senza contatto, figura 5a, prevede che la distanza tra la punta della sonda e la superficie del campione rimanga sempre compresa tra 10 e 100 nm. Le forze agenti che vengono misurate in questa modalità sono più deboli rispetto a quelle misurate nella modalità per contatto; per questo motivo, viene impressa alla sonda una oscillazione leggermente forzata (≥ frequenza di risonanza) e l'ampiezza dell'oscillazione viene

mantenuta costante mediante un dispositivo di controllo. Questa tecnica permette di ottenere basse risoluzioni, inoltre la presenza di sottili strati di fluidi (ad esempio l'acqua), può interferire con il moto oscillatorio della sonda.

La modalità di utilizzo denominata *tapping*, raffigurata in figura 5b, permette di produrre immagini topografiche di elevata risoluzione, anche di campioni le cui superfici possono essere facilmente danneggiate o non analizzabili con le modalità precedentemente trattate. L'ampiezza di oscillazione è generalmente di 5÷15nm e la frequenza con la quale vibra la sonda è compresa tra 50 e 500 kHz. Per eseguire la misurazione occorre che la sonda non sia inizialmente a contatto con la superficie del campione. Una volta portata ad oscillazione la sonda, viene accostata progressivamente al campione fino a quando non si innesca un contatto di natura ciclica. La traslazione del campione e il contatto ciclico permette di acquisire l'immagine della superficie posto che l'oscillazione della sonda venga mantenuta costate mediante un circuito di feedback. La velocità di scansione risulta più elevata se confrontata con la tecnica senza contatto ma più lenta rispetto alla tecnica con contatto.

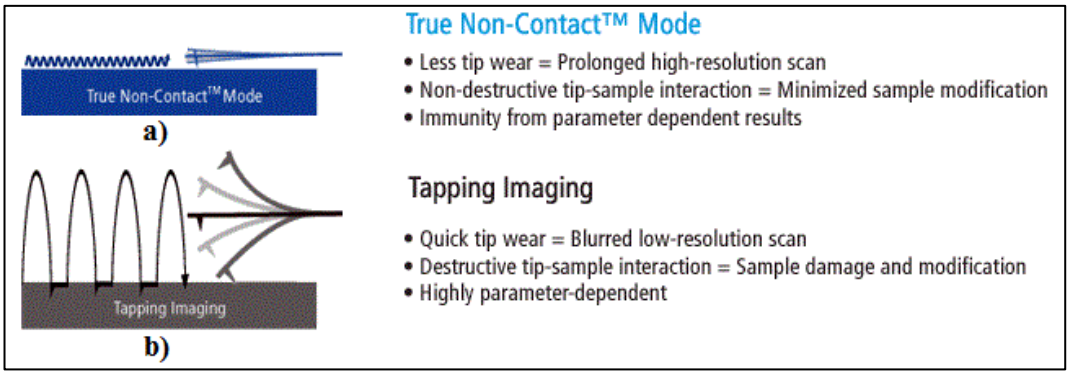

Figura 5: a) Modalità senza contatto; b) Modalità *tapping*

1.2.2 Forze di interazione tra sonda e campione

L'AFM permette la misurazione della forza di interazione che si instaura la punta della sonda e il campione[4]. La forza esercitata sulla punta dalla superficie produce la deflessione della sonda. L'interazione tra l'atomo più distante della punta della sonda e gli atomi della superficie, (entrambi considerati neutrali) può essere approssimata alle forze attrattive di Van Der Waals. Manifestandosi tra atomi di molecole diverse si parla di interazione non covalente o di non legame. L'origine di tale interazione va ricercata nell'interazione elettrostatica tra le nubi elettroniche e i nuclei degli atomi coinvolti, che è modificata dalla presenza degli atomi delle molecole vicine e dall'ambiente circostante. L'energia di interazione U(r) viene approssimata ricorrendo al potenziale semi-empirico di Lennard-Jones (equazione 1.2). Il primo termine descrive l'azione repulsiva prodotta dalle forze elettrostatiche dovuta al principio di esclusione del Pauli, mentre il secondo descrive l'azione attrattiva prodotta dalla polarizzazione dipolo-dipolo.

$$U(r) = 4\,\varepsilon \left(\frac{\sigma^{12}}{r^{12}} - \frac{\sigma^{6}}{r^{6}}\right) \tag{1.2}$$

Dove ε è la profondità della buca di potenziale (energia del legame del dimero), mentre σ è il diametro della sfera che approssima l'atomo o la molecola in un modello a sfera rigida. Il termine repulsivo prevale alle brevi distanze interatomiche ($r \ll \sigma$), mentre il termine attrattivo domina alle distanze maggiori ($r \gg \sigma$). In figura 6 è mostrato l'andamento del potenziale di Lennard-Jones relativamente alle forze che contribuiscono a deflettere la sonda.

[4] Victor L. Mironov. Fondamenti di microscopia a scansione di sonda. Accademia russa delle scienze, Istituto per la fisica delle microstrutture, 2004. (trad. it. [a cura di Giacomo Torzo]).

Nel paragrafo precedente sono state esposte le tecniche di impiego di un AFM. Ogni modalità è soggetta a differenti forze che si stabiliscono tra il campione e la punta della sonda le quali vengono impiegate per produrre immagini. Nella modalità per contatto prevalgono le forze di natura repulsiva come le forze di Van Der Waals e la forza elettrostatica (valore medio 10^{-9} N) prodotta a seguito della differenza di potenziale che localizza le cariche. Queste possono essere intrappolate da una elevata gamma di materiali, tra cui isolanti e semiconduttori. Nella maggior parte dei casi l'AFM lavora in atmosfera ambiente o con un sistema punta-campione immerso in un liquido con il vantaggio di potere osservare i campioni senza l'ausilio di pretrattamenti. Il liquido può ridurre l'accoppiamento meccanico che si instaura tra il campione e la sonda riducendone così i problemi di attrito.

La forza di attrito può causare la produzione di artefatti in fase di acquisizione dati, il danneggiamento della punta della sonda o della superficie del campione. Nel caso di osservazioni in atmosfera ambiente, la superficie del campione è ricoperta da atomi di azoto e da vapore acqueo. La punta portandosi a contatto con la superficie del campione, risente di un menisco, causa della tensione superficiale. Le forze di menisco e le altre forze di natura attrattiva possono essere neutralizzate operando con il sistema punta/campione immerso in liquido. Questa configurazione permette di eliminare le forze di capillarità e di ridurre le forze di Van Der Waals.

Nella modalità senza contatto le forze repulsive agenti sono sostanzialmente più deboli di quelle misurate in modalità per contatto. Le forze che prevalgono in maggiore misura sono quelle attrattive. Generalmente la produzione di immagini mediante la modalità senza contatto, può risultare inadeguata sia nel caso in cui il campione è avvolto da uno spessore di fluido, sia nel caso in cui la sonda si allontanata dalla superficie del

campione ad una distanza tale da non poter più risentire dell'effetto prodotto dalle forze di Van Der Waals.

La modalità dinamica denominata *tapping* permette di superare i problemi relativi all'azione prodotta dalle forze di attrito, di adesione ed elettrostatiche. Inoltre, tra la punta della sonda e la superficie del campione non si instaurano forze laterali di frizione che potrebbero danneggiare il campione, o rallentare la punta alterando i dati acquisiti in fase di misura.

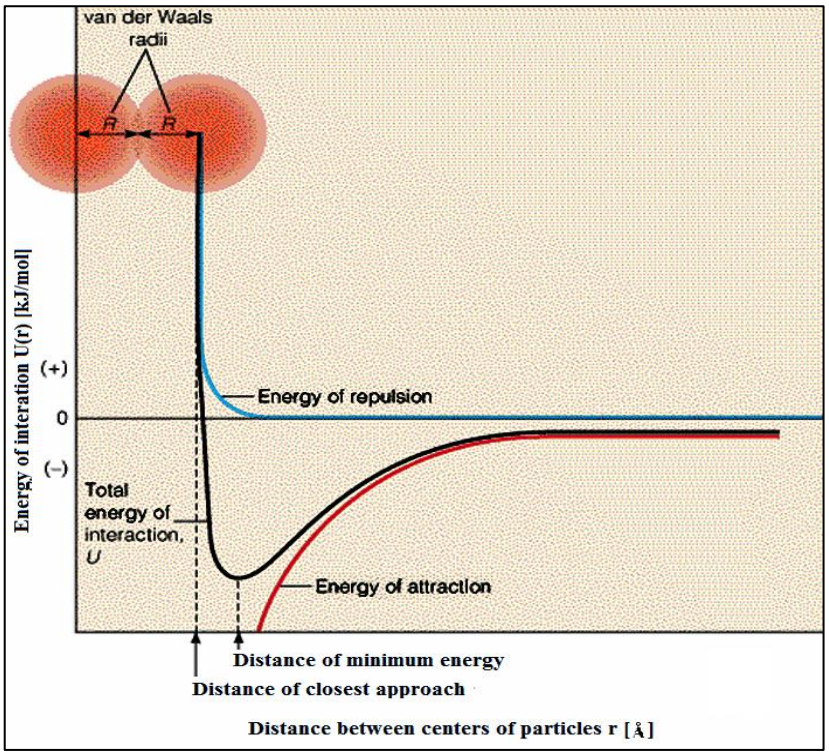

Figura 6: Potenziale semi-empirico di Lennard-Jones.

In figura 7 viene mostrato l'andamento delle forze agenti in funzione della distanza sonda-campione, relativamente alle tecnica esposte in precedenza.

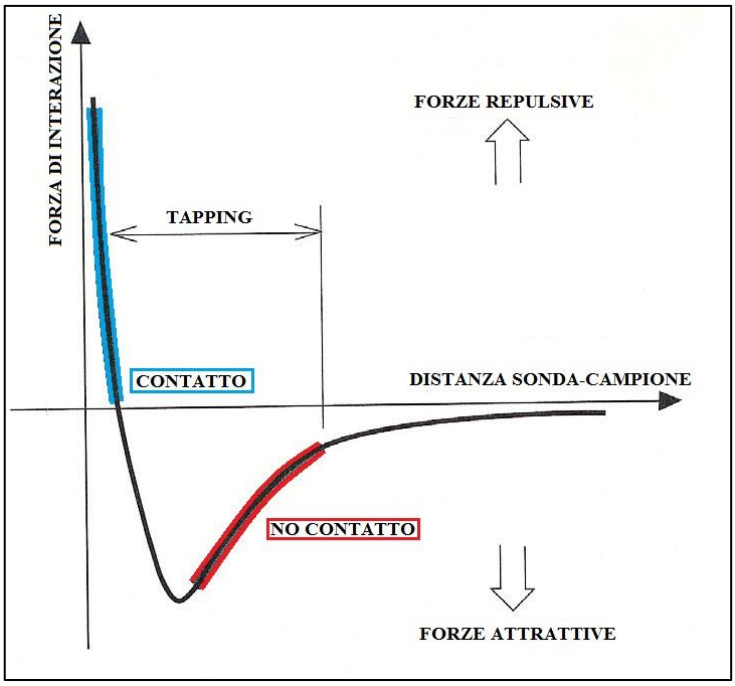

Figura 7: Forza di interazione in funzione della distanza sonda-campione.

1.2.3 Curva forza-distanza

L'AFM, come detto in precedenza, può essere impiegato per misurare o per applicare forze, per mappare le superfici dei campioni sia che questi siano biologici o meno, isolanti o conduttori, in atmosfera ambiente o in liquido. Pertanto è possibile acquisire curve di forza per ogni tipo di superficie con una risoluzione dell'ordine del pN. La curva forza-distanza in forma generale è mostrata in figura 8.

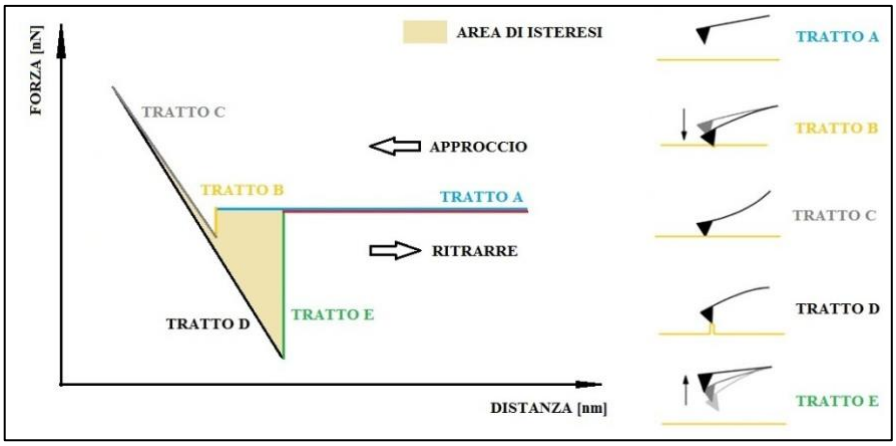

Figura 8: Curva forza-spostamento.

L'andamento è ottenuto facendo muovere il campione (o la punta della sonda) lungo l'asse verticale, acquisendo i valori di deflessione della sonda mediante una tecnica che sfrutta la riflessione di un fascio laser incidente. Inizialmente (tratto A) la sonda, non trovandosi a contatto con la superficie da esaminare, non subisce alcuna deformazione ad opera di forze esterne. Avvicinando la sonda al campione, questa risente della massima interazione attrattiva e si parla di *jump-to-contact* (tratto B); a contatto avvenuto, la deflessione della sonda subisce un incremento ad opera di un dispositivo piezoelettrico che spinge il campione sulla sonda (tratto C). Raggiunta una data forza massima, il processo viene invertito. A seguito dei legami di adesione e/o dei legami formati durante il contatto (forze di capillarità se si opera in ambiente liquido), durante l'allontanamento della sonda dal campione, il legame di adesione permane fino ad una distanza superiore al precedente punto di contatto; si parla di *jump-off-contact* o di isteresi (tratto D). Rotto il legame di adesione, avviene il distacco della sonda (tratto E).

1.3 Schema di funzionamento dell'AFM

Il microscopio a forza atomica, figura 9, è generalmente costituito da tre macro-parti, la testa ottica, il cantilever holder e il basamento. La prima è costituita da una sorgente laser, da un detector ed eventualmente da un insieme di filtri e di lenti, la seconda è costituita dalla sonda e dal suo supporto mentre la terza da una serie di movimentazioni x, y, z denominati scanner. Il microscopio a forza atomica utilizza un sensore denominato *cantilever* lungo 100÷400 μm, costituito da una punta (tip) molto appuntita localizzata nel suo estremo libero; la punta viene portata a contatto con la superficie del campione, il quale è movimentato, lungo le direzioni x e y, tramite un apposito dispositivo denominato *scanner*. In figura 10a è mostrata la traiettoria seguita durante il processo di scansione. Un fascio laser (di una data lunghezza d'onda) colpisce la superficie della leva; questa, deformandosi a seguito delle forze agenti all'interfaccia punta-superficie del campione, riflette il fascio laser verso un fotodiodo a quattro quadranti. Questo dispositivo permette di quantificare gli spostamenti in x e y (deflessione e torsione) del fascio riflesso. Viene definito *guadagno della leva ottica* il rapporto tra la distanza fotodiodo-leva e la sua lunghezza della sonda (leva). Tale coefficiente permette di amplificare di svariate volte la deformazione del cantilever. In figura 10b viene mostrato la variazione d'angolo del fascio laser in funzione della deformazione della leva. Alcune modalità operative prevedono l'uso si un sistema di feedback, mediante il quale il campione viene movimentato lungo la direzione z (in alto o in basso), tramite l'uso di uno scanner, di una quantità uguale o contraria alla deformazione della leva al fine di ristabilire una certa posizione di riferimento. Questo sistema è importante soprattutto nell'analisi dei sistemi biologici, poiché consente di applicare una forza minima sul campione limitandone il danneggiamento.

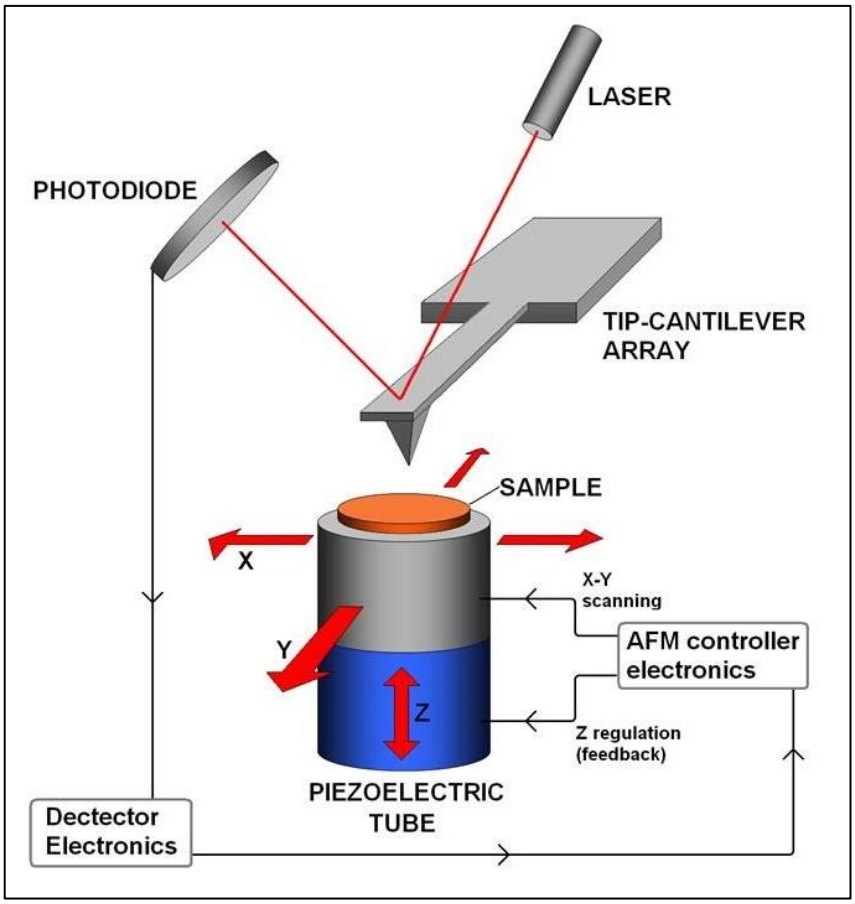

Figura 9: Schema di un AFM.

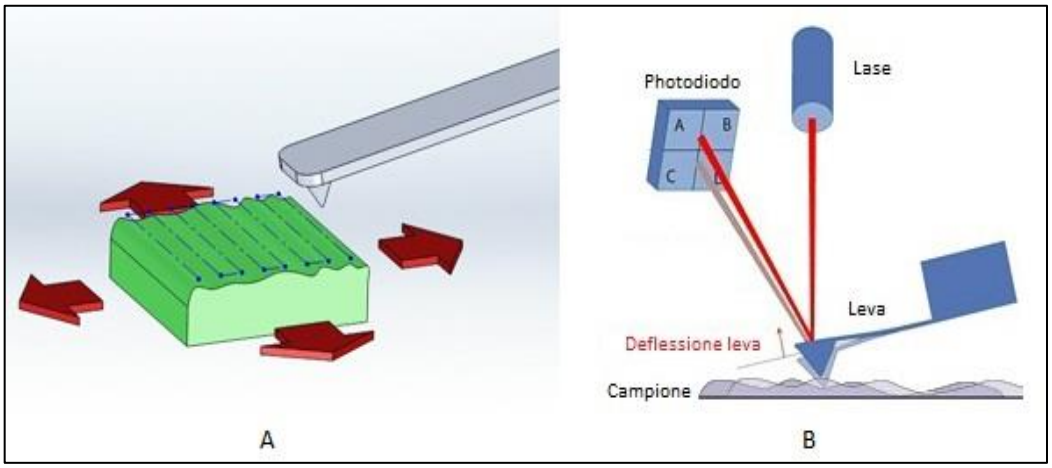

Figura 10**:** a) percorso di scansione; b) variazione d'angolo del fascio laser funzione della deformazione della leva.

1.3.1 Laser – Detector

L'acquisizione di informazioni relative ad una topografia o ad un'analisi di forza, viene effettuata registrando le piccole deflessioni che la sonda subisce. In figura 11 è mostrato come, a seguito della variazione di asperità superficiale del campione, la deflessione della leva produce una variazione d'angolo del fascio laser riflesso sul fotodiodo. La grandezza Y rappresenta lo spazzolamento prodotto dal laser riflesso a seguito della deflessione della leva y. Pertanto, se la sonda ha una lunghezza di 200µm e il detector è situato a 40mm dalla sonda, a seguito di una deflessione di 1µm si otterrà uno spostamento del fascio di $2 \cdot 10^{-4}$m.

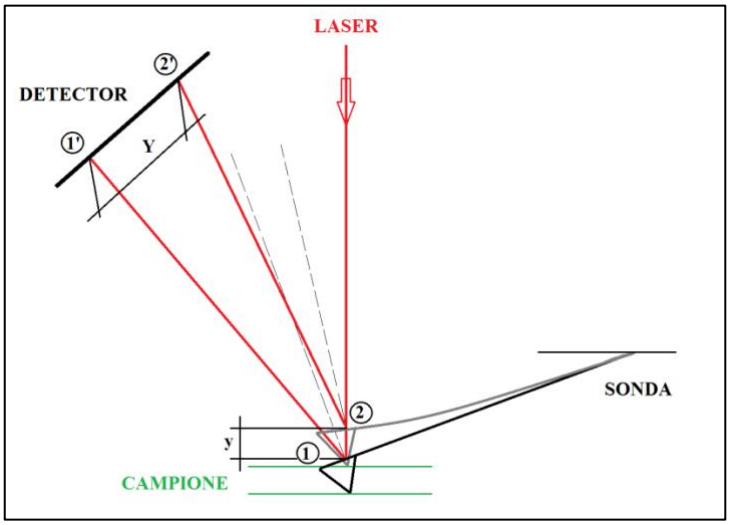

Figura 11: riflessione laser-sonda.

Il laser generalmente utilizzato possiede un picco massimo in uscita di pochi mW con una lunghezza d'onda pari a 670nm.
Il fotodiodo che funge da detector, può possedere due o quattro quadranti; quest'ultimo permette di determinare la deflessione longitudinale e la torsione trasversale della sonda generate durante la scansione. In figura 12 è

mostrato come, a seguito della deformazione della sonda, il fascio laser riflesso permetta di misurare la forza attrattiva/repulsiva o la forza laterale.

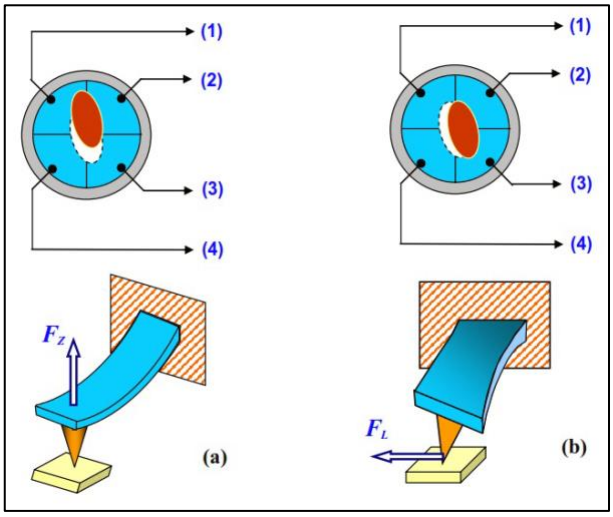

Figura 12**:** a) forza longitudinale attrattiva/repulsiva funzione dello spot laser sul detector; b) forza torsionale funzione dello spot laser sul detector.

Indicando le correnti di riferimento (deflessione nulla della sonda) come $I_{o1}, I_{o2,} I_{o3,} I_{o4,}$ e con $I_1, I_{2,} I_{3,} I_{4,}$ le correnti dovute allo spostamento della sonda, allora le mutue differenze $\Delta I_i = I_i - I_{oi}$ caratterizzano l'intensità e la direzione dello spostamento della sonda (per flessione o per torsione), quindi:

$$\Delta I_z = (\Delta I_1 + \Delta I_2) - (\Delta I_3 + \Delta I_4) \tag{1.3}$$

$$\Delta I_l = (\Delta I_1 + \Delta I_4) - (\Delta I_2 + \Delta I_3) \tag{1.4}$$

L'equazione (1.3) permette di determinare una corrente proporzionale alla deflessione dovuta ad una forza normale alla superficie (figura 12a) mentre

l’equazione (1.4) permette di misurare la torsione dovuta alle forze laterali (figura 12b).

1.3.2 Sonda

Le sonde per AFM sono microscopiche leve (cantilever) dotate di una punta appuntita (tip) situata all’estremità libera e di un *chip* di silicio posto all’estremo opposto (figura 13). Esse vengono prodotte mediante tecniche di fotolitografia e ad attacco chimico di strati di silicio, SiO_2 o , Si_3N_4 depositati su di un wafer di silicio. Un cantilever può avere una lunghezza di 100÷400μm mentre la punta è alta circa 10μm. Il raggio di curvatura della punta, approssimativamente di 5÷10nm, e l’angolo di apertura all’apice (10°÷20°) rappresentano due parametri fondamentali per la valutazione della risoluzione dello strumento, ma allo stesso tempo costituiscono i principali limiti allo sviluppo della microscopia a forza atomica. Le leve possono avere diverse conformazioni geometriche, rettangolari, triangolari mentre le punte possono essere coniche (vedi figura 14), piramidali a base quadrata o sferiche.

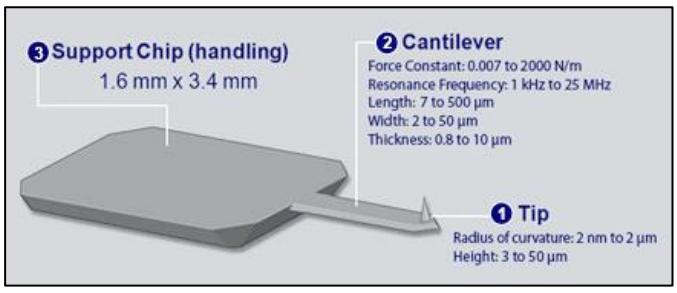

Figura 13: parti che costituiscono la sonda AFM

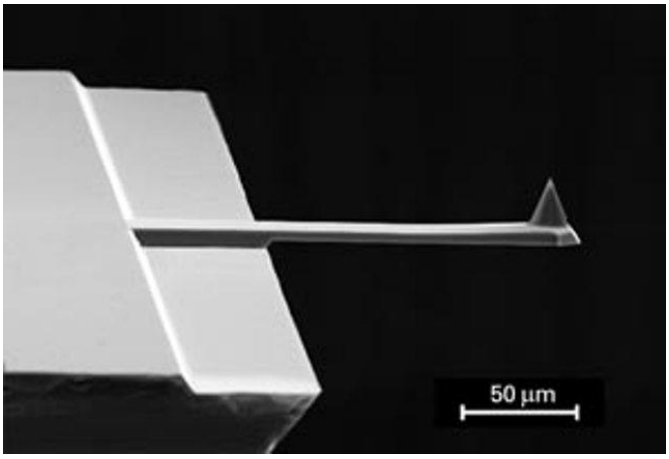

Figura 14: esempio di punta conica.

1.4 Calibrazione del cantilever

Per valutare l'intensità della forza normale esercitata dalla punta sul campione, delle forze di attrazione e delle forze di adesione, è necessario conoscere il valore della costante elastica k del cantilever. Tale costante è definita come il rapporto tra la forza normale applicata all'estremo libero del cantilever e la conseguente deflessione che questo subisce. La costante k viene generalmente fornita dal produttore, il quale però ne fa una stima con un ampio margine di incertezza. Ad esempio, per le punte in Si (L= 230 ±5µm, a= 40±3µm e b=7±0.5µm) utilizzate per le misure di indentazione dinamica, la costante k viene stimata dal produttore nell'intervallo 25-60 N/m. Per una più precisa valutazione di k è necessario implementare una tecnica di misura non distruttiva per caratterizzare ciascun cantilever utilizzato.

L'idea di base è quella di utilizzare il teorema di equipartizione dell'energia. Esso afferma che all'equilibrio termico l'energia è equamente distribuita e ha un valore di $\frac{1}{2}\,k_B T$ per ogni grado di libertà. Quando il cantilever è deflesso di una quantità pari a $\sqrt{\langle x^2 \rangle}$, la sua energia potenziale è $\frac{1}{2}\,k\,\langle x^2 \rangle$.

Eguagliando le due energie si ottiene:

$$\frac{1}{2}\, k\, \langle x^2 \rangle = \frac{1}{2}\, k_B T \tag{1.5}$$

Dove k_B è la costante di Boltzmann, T la temperatura e $\langle x^2 \rangle$ la deflessione quadratica media misurata per un tempo sufficientemente lungo.
Risolvendo il problema di equipartizione dell'energia per il valore k si ha

$$k = \frac{k_B T}{\langle x^2 \rangle} \tag{1.6}$$

Alternativamente, basandosi sulla distribuzione di probabilità delle fluttuazioni e dalla teoria di Hutter e Bechhoefer[5], si giunge alla formula:

$$\langle x^2(\nu) \rangle = \frac{2k_B T}{\pi k Q \nu_k} \frac{\Delta\nu}{\{[1 - (\frac{\nu}{\nu_k})^2]^2 + [\frac{\nu}{\nu_k Q}]^2\}} \tag{1.7}$$

dove:

- k_B è la costante di Boltzmann;
- T è la temperatura del cantilever in Kelvin (in equilibrio termico);
- Q è il fattore di qualità del cantilever;
- ν_k è la frequenza di risonanza, quella frequenza alla quale il cantilever eccitato risponde con la massima ampiezza;
- $\Delta\nu$ è la risoluzione in frequenza;
- k è la costante elastica del cantilever;
- $x^2(\nu_k)$ la deflessione quadratica media alla frequenza di picco.

[5] J. Hutter and J. Bechhoefer. Calibration of atomic-force microscope tips. Rev. Sci. Instrum. 64, 3342 (1993).

La frequenza di risonanza e la rigidità della leva sono fornite solo indicativamente dal costruttore, per conoscerle in maniera più precisa è preferibile calibrarle sperimentalmente. Acquisendo il segnale grezzo di deflessione si possono ricavare, da un fitting non-lineare, i nuovi valori di frequenza e rigidità, grazie alle correzioni apportate tramite esperimenti effettuati da Butt and Jaschke[6].

La formula (1.7), tenendo conto in maniera empirica di rumori elettronici e meccanici, diventerà:

$$\langle x^2(\nu)\rangle = \frac{A}{\nu} + B + \frac{\langle x^2(\nu_k)\rangle}{Q^2} \frac{1}{\{[1-(\frac{\nu}{\nu_k})^2]^2 + [\frac{\nu}{\nu_k Q}]^2\}} \tag{1.8}$$

Dove A e B sono parametri liberi. A è in nm^2 Hz e B in nm^2 .
In realtà noi misuriamo una tensione ΔV che è proporzionale allo spostamento del cantilever e quindi le fluttuazioni vanno corrette secondo la seguente formula:

$$\langle x^2(\nu)\rangle = \frac{3}{4} \langle \Delta V^2(\nu)\rangle \left[\frac{\Delta z_m}{\Delta V_m}\right]^2 \left[\frac{1}{cos\vartheta}\right]^2 \tag{1.9}$$

Dove Δz è lo spostamento del cantilever lungo l'asse z, inclinato di un angolo ϑ rispetto all'orizzontale.
Dal fitting della 1.8 dei dati sperimentali si ricavano i seguenti parametri di fitting: A, B, Q, $\langle x^2(\nu_k)\rangle$ e ν_k .
Infine, ottenuti i parametri di fitting possiamo ricavare la costante di elasticità k[7], data da:

[6] Butt and Jaschke M 1995 Nanotechnology 6. 1-7.
[7] N.A. Burnham, X Chen, C.S. Hodges, G.A. Matei, E.J. Thoreson, C.L. Roberts, M.C. Davies and S.J.B. Tendler. Comparison of Calibration methods for atomic-force microscopy cantilevers. 2002.

$$k = \frac{2}{\pi} \frac{k_B T Q}{x^2(\nu_k)} \frac{\Delta\nu}{\nu_k} \tag{1.10}$$

Il vantaggio di questo metodo di calibrazione consiste nel fatto che è basato sulla forma dello spettro di potenza delle fluttuazioni. In questo modo è possibile separare contributi spuri alle fluttuazioni che non provengono direttamente dal cantilever, ottenendo così risultati più affidabili.

2 Scelta dei componenti Hardware

In questa sezione viene elencato l'hardware necessario per realizzare il sistema presentato nell'introduzione.

- Raspberry Pi;
- Scheda SunCard con chip FFTDI
- AFM – Atomic Force Microscope

2.1 Raspberry Pi

Si è scelto di adottare il Raspberry Pi, un single-board computer[8], implementato su una singola scheda elettronica e sviluppato in UK dalla fondazione Raspberry Pi. La scelta è stata legata a questioni di basso costo, dimensioni ridotte, facilità di utilizzo e soprattutto facente parte della categoria dei dispositivi mobili.

Figura 15: Raspberry Pi.

[8] Raspberry Pi, Wikipedia. https://it.wikipedia.org/wiki/Raspberry_Pi

Da febbraio 2012, la fondazione distribuisce due modelli di Raspberry Pi:

- Modello A: da 256 Mb di RAM e costa 25 dollari. Ha una singola porta USB ed è privo di controller Ethernet.
- Modello B: da 512 Mb di RAM e costa 35 dollari. Ha due porte USB e un controller Ethernet 10/100.

Il Raspberry PI, adottato per lo sviluppo del progetto, è il Modello B, da 512 Mb di RAM ed usa il sistema operativo Linux, distribuzione Debian GNU/Linux.

In tabella 2, sono mostrate le caratteristiche e le specifiche tecniche del modello B:

Prezzo	**35 USD**
SOC	Broadcom BCM2835 (CPU + GPU + DSP + SDRAM)
CPU	700MHz ARM1176JZF-S core (famiglia ARM11)
GPU	Broadcom VideoCore IV, OpenGL ES 2.0, 1080p30 H.264 high-profile decode
Memory (SDRAM)	512 Megabytes (condivisa con la GPU)
USB 2.0 ports	2 (attraverso un hub USB integrato)
Output Video	Connettore RCA per il video composito, HDMI
Memoria	SD / MMC / SDIO card slot
Collegamenti di rete	Ethernet 10/100 (RJ-45)
Periferiche di basso livello	2x13 header pins for GPIO, SPI, I^2C, UART, +3.3 Volt, +5 Vol
Real-time clock	No clock or battery
Corrente (Potenza) assorbita	700 mA, (3,5W)
Alimentazione	5 V via microUSB o GPIO header
Dimensioni	85,60 mm x 53,98 mm (3.370 inch x 2.125 inch)
Sistemi Operativi Supportati	Debian GNU/Linux, Arch Linux e Gentoo

Tabella 2: Specifiche tecniche del Raspberry Pi – modello B.

2.2 Scheda SunCard

L'hardware, realizzato dall'azienda ElbaTech srl, è basato su FT245RL USB FIFO e su un-convertitore ADR430 della Analog Devices Inc. (Figura 16).

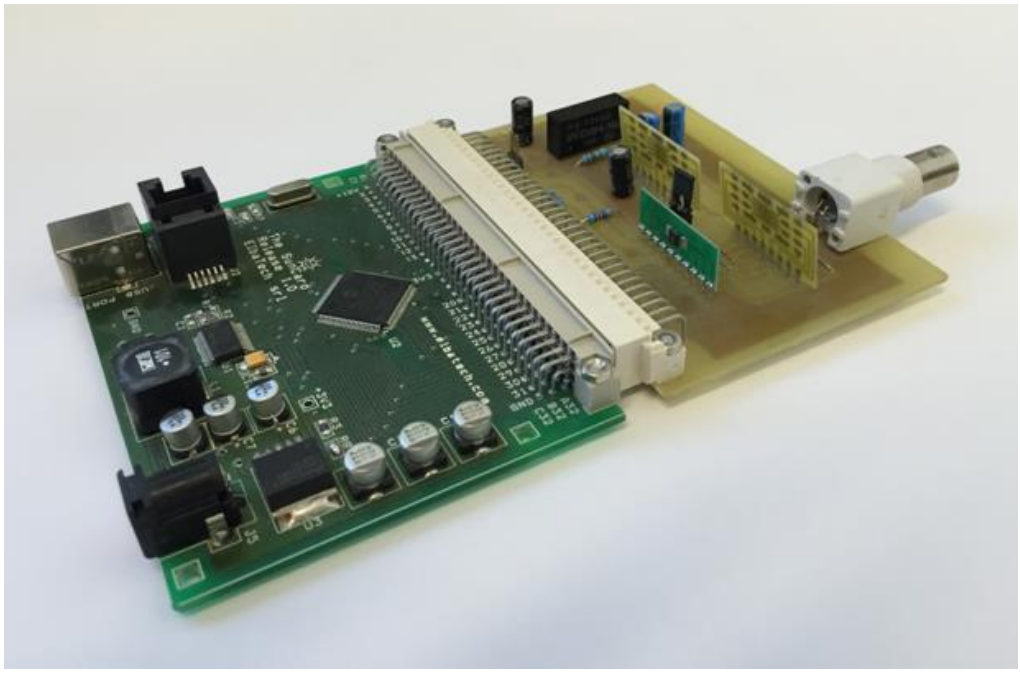

Figura 16: Scheda SunCard della Elbatech.

La scheda ha la funzione di ricevere il segnale analogico e di convertirlo in digitale, tramite il convertitore ADR430. Inoltre, è dotata di un chip FTDI (Future Technology Devices International) per poter pilotare attraverso una connessione USB un dispositivo che accetta il protocollo I^2C.

2.3 AFM – Atomic Force Microscope

Il microscopio a forza atomica, in figura 17, sviluppato nell'ambito del progetto di ricerca *BIOJERKER*, finanziato dall'Unione Europea, interamente dedicato alla progettazione e alla realizzazione di prototipi di misura innovativi, è stato realizzato presso l'Istituto di Biofisica (IBF) del

Consiglio Nazionale delle Ricerche (C.N.R.) di Palermo, in collaborazione con l'Istituto per l'Ambiente Marino Costiero (IAMC) di Capo Granitola (TP) e con la società DAIMAR s.r.l.

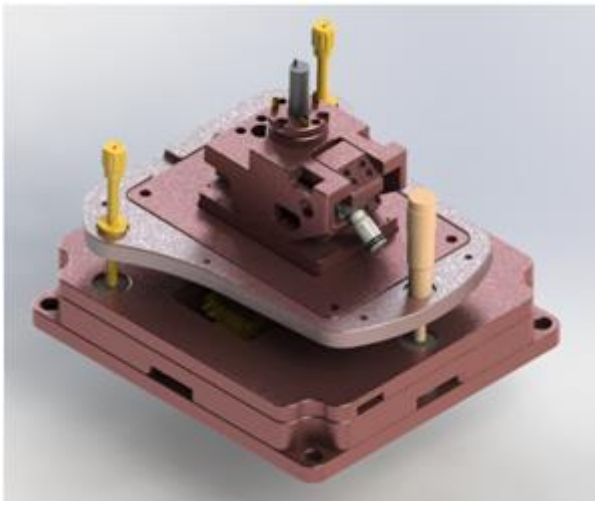

Figura 17: Microscopio a Forza Atomica Biojerker.

L'AFM presenta le seguenti caratteristiche:

- Scansione con movimentazione del campione utilizzando la testa di misura AFS a geometria pura (per proteine);
- Range di scansione standard utilizzando una movimentazione PiezoJena tipo TROTOR: 38x38x38 μm^3;
- E' possibile produrre scansione con movimentazione della punta, utilizzando la testa di misura AFS con accesso ottico (per cellule);
- E' possibile avere range di scansione con movimentazione XYZ n-Point tipo nPBio200 o nPBio300 rispettivamente da 200 e 300 μm^3, per studi su campioni biologici cellulari;
- Livello rumore del deflection detection del cantilever non in contatto < 10 nm RMS
- Livello rumore del deflection detection del cantilever in contatto < 20 nm RMS

- Rumore di posizione utilizzando uno scanner PiezoJena tipo TRITOR 38 XYZ: < 0.02nm RMS (in closed loop);
- Rumore di posizione utilizzando uno scanner nPoint tipo nPBio 2oo e 300 : < 1nm RMS (in closed loop);
- il supporto sonda (cantilever holder) è in materiale plastico detto PEEK, provvisto di un sistema magnetico di ancoraggio alla testa di misura;
- Software di acquisizione autoprodotto realizzato con LabView;

Prima di procedere al test di calibrazione per il calcolo della costante k del cantilever, si deve portare il microscopio in condizioni utili e ottimali per una corretta riuscita della misurazione.

I passi da eseguire sono:

- Verifica della temperatura ambientale $T = costante \approx 23°C$
- Accensione dei dispositivi elettronici utili alla misura e controllo delle connessioni dei cavi;
- Pulizia e montaggio della sonda;
- Verifica del corretto allineamento della sorgente laser con la sonda;
- Allineamento laser-fotodiodo;

Il dispositivo AFM utilizzato per le misurazioni, nel nostro caso in *aria*, a causa della sua elevata sensibilità è introdotto all'interno di un cassone che ha il compito di ridurre l'apporto di rumore sonoro e termico rilevabile dallo strumento. Infatti, rumori ad una data frequenza e opportuni gradienti termici potrebbero influenzare dannosamente la misurazione. La corretta pulizia della sonda, mediante agenti non aggressivi, è importante ai fini della misurazione in quanto permette di eliminare le eventuali impurità depositatesi nell'immediata vicinanza della punta; se non rispettata, potrebbero crearsi in fase di misurazione, delle azioni di distacco-attacco

non volute. Il corretto allineamento del fotodiodo permette di limitarne la sua saturazione in fase di misura.
L'AFM è dotato di un software per l'allineamento laser-sonda-fotodiodo in modo da garantire la corretta acquisizione dei dati.

Una volta che il laser risulti correttamente allineato sui quattro quadranti lo strumento è pronto e successivamente si potrà procedere al calcolo del k.

3. Scelta del s.o. e delle librerie

In questa sezione vengono illustrati i passi necessari riguardanti l'installazione del sistema operativo e delle librerie adottate per il funzionamento del sistema da realizzare, ovvero

- Raspbian OS;
- Python FTDI
- Numpy & Scipy
- JSON
- Tornado Server
- Flot

3.1 Installazione Raspbian OS – Debian Weezy

Per far funzionare il nostro Raspberry Pi, è necessario per prima cosa di essere forniti di una scheda SD[9], che conterrà il sistema operativo desiderato e le librerie che andremo successivamente a installare. La scheda SD in questione è da 32GB e si è scelto di installare il s.o. Raspbian – Debian Wheezy, una versione di Debian Linux.

Figura 19: Raspberry Pi + Debian.

Per la procedura di installazione vedi appendice A.

[9] Installazione S.O. http://www.raspberrypi.org/wp-content/uploads/2012/12/Raspberry_Pi_-_Guida_veloce.pdf

3.2 Python – FTDI

Python- FTDI è una libreria open source per comunicare con i chip FTDI.

In Appendice B la procedura di installazione.

3.3 Numpy

NumPy è il pacchetto fondamentale per il calcolo scientifico con Python. Contiene tra l'altro:

- un potente oggetto *array* N-dimensionale
- sofisticate funzioni (trasmissione)
- strumenti per l'integrazione di C / C ++ e Fortran
- algebra lineare, trasformata di Fourier, e la generazione di numeri casuali secondo diverse distribuzioni di probabilità.

Oltre suoi usi scientifici evidenti, NumPy può anche essere usato come un efficiente contenitore multidimensionale di dati generici. Possono essere definiti tipi di dati arbitrari. [10]

Per l'installazione del pacchetto, da terminale, eseguire:

$ sudo apt-get install python-scipy

3.4 Scipy

E' un software open-source per la matematica, scienza e ingegneria. La biblioteca SciPy dipende da NumPy, che offre un comodo e veloce gestione di array N-dimensionale. La biblioteca SciPy è costruita per funzionare con

[10] Numpy. Wikipedia. http://www.numpy.org/

le matrici di Numpy, e fornisce molti routine numeriche user-friendly ed efficienti, come la routine per l'integrazione numerica e l'ottimizzazione.

Per l'installazione:

$ sudo apt-get install python-scipy

3.5 JSON

JSON, acronimo di JavaScript Object Notation, è un formato di scambio dati leggero. Esso si basa su un sotto-insieme del linguaggio di programmazione JavaScript. JSON è un formato di testo che è completamente indipendente dal linguaggio, sfrutta le convenzioni dei linguaggi: C , C ++, C #, Java, JavaScript, Perl, Python, etc. Queste caratteristiche rendono JSON un linguaggio di scambio dati ideale.[11]

Per l'installazione di JSON, scaricare il pacchetto "simplejson-X.X.X.tar.gz" (con X.X.X l'ultima versione disponibile) dal sito: https://pypi.python.org/pypi/simplejson/.

Una volta scaricato, scompattare e spostare la cartella in "/usr/local/lib/". Successivamente, dentro la cartella spostata, eseguire da terminale il comando:

$ sudo python setup.py install

Per maggiori informazioni, riguardo l'installazione, visualizzare il sito web: https://pypi.python.org/pypi/simplejson/

[11] JSON. http://json.org/

3.6 Tornado Web Server

Tornado è un framework, leggero e scritto in Python, per la creazione di Web Server ad alte prestazioni. Le applicazioni sono progettate per essere elaborate in modalità non bloccante e supporta molte caratteristiche importanti, ad esempio i WebSocket.

Si è scelto di utilizzare Tornado per le sue elevate prestazioni. La tabella seguente mostra un test delle prestazioni di Tornado[12] contro altri server basati su Python:

Prestazioni su AMD Opteron, 2.4 GHz, quattro cores:

Server	Setup	Richieste al secondo
Tornado	Nginx, quattro interfacce	8213
Tornado	Una interfaccia, single-threaded	3353
Django	Apache/mod_wsgi	2223
Web.py	Apache/mod_wsgi	2066
CherryPy	Standalone	785

Tabella 3: Elenco Prestazioni dei Server

Per la procedura di installazione vedi appendice C.

3.7 Flot

Flot è una libreria gratuita basata su JavaScript per jQuery.

Con Flot è possibile creare dei grafici esteticamente gradevoli in maniera molto semplice e con una grande varietà di opzioni. Dal sito

[12] Tornado Web Server - https://en.wikipedia.org/wiki/Tornado_%28web_server%29

http://www.flotcharts.org/ scaricare l'ultima versione, nel nostro caso la 0.8.3, inserirla nell'apposita directory e scompattarla.

Dopo aver dichiarato lo script di Jquery e Flot, su un file HTML, basta inserire dove si vuol far comparire il grafico l'id con la seguente stringa: *placeholder*. Ad es.:

```
<div id="placeholder"
style="width:100%;height:120px;"></div>
```

Possiamo notare, dall'esempio sopra, che si dichiarano le dimensioni del grafico, tramite i parametri di *width* e *height*, e poi sarà lui stesso a riadattare il grafico in base alle misure dichiarate.

Per chi conosce Javascript e HTML sarà tutto limpido e chiaro: l'idea è quella di avere delle *options*, delle variabili di dati da graficare e un *contenitore* per mettere tutto assieme.

Di seguito mostro un esempio delle variabili di *options*, che sono quasi tutte facoltative:

```
var options = {
        legend: {
                show: false
        },
        series: {
                lines: {
                        show: true
                },
                points: {
                        show: true
                }
        },
        yaxis: {
                ticks: 10
        },
        selection: {
                mode: "xy"
        }
};
```

e poi si scrivono i dati da plottare nel grafico (d1):

var plot = $.plot("#placeholder", [d1]);

In questo caso “d1″ equivale a una variabile contenente i dati x, y del tipo:

```
var d1 = [ [x1, y1], [x2, y2], ... ]
```

Con questa libreria i grafici possono essere adattati alle necessità dell’utente, agendo sul:

- formato delle linee o barre (colore, background, punti, ombre, etc)
- formato della legenda (colore, background, margini, opacità, etc)
- formato degli assi x, y (valori minimi, massimi, valore dei punti, formato dei numeri, etc)
- formato della griglia sottostante (colore, background, opacità, cliccabile o no, etc)

In più essa offre una certa interattività con il grafico stesso: si può richiedere lo zoom del grafico, o il fatto che sia cliccabile restituendo così informazioni al click del mouse. Funzionalità per mostrare, per esempio, il valore di un punto passandoci sopra, e molte altre caratteristiche che in questo momento mancano ma che sono in via di implementazione.

Flot viene comunemente giudicata una delle librerie più belle e complete per creare grafici usando metodi avanzati ma allo stesso tempo semplici e intuitivi.

4. Strumenti di sviluppo

Gli strumenti utilizzati per lo sviluppo dell'applicativo comprendono linguaggi di programmazione e librerie, grazie alle quali si è arrivati alla realizzazione del prodotto software, finito e utilizzabile dall'utente.

4.1 Linguaggi utilizzati, lato Client

4.1.1 HTML

L'HTML, HyperText Markup Languag, è un linguaggio di formattazione che viene utilizzato per descrivere le modalità di impaginazione e/o visualizzazione grafica (layout) del contenuto, sia esso testuale che non, di una pagina web tramite dei tag di formattazione.

A differenza, ad esempio del PHP, l'HTML non è un linguaggio di programmazione, anche se supporta l'inserimento di script e oggetti quali immagini.
Infatti a differenza degli altri linguaggi non richiede alcuna dichiarazione di variabili, funzioni, etc, dal momento che il suo scopo è solamente quello di strutturare e decorare dei dati testuali.
Pertanto il compito del linguaggio HTML è quello di gestire i contenuti specificandone la struttura grafica, ossia il layout, all'interno della pagina web, la quale viene realizzata attraverso l'utilizzo di tag diversi e dove ogni tag va a specificare un diverso ruolo dei contenuti che esso stesso va a contrassegnare.
I documenti ipertestuali scritti in HTML vengono memorizzati in file la cui estensione è .html/.htm .

4.1.2 CSS

Il CSS (Cascading Style Sheets, fogli di stile) è un linguaggio che viene utilizzato per formattare i documenti HTML.
Il CSS è stato introdotto per separare i contenuti dalla formattazione e per fare in modo che la programmazione sia più leggibile e facile da utilizzare.
Ad ogni pagina possono essere collegati uno o più fogli di stile.
Il codice CSS da inserire in una pagina deve essere strutturato sotto forma di una o più regole, dove ognuna di esse rappresenta delle istruzioni di tipo *proprietà : valore,* le quali vengono applicate dal browser in fase di rendering agli elementi HTML a cui si riferiscono e i quali vengono specificati attraverso un selettore.
I documenti scritti in CSS vengono memorizzati in file la cui estensione è .css .

4.1.3 Javascript

Il Javascript è un linguaggio di scripting orientato agli oggetti, comunemente utilizzato nella programmazione web per dare interattività alle pagine HTML.

E' nato nel 1995 con il nome LiveScript ad opera della Netscape Communications, la società che ha prodotto Netscape Navigator, il browser più diffuso e utilizzato nel mondo nella prima metà degli anni '90.
LiveScript fu inizialmente incluso nel browser Netscape Navigator 2.0. in seguito anche Microsoft lavorò sul linguaggio producendo una sua variante chiamata JScript, che integrò nella versione 3.0 di Internet Explorer.

Per esigenze di unificazione la Netscape Corporation propose una standardizzazione del linguaggio e nel 1999 fu approvato lo standard ECMAScript (ECMA-262), che corrisponde al nome di Javascript.
Oggigiorno il nucleo del linguaggio è incorporato all'interno della maggior parte dei browser di navigazione.
JavaScript è impiegato principalmente per l'esecuzione di script eseguibili nei browser. In questo ambito, JavaScript permette di rendere dinamiche e interattive le pagine affinché mostrino contenuti diversi in base alle azioni dell'utente o a situazioni contingenti.
Mediante JavaScript è anche possibile aggiungere effetti grafici o accedere alle funzionalità del browser in cui le pagine sono visualizzate.
Con l'evoluzione dei browser, JavaScript ha subito numerose modifiche, acquisendo progressivamente nuove funzionalità.
Sintatticamente il linguaggio assomiglia al C, al C++ e a Java, anche se esistono profonde differenze: Javascript è un linguaggio interpretato, non compilato, ed è debolmente tipizzato.
Una delle principali limitazioni di JavaScript è la possibilità che gli script vengano facilmente disabilitati dall'utente tramite le impostazioni del browser.
Questo è possibile poiché il JavaScript è un linguaggio lato client, viene eseguito cioè sul computer dell'utente, che ha quindi tutto il diritto di disabilitare alcune funzionalità. Per questo motivo la gestione di dati sensibili viene affidata ad altri linguaggi lato-server.
Un'altra grande limitazione all'uso dei JavaScript è la compatibilità: più che per la programmazione HTML o CSS, un programmatore JavaScript deve essere molto attento che il suo lavoro sia compatibile con differenti browser e versioni più o meno recenti.
JavaScript in alcuni casi risulta non essere ottimale, il suo utilizzo è legato alla forte interattività che comporta l’aggiornamento dell’intera pagina web. Per ovviare a questa problematica è stata utilizzata la tecnica Ajax.

4.1.4 Ajax

Ajax è una tecnica di programmazione, basata su *JavaScript*, che fa uso di alcune funzionalità per comunicare *asincronamente* con il server. È asincrono nel senso che i dati sono richiesti al server e caricati in background senza interferire con l'operatività della pagina corrente.
Le applicazioni che fanno uso di questa tecnica possono inviare le richieste per ottenere solo i dati necessari, senza quindi ricevere un'intera pagina web da caricare. Se infatti siamo all'interno di una pagina che visualizza una mappa di una città nella quale si vogliono evidenziare tutti i monumenti, al click sul pulsante "Monumenti" non ci si aspetta il caricamento dell'intera pagina ma che vengano evidenziati sulla mappa i punti corrispondenti ai monumenti. Ajax garantisce proprio questo tipo di comportamento. La richiesta relativa ai monumenti viene inviata al server il quale la elabora e restituisce i dati che vengono successivamente processati dal client, producendo la visualizzazione dei monumenti sul browser dell'utente.
L'uso di questa tecnica porta ad una velocità di caricamento delle pagine molto maggiore rispetto alle applicazioni che usano la comunicazione sincrona, poichè viene ridotta in modo considerevole la quantità di dati scambiati. Anche il tempo di elaborazione delle informazioni da parte del server si riduce notevolmente, dato che non deve elaborare i dati relativi all'intera pagina ma solo la parte corrispondente alla richiesta.

4.1.5 JQuery

JQuery è una libreria di funzioni Javascript open source e cross-browser, concepita per semplificare la programmazione lato client delle pagine HTML.

Grazie a JQuery è possibile realizzare, con pochissime righe di codice, script altrimenti molto complessi che consentono di aggiungere effetti su testo, immagini, controlli di varia natura e di interagire in modo semplice e diretto con i CSS e AJAX.
JQuery si distingue nel panorama delle librerie Javascript per la sua sintassi sintetica e intuitiva, e non a caso il suo motto è *Write less, do more*, ovvero *Scrivi meno, fai di più*.
Uno dei vantaggi maggiori di JQuery consiste nel fatto che è in grado di gestire molti problemi di compatibilità tra browser. Chiunque programmi in Javascript è consapevole dei problemi, anche relativamente semplici, che possono esserci tra i DOM (Document Object Model) dei vari browser.
Usando JQuery non ci si preoccupa più di incorrere in questi problemi, poiché sono costantemente monitorati e gestiti da un attento team di sviluppo e da una community sempre più numerosa di sviluppatori che contribuisce costantemente al suo miglioramento.
La libreria JQuery è stata progettata per essere il più possibile espandibile.

Includendo solo una serie di caratteristiche nel core e fornendo un framework per estendere la libreria, ha semplificato la creazione e la distribuzione di plugin che arricchiscono le funzionalità di base della libreria.
Attualmente esistono centinaia di eccellenti plugin che possono essere scaricati dalla rete, a cui se ne aggiungono sempre nuovi. JQuery mette a disposizione anche una libreria separata denominata JQuery UI (User Interface, Interfaccia utente) che si focalizza sull'implementazione delle interfacce grafiche.
JQuery UI comprende una serie di controlli e widget avanzati altamente personalizzabili come menu a fisarmonica, cursori, data picker, finestre di dialogo e molti altri, che permettono di assemblare rapidamente solide interfacce utente con il minimo sforzo.

Tutte queste caratteristiche hanno reso JQuery un componente indispensabile e fondamentale, al punto che molti ormai confondono lo stesso Javascript con JQuery.
La libreria è stata sviluppata da John Resig che ha pubblicato la prima versione nell'Agosto 2005, anche se la prima versione stabile venne rilasciata solo nell'Agosto dell'anno successivo. Da allora la libreria ha subito diverse revisioni e aggiornamenti da parte di un gruppo sempre più folto di programmatori.
L'ultima versione disponibile è la 1.10.2, quella sfruttata per l'implementazione del codice dell'applicativo, ed è scaricabile gratuitamente dal sito ufficiale http://jquery.com e disponibile in due versioni: quella per la produzione e quella per lo sviluppo.
La versione per la produzione è *minimizzata*, ovvero i simboli (nomi di variabili, funzioni e così via) sono stati ridotti all'osso e il codice compresso con gzip (GNU zip).
Questa versione ha il vantaggio di essere molto compatta e di richiedere poca banda, il browser è quindi in grado di scaricarla molto velocemente, ma è poco adatta per il debug.
La seconda è invece la versione *normale* della libreria, ed è costituita da numerose linee di codice in chiaro facilmente visionabili in fase di sviluppo.

4.2 Linguaggi utilizzati, lato Server

4.2.1 Python 2.7

Dopo aver installato le librerie necessarie, si è scelto di utilizzare la versione 2.7 di Python. Esso è un linguaggio di programmazione dinamico orientato agli oggetti, moderno, dalla sintassi semplice e potente che ne

facilita l'apprendimento. Viene utilizzato per molteplici applicazioni di sviluppo: per realizzare interfacce grafiche, per interagire con i database, per lo sviluppo di web application, etc.

Python è un linguaggio multi-paradigma, supporta anche quello Object Oriented con supporto all'ereditarietà multipla ed offre una tipizzazione dinamica forte e il controllo sui tipi viene eseguito al runtime. In parole povere una variabile è un contenitore al quale viene associata un'etichetta, il nome, che può essere associata a diversi contenitori anche di tipo diverso durante il suo tempo di vita.
Python usa un garbage collector per la liberazione automatica della memoria, sollevando il programmatore dal difficile compito di gestire la memoria.
La gestione degli errori in Python avviene tramite le eccezioni, che sono provocate da un evento anomalo o imprevisto che cambia il normale flusso d'esecuzione del codice.
Ogni eccezione può essere gestita e controllata oppure non gestita, in questo caso l'errore può anche provocare l'arresto del programma, fornendo al programmatore tutte le informazioni necessarie per localizzarlo.
Python è fornito di una libreria built-in estremamente ricca che, unitamente alla gestione automatica della memoria e a robusti costrutti per la gestione delle eccezioni, fa di Python uno dei linguaggi più ricchi e comodi da usare.
Python è un linguaggio interpretato in quanto un interprete si occupa di analizzare il codice sorgente e, se sintatticamente corretto, di eseguirlo. L'essere interpretato rende Python un linguaggio estremamente portabile. Una volta scritto un sorgente, esso può essere interpretato ed eseguito sulla gran parte delle piattaforme utilizzate: Mac, Windows, Linux. Per il suo funzionamento, basta la presenza della versione corretta dell'interprete di Python e nei rari casi in cui non sia disponibile per la propria piattaforma, basta avere un compilatore C per scaricare il codice sorgente e compilare

l'interprete.
Infine Python è un software open source, completamente free.

Di seguito, elenco alcune aziende importanti e utilizzatori di Python[13]:

- Google: usa Python per il suo sistema di ricerca e per le sue creazioni (youtube).
- BitTorrent: il popolare software di P2P è scritto in Python.
- Maya e Blender: due popolari software di sviluppo 3D.
- Intel, Cisco, HP, Seagate, Qualcomm e IBM: usano Python per il testing dell'hardware.
- Industrial Light & Magic, Pixar: usano Python nella produzione dei film animati.
- Nasa, Fermilab: usano Python per lavori scientifici.
- IRobot: utilizza Python per lo sviluppo commerciale dei robot.

4.2.2 Storia

Python fu creato nel lontano natale del 1989 da Guido Van Rossum[14], in figura 25, che invece di passare le vacanze a decorare l'albero, decise di scrivere un linguaggio che correggesse in gran parte, se non tutti, i difetti che secondo lui erano presenti negli altri linguaggi; pubblicò la sua creazione nel febbraio del 1991. Il nome Python deriva dalla passione di Van Rossum per un gruppo di comici inglesi degli anni settanta, i Monty Python; curiosamente anche il termine *spam* viene da uno degli sketch di questo gruppo.

[13] Mark Lutz, Learning Python. O'Reilly, 2009.
[14] Nascita e Storia di Python, Wikipedia. 2015. https://it.wikipedia.org/wiki/Guido_van_Rossum

Figura 25: Guido Van Rossum.

Con il rilascio della versione 3.0 di Python, nel dicembre 2008, si è assistito ad una vera rivoluzione che ha introdotto molti miglioramenti e ulteriori novità al linguaggio.

5 . Caso di Studio: Analisi e Progettazione

5.1 Analisi dei Requisiti

Il primo e fondamentale passo per la progettazione dell'applicativo da sviluppare è quello di raccolta e analisi dei requisiti funzionali che l'applicativo deve soddisfare.
Per raccolta dei requisiti si intende la completa individuazione dei problemi che l'applicazione da realizzare deve risolvere e le caratteristiche che tale applicazione dovrà avere. Per caratteristiche del sistema si intendono sia gli aspetti statici (i dati) sia gli aspetti dinamici (le operazioni sui dati). I requisiti vengono inizialmente raccolti in specifiche espresse generalmente in linguaggio naturale e, per questo motivo, spesso ambigue e disorganizzate.
L'analisi dei requisiti consiste nel chiarimento e nell'organizzazione delle specifiche dei requisiti. Le principali fonti di informazioni per la raccolta dei requisiti sono:

1. Il progettista dell'AFM: informazioni ottenute mediante interviste, colloqui, con il progettista stesso;
2. Responsabile del progetto del sistema informativo: interviste e studio preliminare per dettagliare le procedure da eseguire;
3. La documentazione esistente: reperimento materiali, specifiche, per realizzare il sistema;

L'applicativo dovrà fornire all'utente la possibilità di eseguire svariati tipi di operazioni, tra le quali l'inserimento dei parametri di calibrazione, la

visualizzazione dello spettro, la possibilità di zoomare un punto preciso del segnale selezionando un range di valori, ecc. Sarà necessario, inoltre, effettuare un fitting della Trasformata discreta di Fourier, per ottenere i parametri per la calibrazione del cantilever.

Occorre tuttavia tener presente anche che la comunità scientifica non possiede competenze particolari in ambito informatico, provocando spesso problemi di utilizzo delle varie applicazioni create in precedenza. La creazione di un'interfaccia semplice ed intuitiva facilita dunque molto il lavoro di indagine scientifica. Si è quindi pensato di creare una Web Application che si avvicinasse molto ad una Desktop Application, con il vantaggio di permettere l'accesso in qualsiasi parte del pianeta avendo a disposizione solo un PC (o anche un semplice smartphone o tablet) con connessione ad internet ed un browser, senza avere i comuni problemi di installazione e compatibilità causati dai vari sistemi operativi.

5.2 Requisiti di funzionamento dell'applicativo

Il requisito principale dell'applicazione è di fornire un'interfaccia grafica user–friendly, comprensibile dagli utenti. I tre pilastri fondamentali su cui, in generale, si basa una GUI sono:

- **Struttura:** suddivisione delle varie funzioni con i relativi criteri d'interazione;
- **Reattività:** reazione ad eventi con delle azioni pianificate;
- **Visualizzazione:** dotazione di un aspetto piacevole e in grado di facilitare la comprensione degli oggetti presenti;

La GUI è uno strumento reattivo, cioè esegue un'azione solo dopo un input esterno (come ad esempio l'interazione con i bottoni) quindi ogni suo componente deve essere chiaro ed inconfondibile.

Nonostante queste regole, all'utilizzatore vengono comunque richieste alcune conoscenze di base per utilizzare la Web Application:

- Saper utilizzare un browser;
- Avere i fondamenti della navigazione sul web;
- Avere una conoscenza scientifica di base sulla microscopia a forza atomica e delle forze di interazione tra sonda e cantilever.

I principali requisiti di funzionamento dell'applicazione sono:

1. **Inserimento dei valori:** l'utente deve poter inserire i valori dei parametri di ciascun oggetto, ad es. parametri di inizializzazione del sistema, di setup per la calibrazione del cantilever;
2. **Modifica dei valori:** l'utente deve poter modificare i valori dei parametri di ciascun oggetto;
3. **Visualizzazione delle informazioni:** l'utente deve poter visualizzare in un'unica pagina tutte le informazioni relative alle proprietà meccaniche della leva;
4. **Estendere il range sui grafici:** l'utente deve poter selezionare sui grafici il range desiderato per analizzare e visualizzare meglio le informazioni relative alla Trasformata di Fourier e al fitting;
5. **Visualizzare il significato degli oggetti:** l'utente deve poter visualizzare per ogni oggetto/parametro la relativa descrizione in modo da sapere il significato;

6. **Visualizzare le formule:** l'utente deve poter visualizzare la formula associata ai parametri di calibrazione e del fitting cliccando sulla voce stessa. Al click si aprirà una finestra di popup.

Inoltre, il funzionamento dell'applicativo dovrà attivare/interrompere lo scambio dei dati in real-time tramite due pulsanti di Start e di Stop.

5.3 Specifiche del sistema da realizzare

Le caratteristiche e i componenti principali del sistema da realizzare saranno:

- L'architettura dell'applicativo sarà di tipo client/server e la comunicazione si baserà sul protocollo http (protocollo di trasferimento di un ipertesto), il quale viene utilizzato come sistema principale per la trasmissione delle informazioni sul Web.
- Verrà utilizzato il Raspberry Pi, un single board computer, per la parte di Server e messo in comunicazione con una scheda SunCard per la conversione del segnale analogico in digitale e la sua acquisizione.
- La scheda SunCard ha la funzionalità di ricevere il segnale analogico dallo strumento di misura AFM e di convertirlo in digitale, tramite il convertitore ADR430. E' anche dotata di un chip FTDI per pilotare attraverso una connessione USB un dispositivo che accetta il protocollo I2C.
- L'AFM, per la determinazione della costante elastica K del cantilever

5.4 Il caso d'uso

Per poter descrivere il funzionamento dell'applicativo, essendo esso rappresentato da un sistema dinamico, sono stati utilizzati, durante la sua progettazione, degli Use Case Diagrams.
Nell'applicazione web da sviluppare è stato previsto un singolo caso d'uso: uno scenario dove è possibile vedere tutte le operazioni che un utente può effettuare (vedi figura 20).

L'utente potrà, quindi:

- Inserire e/o modificare i parametri di inizializzazione e/o di setup;
- Cliccare il pulsante di Start per lo scambio dei dati in real-time;
- Cliccare il pulsante di Stop per fermare lo scambio dati;
- Cliccare il pulsante di Reset Data per re-inizializzare il sistema;
- Visualizzare i parametri di fitting, impostati nella fase di inizializzazione, dei grafici e soprattutto il valore della costante k;
- Zoomare il grafico sull'asse delle frequenze, selezionando il range desiderato, in modo da facilitare il processo di fitting.
- Visualizzare per ogni voce di menù, il relativo dettaglio, la formula usata per il calcolo dei parametri di fitting e quella per ricavare la costante di elasticità k.

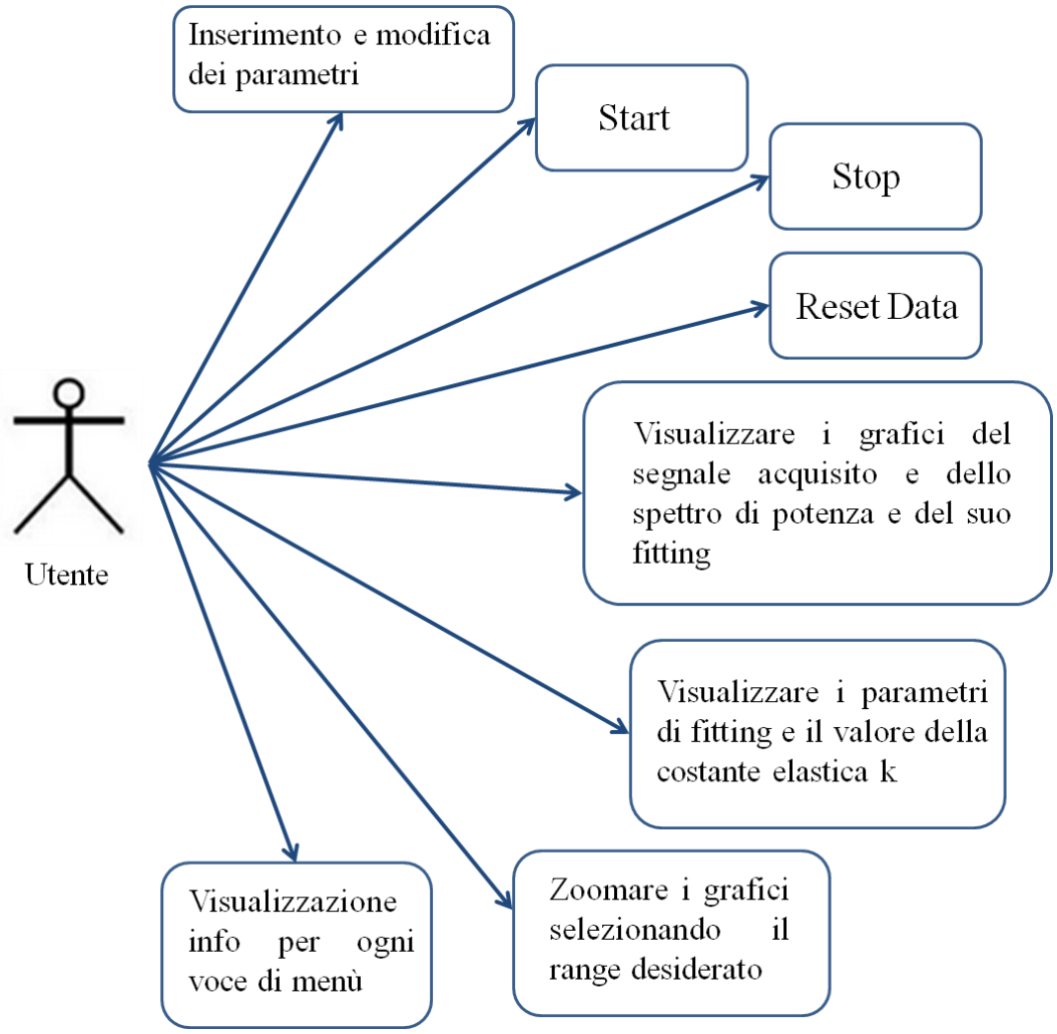

Figura 20: Use case diagram per l'interazione tra utente e applicativo

5.5 Activity Diagram

Attraverso gli Activity Diagrams è possibile modellare le azioni necessarie per compiere un'attività ed il flusso di controllo tra di esse. In figura 21 è stato definito un activity diagram dell'applicativo, che si pone ad un elevato livello di astrazione e che descrive tutte le possibili azioni ed il relativo flusso, che possono essere eseguite da un utente.

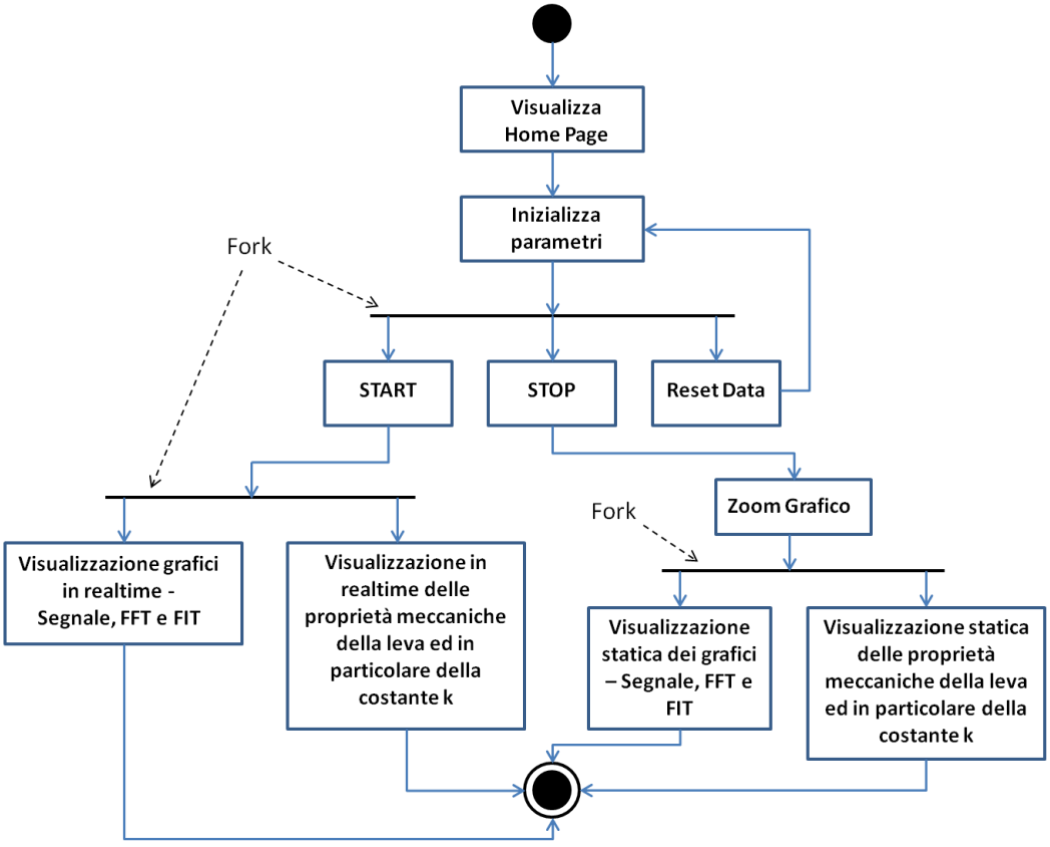

Figura 21: Activity Diagram dell'applicativo.

5.6 Sequence diagram

Il Sequence Diagram viene utilizzato per modellare la comunicazione tra un oggetto ed un altro in relazione al trascorrere del tempo. L'idea chiave qui è che le interazioni tra gli oggetti avvengano seguendo un ordine ben preciso e che tale sequenza avvenga, nel tempo, dall'inizio alla fine.
In figura 22 è riportato lo schema:

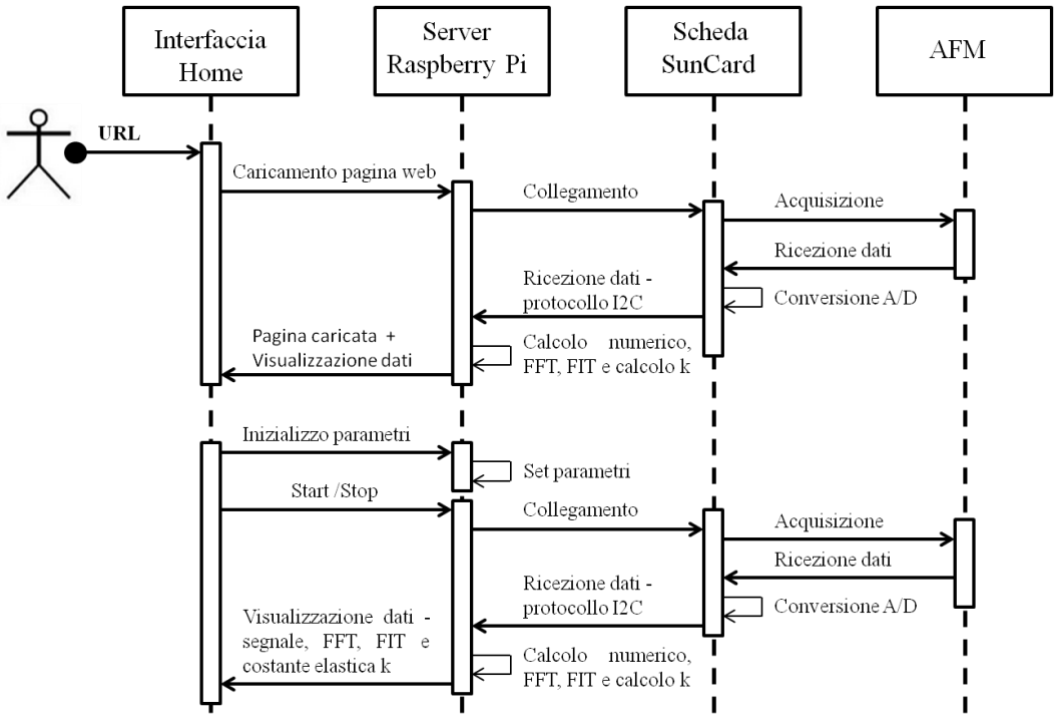

Figura 22: Sequence diagram dell'applicativo

5.7 Architettura del sistema

Si è scelto di adottare il paradigma client/server in quanto l'applicativo sviluppato si basa su un modello di interazione tra processi software, dove i processi interagenti si suddividono in:

- Client che richiedono i servizi;
- Server che offrono servizi.

Il processo client svolge un ruolo attivo, in quanto genera autonomamente richieste di servizi, ed è dedicato ad interagire con l'utente finale.
Il processo server è reattivo: svolge una computazione in modo iterativo solo a seguito di una richiesta da parte di un qualunque client ed è sempre in esecuzione per essere pronto a rispondere alle richieste.
Un'applicazione Web, nella maggior parte dei casi, si sviluppa su tre livelli logico-funzionali (applicazioni *Three-Tier*) ma che possono essere distribuiti anche su più livelli (applicazioni *Multi-Tier*). Nel nostro caso, l'architettura viene denominata two-tier in quanto in essa sono presenti un client, con funzioni sia di interfaccia utente sia di gestione dell'applicazione, e un server, dedicato alla gestione dati. Il server funge anche da web server. Dunque, il termine architettura two-tier, ovvero a due strati, indica una particolare tipologia di architettura software per l'esecuzione di un'applicazione, per la quale è pertanto prevista una suddivisione in due diversi livelli:

1. **Livello di presentazione** (Presentation layer): Il livello di presentazione costituisce l'interfaccia utente dell'applicazione Web e corrisponde a quello che nelle applicazioni client/server standard è il client. Esso è costituito dal browser e si occupa di acquisire dati e visualizzare risultati.

2. **Livello dei Dati** (Data layer): Quest'ultimo livello si occupa della gestione dei dati, i quali sono necessari al funzionamento dell'intero sistema. I dati vengono processati dal Raspberry Pi (Server) che ha il compito di acquisire il segnale dall'AFM, tramite la scheda SunCard, sfruttando il protocollo di comunicazione I^2C. Una volta acquisito il segnale si procede al calcolo dei parametri di calibrazione del cantilever per poi trasmetterli al livello di presentazione (Client).

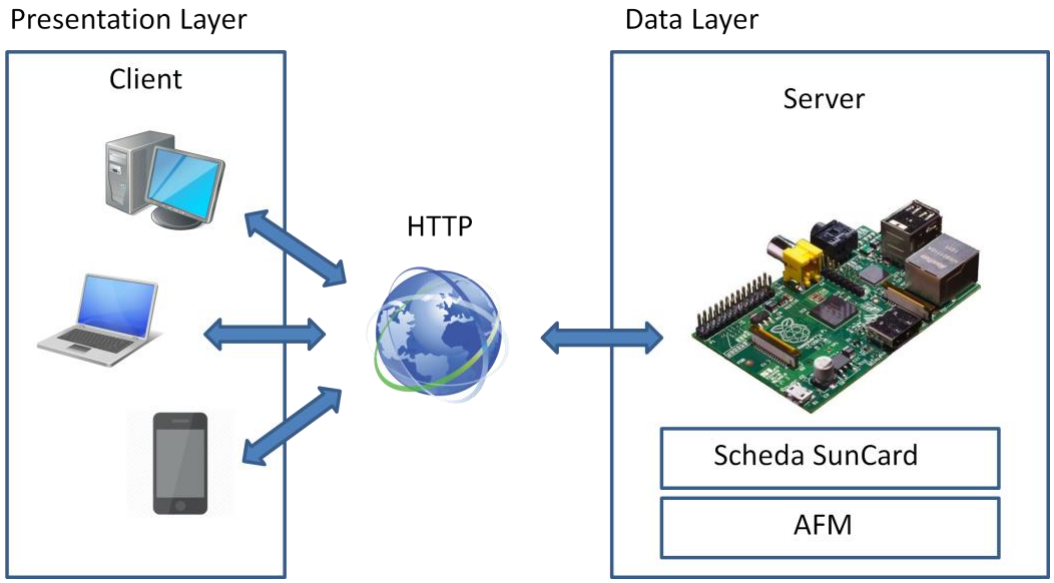

Figura 23: Architettura two-tier dell'applicativo.

Di seguito l'architettura hardware/software del sistema:

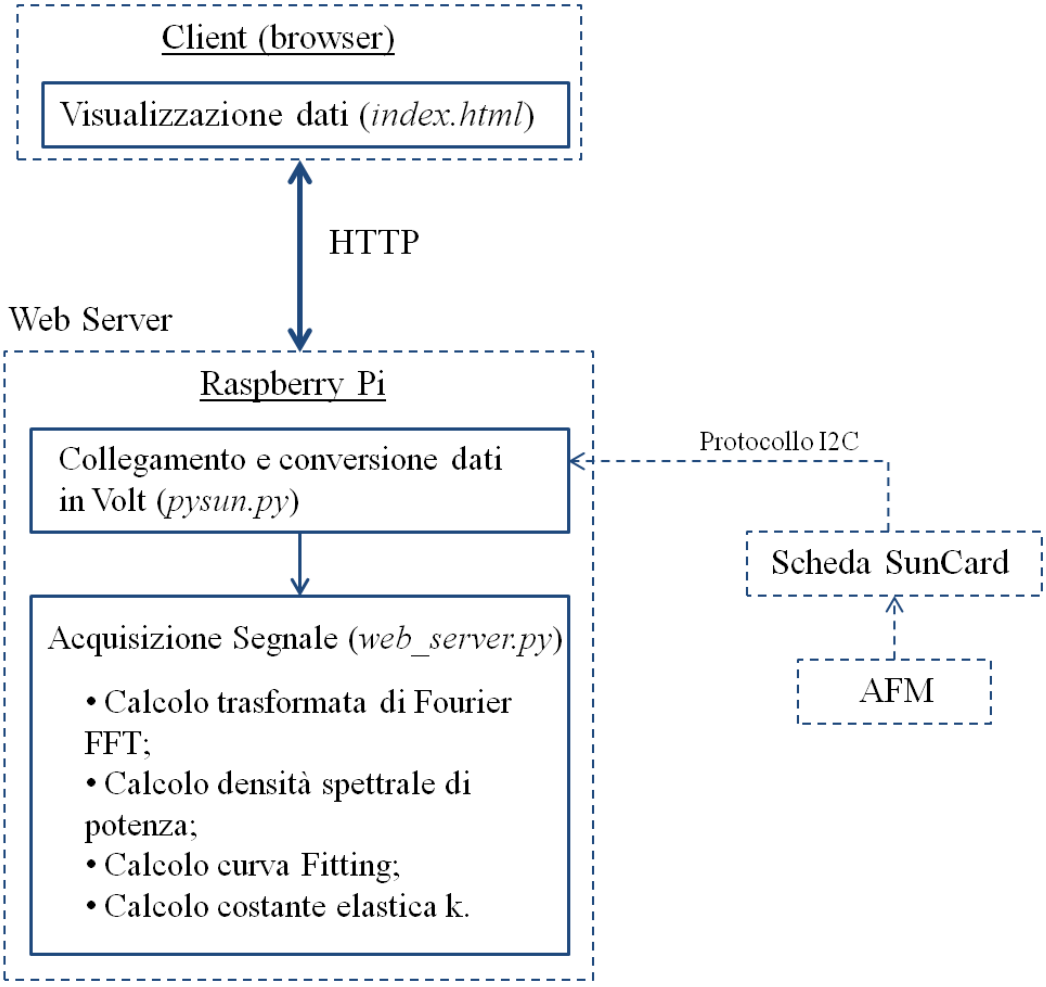

Figura 23 bis: Architettura hardware/software del sistema.

L'implementazione del blocco relativo al collegamento e conversione dati in Volt viene approfondito nella sezione 6.1.

Per il blocco relativo all'acquisizione del segnale e calcoli matematici si rimanda alle sezioni 6.2-6.7

Per quello di visualizzazione dati, interfaccia grafica, si rimanda in appendice D3.

6. Implementazione dell'applicazione

In questa sezione viene analizzato il progetto dell'applicazione dal punto di vista implementativo.

Viene analizzato il codice *web_server.py* che è servito per l'acquisizione del segnale e per il calcolo dei parametri di calibrazione.

Viene anche analizzato il codice *pysun.py* della scheda SunCard, fornito dalla ElbaTech s.r.l., con le opportune modifiche necessarie per il suo funzionamento.

6.1 Codice pysun.py della scheda SunCard

Prima di eseguire il codice, occorre scaricare la libreria pylibftdi (Python-FTDI, vedi capitolo 3). Successivamente, per permettere il collegamento del Raspberry Pi con la scheda SunCard, si deve specificare il Device FTDI:

Il tipo di Device è: "FTDI:FT245R USB FIFO:AD021LOS" con id= "AD021LOS". A questo punto, apriamo l'idle Python2.7, una volta acceso il Raspberry PI, lo inseriamo nel codice del file pysun.py, come evidenziato in verde nella figura 24:

```
class suncard():
    def __init__(self):
        try:
            #print "Apro collegamento con la Scheda"
            self.device = ftdi.Device(device_id='AD021LOS')
        except:
            #print "Errore di Collegamento con la Scheda"
            if logging:
                print sys.exc_info()[0]
    ....
```

Figura 24: Inserimento tipo di Device FTDI.

Analizzando brevemente il codice, in particolare la classe *suncard()*, e guardando la figura 25, notiamo tre principali comandi di acquisizione dei dati di nostro interesse:

```
# --> 1-Indicare i dati di nostro interesse
r = self.device.write(chr(CMD_SETNDATA ))
r = self.device.write(chr(lsb ))
r = self.device.write(chr(msb ))
buf = struct.unpack('<B',self.device.read(1))

if buf[0] != CMD_SETNDATA:
    print('Suncard error')
    return False

# 2-Send the start acquisition command
# --> 2-Inviare il comando di acquisizione
r = self.device.write(chr(CMD_ACQUIRE_NDATA ))
buf = struct.unpack('<B',self.device.read(1))

if buf[0] != CMD_ACQUIRE_NDATA:
    print('Suncard error')
    return False

# 3-retrieve the data from the board
# --> recuperare i dati dalla scheda
r = self.device.write(chr(CMD_SEND_NDATA ))
```

Figura 25: Comandi di acquisizione dati.

- CMD_SETNDATA: si dichiarano il numero dei punti che si desiderano campionare, acquisendoli dall'AFM;
- CMD_ACQUIRE_NDATA: indica il comando di Start, la scheda è pronta, per l'acquisizione dei dati;
- CMD_SEND_NDATA: recupera i dati, in base al numero dei punti dichiarati nel comando CMD_SETNDATA.

Una volta acquisiti i dati, vengono memorizzati in una lista (array), uno ad uno, tramite il metodo *append*, fino al numero dei punti desiderati e che si vogliono acquisire. Il metodo *append* ha la funzionalità di aggiungere i valori nella lista *voltarray* (figura 26).

```
i = 0
voltarray = []
start = time.time()
while(len(voltarray)<nData):
    buf = self.device.read(2)
    if buf != '':
        if len(buf)==1:
            print 'Suncard error! Requested buffer was too high'
            break
        try:
            buf = struct.unpack('<BB',buf)
            voltarray.append(self._getVolt(buf[0],buf[1]))
        except:
            print 'Risultato inatteso'
            print 'at N={0}'.format(len(voltarray))
            break
    elif len(voltarray)>0:
        ora = time.time()
        if (ora-start)>READ_TIMEOUT:
            print "Timeout of the acquisition at {0}"
            print "data".format(len(voltarray))
            break
return voltarray
```

Figura 26: Memorizzazione dati acquisiti.

Questi valori prima di essere memorizzati nella lista *voltarray* vengono convertiti in Volt, tramite la funzione:

_getVolt(self, msb, lsb=None)

come in figura 27:

```
def _getVolt(self,msb,lsb=None):
    if lsb == None:
        (msb,lsb)=msb
    word = msb * 256 + lsb
    if (msb >= 0x80) :
        volt = (HWRANGE/2.0) - (BITSTEP * (word - 32768))
    else:
        volt = -BITSTEP * word
    return volt

def get_value(self):
    r = self.device.write(chr(CMD_CONVERT))
    buf = struct.unpack('<BBB',self.device.read(3))
    if buf[0] != CMD_CONVERT:
        print('Suncard error')
        return False
    return self._getVolt(buf[1],buf[2])
```

Figura 27: Conversione in Volt dei dati acquisiti.

La funzione *get_value(self)* restituisce il segnale, dato dal numero dei punti specificati nel comando CMD_SETNDATA, dallo strumento AFM.

Per verificare il corretto valore ricevuto dalla scheda, in volt, si è utilizzato un alimentatore da Banco (generatore di tensione) ed un Multimetro Digitale.

Analizzando i valori, variando la tensione tra 0 e 6 V, tra quelli letti dal multimetro e quelli del codice si è notato una piccola discrepanza di differenza dell'ordine del mV. Successivamente, si è ricavato la retta per trovare la correzione adatta per una corretta calibrazione di conversione in Volt del codice. Dalla retta, di figura 28, si ricava:

$$voltarray = 0.0283 + 0.903\ X_F$$

dove:

$$X_F = (voltarray - 0.0283) / 0.903$$

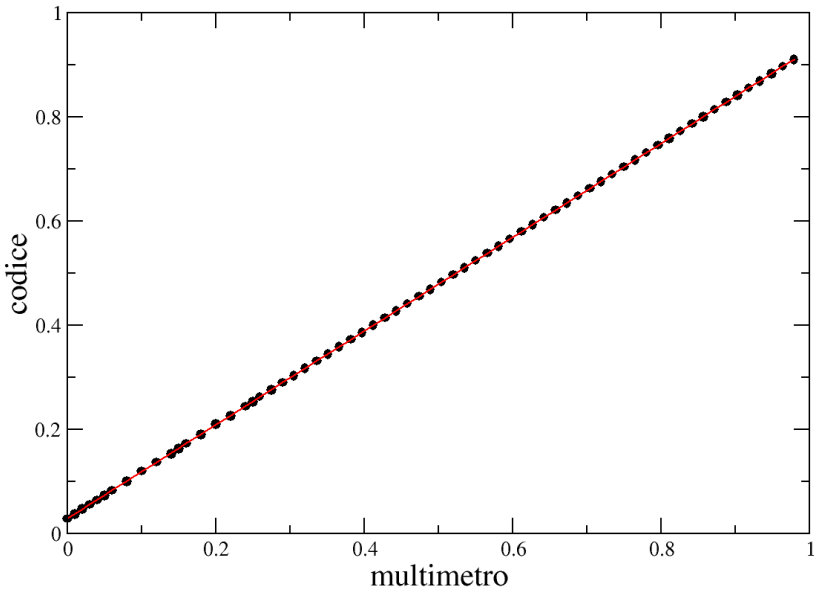

Figura 28: Confronto Valori ricavati da un multimetro digitale e il sw.

In figura 29 troviamo il codice corretto:

```
i = 0
voltarray = []
start = time.time()
while(len(voltarray)<nData):
    buf = self.device.read(2)
    if buf != '':
        if len(buf)==1:
            print 'Suncard error! Requested buffer was too high'
            break
        try:
            buf = struct.unpack('<BB',buf)
            voltarray.append(self._getVolt(buf[0],buf[1]))
        except:
            print 'Risultato inatteso'
            print 'at N={0}'.format(len(voltarray))
            break
    elif len(voltarray)>0:
        ora = time.time()
        if (ora-start)>READ_TIMEOUT:
            print "Timeout of the acquisition at {0}"
            print "data".format(len(voltarray))
            break
voltarray = np.array(voltarray, float)
voltarray = (voltarray - 0.0283)/0.903
return voltarray
```

Figura 29: Correzione del voltaggio.

In questo modo abbiamo il corretto voltaggio acquisito.

Inoltre, il range di funzionamento della scheda è limitato a ± 1V. Un range ideale quando il laser dell'AFM risulti ragionevolmente allineato rispetto ai quattro quadranti (fotodiodo).

In appendice D1 si può visualizzare il codice completo.

6.2 Acquisizione Segnale

Per l'acquisizione del segnale, creo un nuovo file *web_server.py* con una nuova funzione: *get(self)*, vedere figura 30:

```
def get(self):
    #Data Signal
    #Prelevo il segnale dall'AFM, collegandomi alla SunCard.
    #Acquisisco 1000 punti e li invio al web server per il plot
    #del grafico
    p1 = suncard()
    numPoints=1000
    xs=np.arange(numPoints)
    ys = np.array(p1.get_values(numPoints), dtype=float)
    ys=np.roll(ys,-1)
    data = [ list(a) for a in zip(xs, ys)]
```

Figura 30**:** codice implementato per l'acquisizione del segnale.

Apro il collegamento con la classe *suncard()*, definita nel paragrafo 6.1, dichiarando il numero dei punti che si desiderano acquisire e li memorizzo in una variabile xs, tramite la funzione *arange* di *numpy*. La funzione serve per creare un array con valori equispaziati tra di loro, nel nostro caso di 1 punto. Di seguito, in figura 31, un esempio:

```
>>> xs = numpy.arange(numPoints)
>>> print xs
array([0, 1, 2, 3, … , 999])
```

Figura 31: Esempio del metodo arange di numpy.

Successivamente, memorizzo in una nuova variabile, *ys*, i valori acquisiti tramite la classe SunCard, con la funzione *get_values()*, come vettore array di tipo float.

Ed infine memorizzo i dati di *xs* e *ys*, secondo il formato specificato nelle API di riferimento di Flot, in una variabile *data*, in modo da poter visualizzare il grafico in un sistema di assi cartesiani. Il formato dati è il seguente:

```
[ [x1, y1], [x2, y2], ... ]
```

Possiamo notare che si tratta di una matrice di punti.

6.3 La Trasformata discreta di Fourier

Nel metodo scelto per la calibrazione della costante k, lo spettro del segnale di deflessione ha un ruolo centrale. E' infatti relativamente facile trovare l'espressione teorica per lo spettro delle fluttuazioni termiche di un oscillatore armonico (cantilever)[15] ed estrarre dalla sua forma le informazioni rilevnti per il nostro sistema. Da un punto di vista numerico è quindi necessario calcolare lo spettro di potenza del segnale acquisito e per fare questo si può utilizzare la trasformata di Fourier. Jean Baptiste Joseph Fourier[16] (1768-1830), era un matematico e fisico francese che, presentando i suoi esperimenti sulla propagazione del calore, nella città di Grenoble, affermava che qualunque segnale periodico – continuo potesse essere rappresentato come somme pesate di sinusoidi.

Questa teoria di Fourier fu molto contrastata da Joseph Louis Lagrange[17], il quale sosteneva che per segnali contenenti punti angolosi, ad esempio le onde quadre, gli studi di Fourier perdevano significato e per 15 anni, fino alla morte di Lagrange, il lavoro di Fourier non fu più pubblicato. In realtà l'approccio di Fourier non era sbagliato ed anche l'obiezione di Lagrange è valida: se da un lato è vero che un segnale contenente angoli non potesse essere rappresentato come somme di sinusoidi, si riesce lo stesso ad ottenere una approssimazione così accurata del segnale in modo da avere

[15] N.A. Burnham, X Chen, C.S. Hodges, G.A. Matei, E.J. Thoreson, C.L. Roberts, M.C. Davies and S.J.B. Tendler. *Comparison of Calibration methods for atomic-force microscopy cantilevers*. 2002.
[16] Jean Baptiste Joseph Fourier. Wikipedia, 2015.
[17] Joseph Louis Lagrange. Wikipedia, 2015. https://it.wikipedia.org/wiki/Joseph-Louis_Lagrange

una differenza pari a zero. Questo fenomeno è conosciuto come il fenomeno di Gibbs[18].

Esistono quattro tipi di segnali possibili, ognuno con la sua trasformata di Fourier, e vengono suddivisi in: segnale periodico continuo o discreto, segnale aperiodico continuo o discreto.
I quattro tipi di trasformate sono:

1) Aperiodico Continuo: Il segnale non ha periodo e assume infiniti valori. Per la sua analisi si utilizza la Trasformata di Fourier.
2) Periodico Continuo: Il segnale assume infiniti valori che si ripetono con un certo periodo. In questo caso si usa la Serie di Fourier.
3) Aperiodico Discreto: Il segnale non ha periodo e assume solo un numero infinito di valori. In questo caso si utilizza la Trasformata di Fourier a tempo discreto.
4) Periodico Discreto: Il segnale assume un numero infinito di valori che si ripetono con un dato periodo. La trasformata utilizzata è la Trasformata discreta di Fourier.

Ognuno dei quattro tipi di segnale usato per queste trasformate deve essere di lunghezza infinita.
Nel nostro progetto, viene utilizzata la versione discreta della trasformata di Fourier.
L'operazione che trasforma un segnale nel dominio del tempo ad uno nel dominio delle frequenze si chiama DFT: trasformata discreta di Fourier. Esiste anche l'operazione inversa e si chiama trasformata inversa di Fourier. Entrambe le operazioni possono essere espresse come algoritmi ed utilizzate da elaboratori.

[18] Fenomeno di Gibbs. Wikipedia, 2015. https://it.wikipedia.org/wiki/Fenomeno_di_Gibbs

Quando si deve utilizzare un elaboratore per eseguire calcoli, è importante valutare l'efficienza dell'algoritmo da utilizzato e cioè valutare quanto tempo impiega il nostro algoritmo per eseguire il calcolo e di quanto cresce il tempo di esecuzione all'aumentare dei dati inseriti in ingresso. Il calcolo della DFT tramite correlazione richiede un tempo di esecuzione pari a: $k_{DFT*}N^2$. Dove k_{DFT} è una costante di proporzionalità che per un processore a 100Mhz equivale a 7 microsecondi. Ad esempio, se consideriamo un segnale di N = 1024 punti il tempo di esecuzione è di circa 7 secondi. Questo tempo è enorme, soprattutto considerando la piccola dimensione del segnale. Un algoritmo più efficiente è la FFT: Fast Fourier Transform, un algoritmo che riesce ad ottenere un tempo di esecuzione pari a $k_{FFT*}N*log2N$. In questo caso, k_{FFT} equivale a 10 microsecondi su processore a 100Mhz, quindi un segnale di N = 1024 punti viene trasformato in 70 millisecondi.

Comparando i due tempi, DTF e FFT, si capisce che la FFT permette di effettuare calcoli anche su segnali di dimensioni abbastanza grandi, poichè la crescita logaritmica assicura che il tempo di calcolo non aumenti troppo bruscamente all'aumentare dei punti.

Questo algoritmo di FFT, attribuito a Cooley e Tuckey, venne scoperto ed usato da Gauss nel XIX secolo.

In questo progetto viene sfruttata la libreria di Numpy (vedere capitolo 3), per il calcolo della FFT.

La FFT y[k] di lunghezza N della sequenza x[n] è definita come

$$\boldsymbol{y[k]} = \sum_{\boldsymbol{n=0}}^{\boldsymbol{N-1}} \boldsymbol{e^{-2\pi j\frac{kn}{N}} x[n]} \tag{6.1}$$

Nel codice, in figura 32, lo vediamo come segue:

>>> np.fft.fft(x)

Dove x è il vettore contenente i dati, prelevati dalla SunCard, di 1000 campioni, specificati nel paragrafo 6.1

```
#FFT
media = (sum(x) / len(x))
x = x - media
yf_current = np.fft.fft(x)
yf = yf + np.fft.fftshift(yf_current)
counter = counter +1.
yplot = yf/counter
```

Figura 32: implementazione della Fast Fourier Transform.

In figura 33 è possibile visualizzare un esempio di segnale acquisito dalla scheda *SunCard* (in blu), e la sua *FFT* (in arancione).

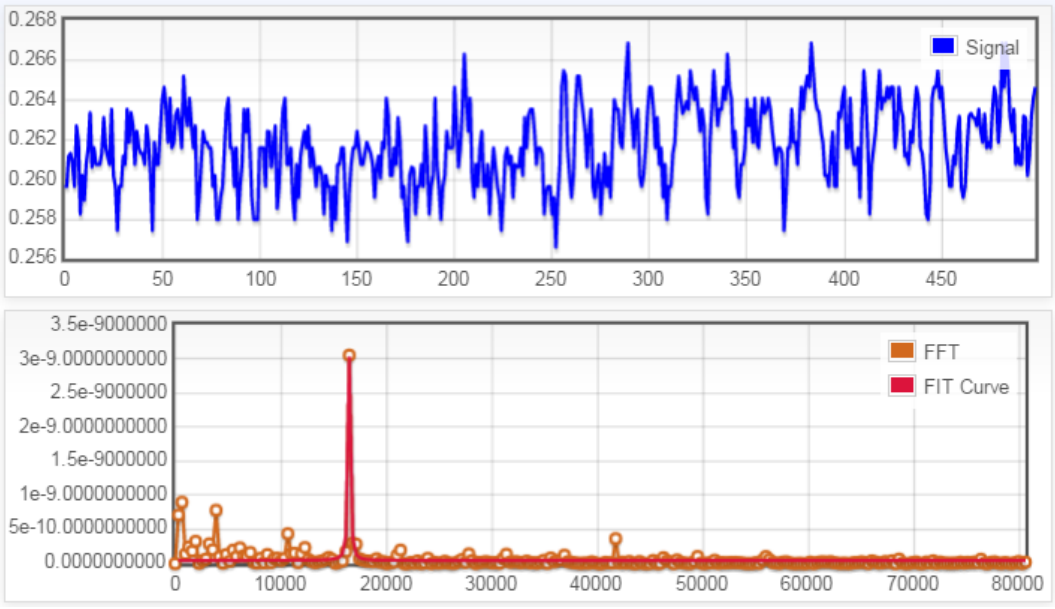

Figura 33: Segnale Sinusoidale e rispettiva FFT.

6.4 Densità Spettrale di potenza

Lo scopo di questo paragrafo è di introdurre il concetto di densità spettrale di potenza. Si tratta della trasformata di Fourier dell'autocorrelazione, con alcune precisazioni nel caso in cui tale trasformata non sia una funzione tradizionale. [19]

Per una funzione deterministica, la trasformata di Fourier può essere utile per molti scopi assai diversi. Uno di questi è che il suo modulo descrive il contributo dato alla funzione dalle varie frequenze, ed il modulo quadro descrive il contributo dato dalle varie frequenze all'energia totale. E' interessante sapere che in un segnale sono più attive certe frequenze piuttosto che altre; ad esempio questo può mettere in rilievo la parte di rumore rispetto al segnale puro. Nell'analisi di un fluido, sapere che certe frequenze danno un contributo maggiore di altre all'energia ci dice quali scale spaziali sono maggiormente attivate, fatto che magari sarebbe impossibile rilevare da una visualizzazione del campo di velocità. Queste sono alcune delle motivazioni per l'interesse verso lo spettro di Fourier (il modulo della trasformata) e lo spettro dell'energia (il modulo quadro).

Se ora abbiamo a che fare con un processo stocastico stazionario, le stesse considerazioni sono interessanti. L'energia delle realizzazioni tipiche è infinita, per cui diventa rilevante l'energia per unità di tempo, ovvero la potenza. Quindi ci poniamo le stesse domande precedenti, relativamente alla potenza: in quale misura contribuiscono le varie frequenze alla potenza?

Da un punto di vista un po' sperimentale, se abbiamo un segnale che presupponiamo proveniente da un processo stazionario, lo campioniamo ad intervalli regolari, su un intervallo di tempo finito, e ne calcoliamo la FFT. Ciò che otteniamo è un'approssimazione della distribuzione F(f) (si pensi ad F(f) come una sorta di trasformata di Fourier del segnale in esame).

[19] Power Spectrum. https://en.wikipedia.org/wiki/Spectral_density

In un opportuno limite, in cui il segnale è campionato con maggior frequenza e su un intervallo sempre più lungo, ci aspettiamo che il risultato della FFT (opportunamente scalato) tenda alla distribuzione F(f). Calcolando il modulo quadro del risultato della FFT, abbiamo una prima indicazione approssimata del contributo dato da certe frequenze. In questo modo stiamo calcolando in modo approssimato $|F(f)|^2$, che è il contributo del segnale a frequenza f all'energia, ed è infinito, come l'energia è infinita. L'approssimazione data dalla FFT è finita, ma se raffiniamo il procedimento, al limite otterremo infinito. Comunque non è difficile trovare un modo per rinormalizzare i risultati della FFT in modo da avere un limite finito (che non sarà $|F(f)|^2$). Questo limite finito è proporzionale al contributo dato dalla frequenza f alla potenza.

Le figure 34 a) e b) mostrano un segnale aleatorio e l'approssimazione di $|F(f)|^2$ ottenuta tramite FFT.

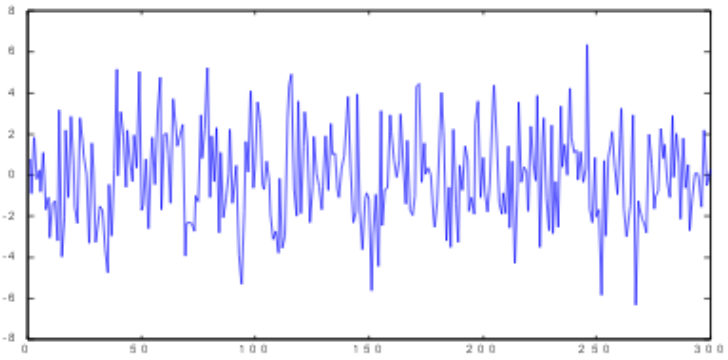

Figura 34: a) un segnale aleatorio x(t).

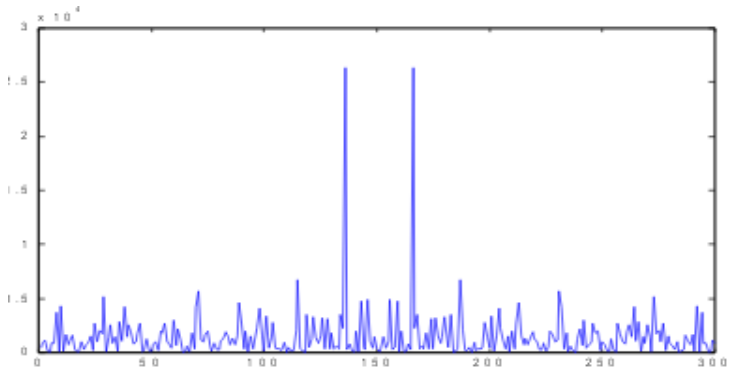

Figura 34: b) Modulo quadro della FFT della x(t).

Il segnale è stato ottenuto numericamente come sovrapposizione di un segnale deterministico con una certa frequenza f_0 più un white noise. Lo spettro ottenuto tramite FFT mostra chiaramente l'importanza della frequenza f_0. Le altre frequenze sono presenti, in modo piuttosto irregolare, però con ampiezza di poco variabile intorno ad una costante, questo è collegato al fatto che la densità spettrale di potenza di un white noise è costante.

Il procedimento fino ad ora illustrato è immediato e naturale avendo a disposizione un software, ma conduce ad oggetti alquanto complicati. Il processo generalizzato F(f) è teoricamente un oggetto complicato; è aleatorio, cioè dipende dalla realizzazione: per una singola realizzazione descrive ciò che vogliamo, ma non ha valore per un'altra, quindi ad esempio averne una registrazione in base ad un esperimento non ci dice molto dell'esperimento successivo; ed infine, come si vede dalla figura 4.9 b), ha un andamento al variare di f molto irregolare, non corrisponde ad una funzione analiticamente semplice e non si presta pertanto a vari tipi di ragionamenti a cui invece siamo abituati per segnali deterministici. Ad esempio, per questi ultimi ci sono vari esempi basilari di cui conosciamo analiticamente lo spettro, e questi esempi servono spesso come modelli o da guida nell'analisi di problemi più complessi. Per queste ragioni si introducono delle grandezze medie, meno precise per svolgere un'analisi spettrale delle singole realizzazioni, ma analiticamente più utili e che descrivono proprietà dell'intero processo.

Nel nostro caso, il calcolo della densità spettrale di potenza, normalizzato, ci permetterà di ottenere un'ampiezza normalizzata dello spettro ed è dato da:

$$\frac{\|F_T(f)\|^2}{N^2} \tag{6.2}$$

In figura 35 la sua implementazione:

```
#Power Spectrum
yplot = (np.abs(yplot)**2) /(numPoints **2)
xf = np.fft.fftfreq(numPoints, 1.)
xf = np.fft.fftshift(xf)
k = yplot.argmax()
valmax x = np.abs(xf[k])   ----→ Frequenza
```

Figura 35: implementazione della Power Spectrum.

6.4.1 Determinazione del rate di acquisizione

Per verificare il corretto valore della frequenza del software si è utilizzato un oscilloscopio e un generatore di funzioni. La frequenza è stata fatta variare tra 5-15KHz. Successivamente, si è provveduto ad apportare la giusta correzione per una corretta calibrazione di conversione in Hz del codice. Dalla formula:

$$r = \frac{F_{osc}}{f} \tag{6.3}$$

dove F_{OSC} è la frequenza letta dall'oscilloscopio ed f la frequenza letta dal codice. Il rapporto è la correzione da apportare. Nel caso in cui l'oscilloscopio si trovi ad una frequenza di 11KHz, il codice risulta essere ad una frequenza di 0.068 Hz, per cui si ha:

$$r = \frac{11KHz}{0{,}068\ Hz} = 161.764\ \ sample/sec \tag{6.4}$$

Per cui l'implementazione della Power Spectrum di figura 35, con la giusta correzione, diventerà come mostrato in figura 35bis:

```
#Power Spectrum
yplot = (np.abs(yplot)**2) /(numPoints **2)
xf = np.fft.fftfreq(numPoints, 1.)
xf = np.fft.fftshift(xf) * 161764.
k = yplot.argmax()
valmax x = np.abs(xf[k])   ----→ Frequenza
```

Figura 35 bis: Power Spectrum corretta.

6.5 Curve Fitting

Il *curve fitting*[20] viene definito, da wikipedia, come:

"il processo di costruzione di una curva o di una funzione matematica, che abbia la migliore corrispondenza ad una serie di punti assegnati, possibilmente soggetti a limitazioni."

Per la costruzione della nostra curva si è implementato il seguente codice, in figura 36 a), che fa riferimento al modello matematico descritto nel capitolo 1, formula (1.8) :

```
#Formula Curve Fitting
def func(xdata2, nu_k, A, B, x2_nu_k, Q):
    y = A/xdata2
    y = y+ B
    y1=x2_nu_k / (Q*Q)
    y2=1- (xdata2/nu_k) * (xdata2/nu_k)
    y3=xdata2/(nu_k * Q)
    y=y+y1/(y2*y2+y3*y3)
    return y
```

Figura 36: a) implementazione della curva in base al modello matematico della formula (1.8).

[20] https://it.wikipedia.org/wiki/Curve_fitting

```
#Curve FIT
ypopt = func(xdata2, nu_k, A, B, x2_nu_k, Q)
popt, pcov = curve_fit(func, xdata2, ydata2,p0=[nu_k, A, B, x2_nu_k, Q])
ypopt = func(xdata2, *popt)
dataFIT = [ list(a) for a in zip(xdata2[numPoints/2:], ypopt[numPoints/2:])]
nu_k_fit = popt[0]
A = popt[1]
B = popt[2]
x2_nu_k_fit = popt[3]
Q = np.abs(popt[4])
method_k = ((2*kb*T*Q*delta_nu) / (np.pi * nu_k_fit * x2_nu_k_fit))
method_k = float(method_k)* (10**+18)
```

Figura 36: b) esecuzione del curve fit.

Successivamente, si richiama la funzione *func* di figura 36 a) e si esegue il *curve_fit* come in figura 36 b) richiamandola tramite la libreria *scipy*, descritta nel paragrafo 3.4

6. 7 Implementazione del calcolo della costante elastica k

Dopo aver descritto e implementato il codice, nel paragrafo 6.2, per il segnale da acquisire, nel paragrafo 6.3, per la FFT del segnale, nel paragrafo 6.4, per la densità spettrale di potenza e, nel paragrafo 6.5, per il curve fitting si può finalmente calcolare la costante di elasticità k. Per la sua implementazione si fa riferimento alla formula 1.10 del capitolo 1.
In figura 37:

```
method_k = ((2*kb*T*Q*delta_nu) / (np.pi * nu_k_fit * x2_nu_k_fit))
method_k = float(method_k)* (10**+18)
```

Figura 37: implementazione della costante k.

In appendice D2 si può visualizzare il codice completo.

7. Risultati ottenuti

7.1 Risultati

In questo paragrafo mostriamo i risultati ottenuti dallo sviluppo dell'applicazione Web. L'applicazione sviluppata fornisce all'utente la possibilità di poter calibrare il cantilever oltre a visualizzare determinate informazioni relative al suo spettro. Nel tempo l'applicazione è cresciuta e sono state aggiunte nuove ed interessanti funzionalità raggiungibili attraverso nuovi bottoni e nuove voci di menù. In questo capitolo verrà fornita una dettagliata spiegazione degli elementi che sono stati sviluppati per la realizzazione dell'interfaccia grafica, per fornire agli utenti che vogliono compiere la calibrazione un modo semplice ed efficace per poter interagire con l'applicazione.

7.2 Homepage

La figura 38 mostra la *homepage* dell'applicazione visualizzata e sempre a disposizione dell'utente. In essa sono presenti tutte le funzionalità descritte nei capitoli precedenti.

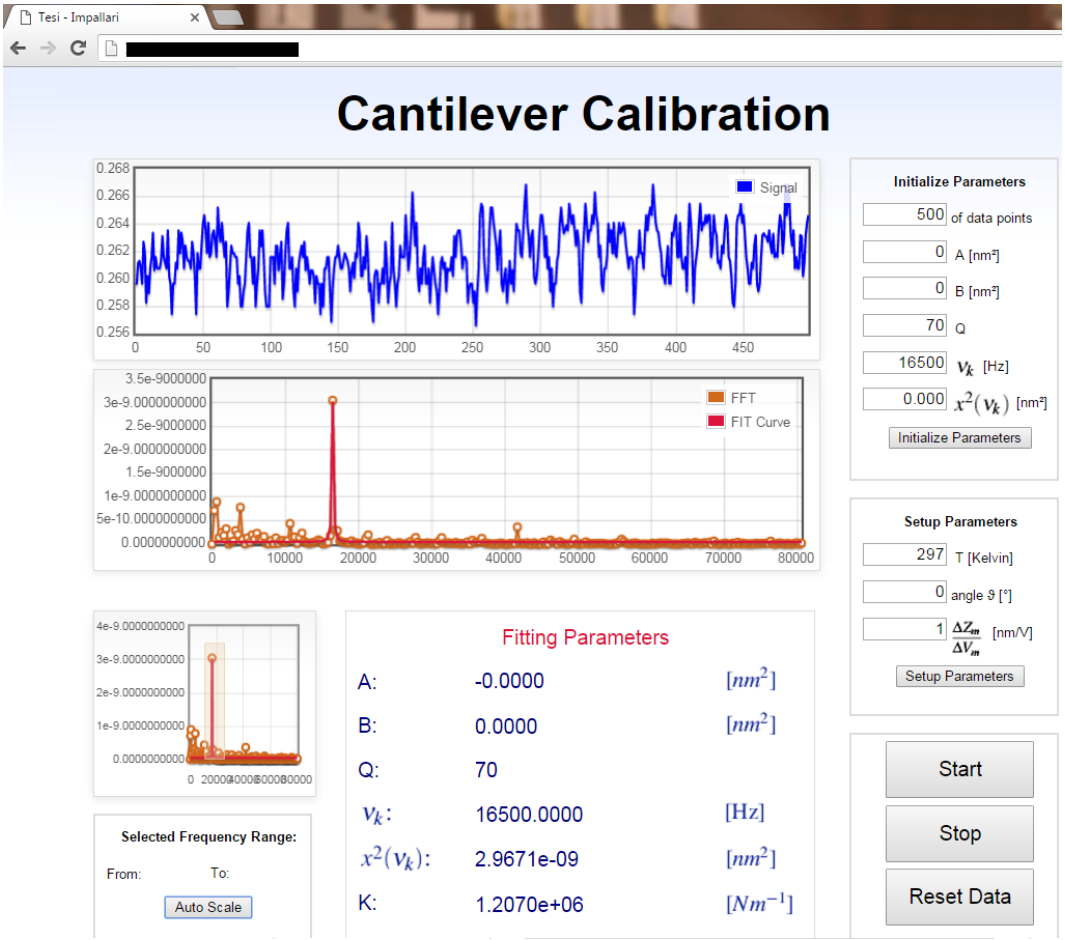

Figura 38: homepage della Web Application.

Possiamo vedere che l'applicazione di figura 38 è composta da diversi elementi principali:

- Due voci di menù, inizializzazione dei parametri e di setup parametri;

- Un grafico per la visualizzazione del segnale acquisito tramite la SunCard (che si occupa di ricevere i dati sfruttando il protocollo di comunicazione I^2C);
- Un secondo grafico per lo spettro di potenza, calcolato tramite una Fast Fourier Transform, e relativo fitting;
- L'interfaccia è anche dotata di un ulteriore grafico di overview per zoomare il range delle frequenze che si desidera visualizzare/fittare con una semplice selezione del mouse; Comprende anche un pulsante di auto scale per resettare lo zoom.
- Ed infine dei pulsanti di *Start*, *Stop* e *Reset Data*.

Di seguito verranno analizzate le funzionalità fornite e soprattutto come un utente normale, senza conoscenze informatiche, si approccerebbe al suo utilizzo.

1) Initialize Parameters, figura 39: Questo pannello serve ad inizializzare i parametri del fitting e il numero dei punti che si vogliono acquisire.

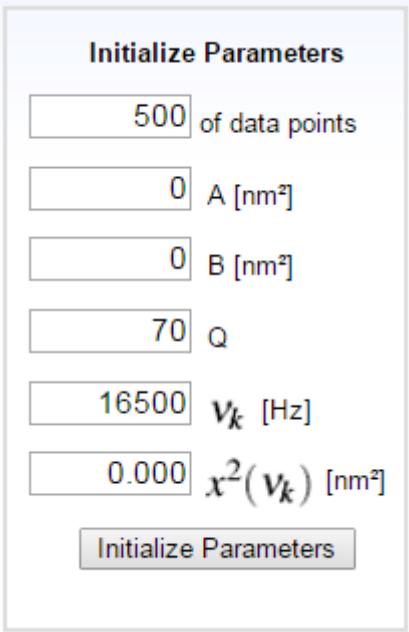

Figura 39: Pannello di inizializzazione parametri.

Elenco le voci della label:

- Number of data points: numero dei punti che si desiderano acquisire;
- A: ampiezza della componente $1/\nu$ in $nm^{2*}Hz$
- B: Spectrum Offset in nm^2
- Q: Fattore di Qualità
- ν_k: frequenza in Hz del picco massimo dello spettro
- $X^2(\nu_k)$: Fluttuazione in nm^2 al picco

2) Setup parameters, figura 40: in quest'altro pannello si settano i valori per determinare le proprietà del cantilever.

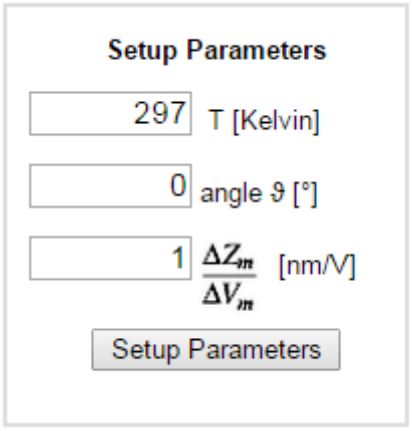

Figura 40: Setup parametri.

- T: Temperatura rilevata nell'AFM, tramite un misuratore portatile. Espressa in Kelvin.
- Angolo ϴ in gradi
- Rapporto $\frac{\Delta Zm}{\Delta Vm}$ in nm/V

3) Pulsanti dell'applicazione, figura 41. Il pulsante di start serve per eseguire l'applicazione e a sua volta per avviare lo scambio dei dati con la SunCard. Si noteranno i campioni acquisiti muoversi. Stop per fermare il flusso dei dati. Reset Data per svuotare la lista *data* contenente i dati accumulati durante la fase di campionamento del segnale.

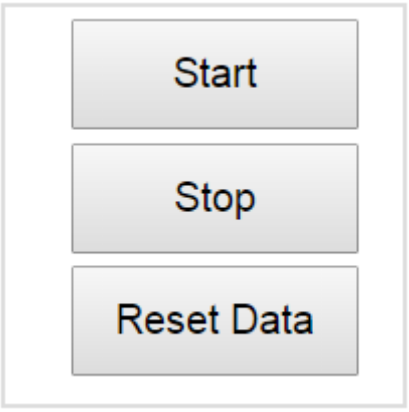

Figura 41: Pulsanti di avvio, stop esecuzione e reset valori.

5) Grafici di acquisizione, in figura 42, del suo spettro e relativo fitting in figura 43:

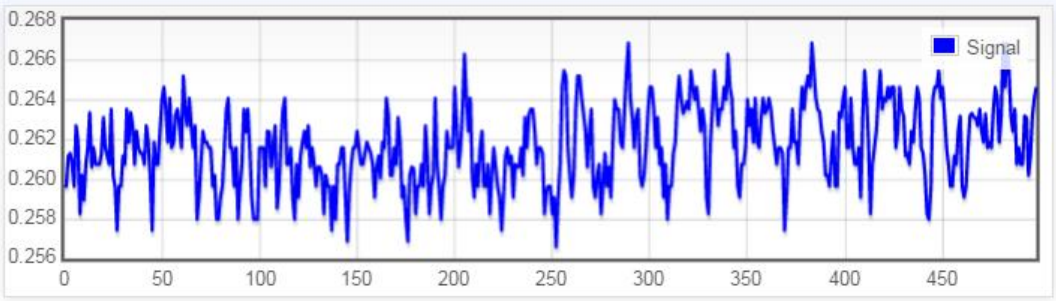

Figura 42: campionamento del segnale.

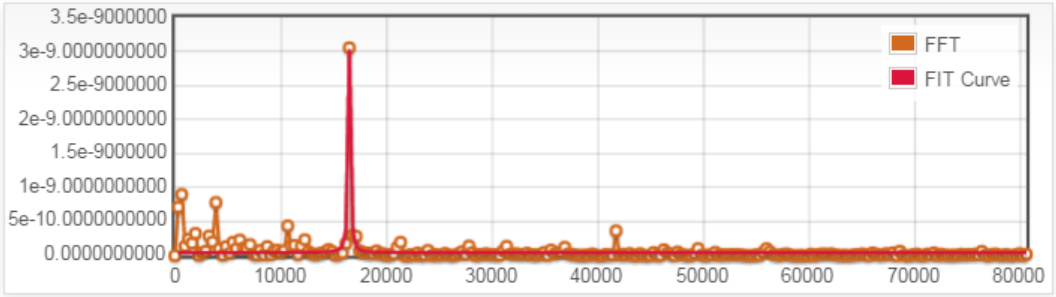

Figura 43: Spettro del segnale campionato e Fitting.

6) Pannello visualizzazione proprietà del cantilever, figura 44:

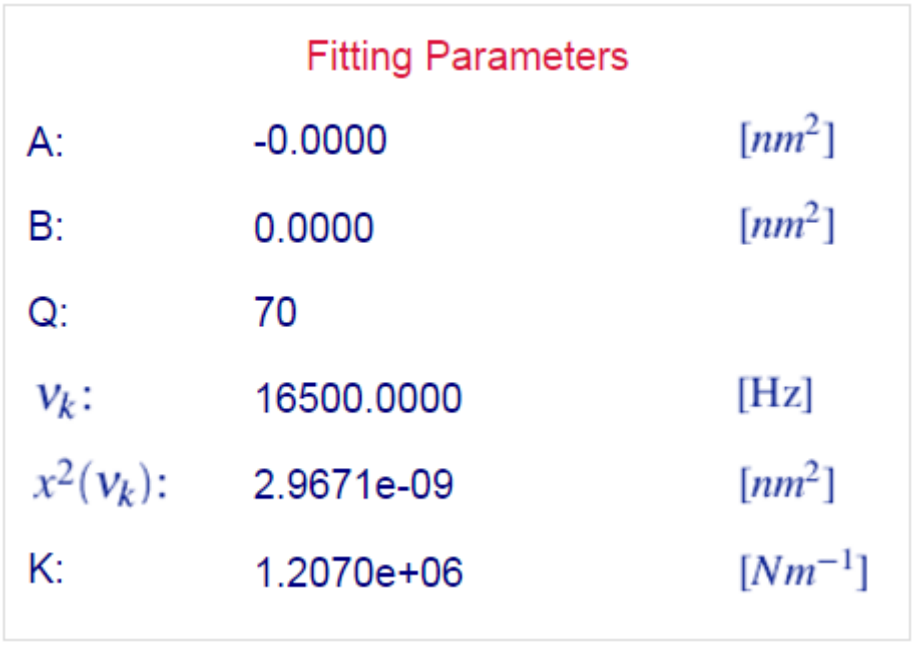

Figura 44: Proprietà meccaniche della leva, ricavate tramite fitting e relativo valore della costante elastica k.

Cliccando sulla voce *Fitting Parameters*, di figura 44, si aprirà una finestra di *popup* contenente la formula (1.8), in figura 45.

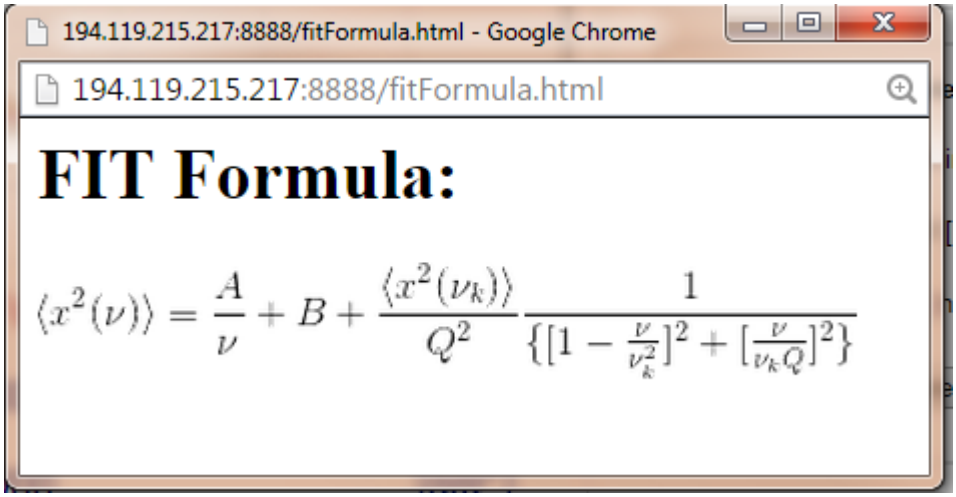

Figura 45: Formula per il FIT dei parametri.

Stessa cosa, in figura 46, cliccando sulla voce *K* di figura 44:

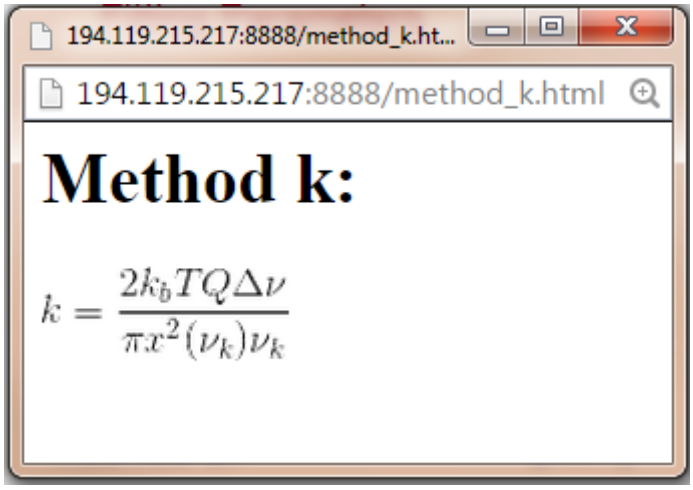

Figura 46: Formula per ricavare la costante k del cantilever.

6) Zoom con overview dello spettro: Con una semplice selezione del mouse si imposta il range di valori da zoomare, sull'asse delle frequenze e delle ampiezze, dello spettro di figura 43 e di visualizzare il range selezionato nel pannello *Selected Frequency Range* come in figura 47.

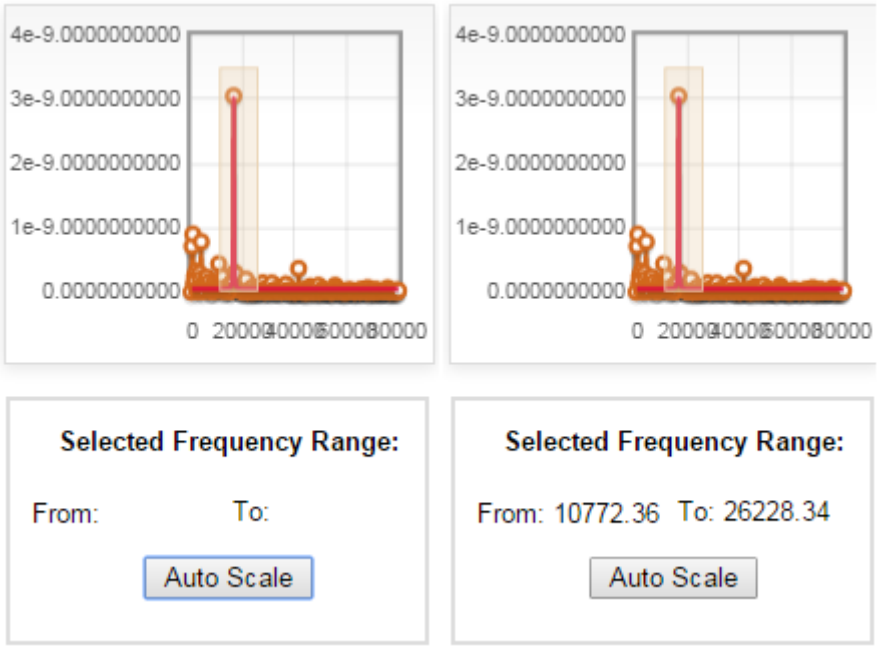

Figura 47: a) Overview senza zoom. b) Overview con range selezionato.

Successivamente, selezionato il range, come in figura 47 b), il grafico di figura 43 diventerà:

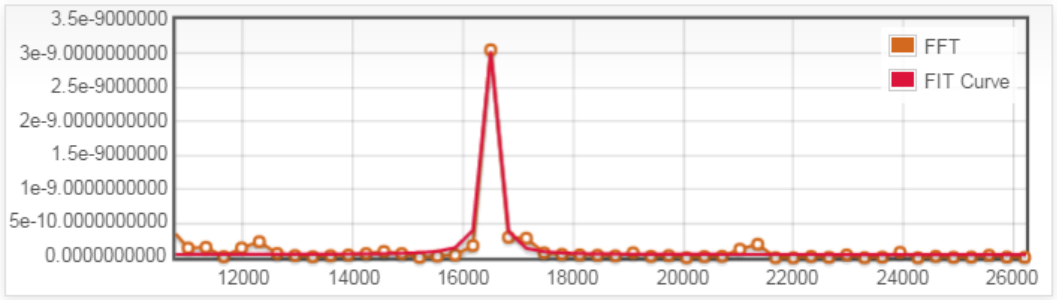

Figura 48: Spettro zoomato con i nuovi valori di range selezionati.

Il fitting viene effettuato sul range di frequenze selezionato.

Cliccando il pulsante *Auto Scale* il range verrà resettato e quindi lo spettro ritornerà come in figura 43.

A questo punto, una volta descritti i dettagli, possiamo notare che l'interfaccia grafica sia abbastanza minimale e intuitiva. Questo dovrebbe rendere il suo utilizzo e apprendimento molto semplice per l'utente.

Conclusioni

Nel presente lavoro, condotto presso il laboratorio dell'Istituto di BioFisica del C.N.R. di Palermo, sono state descritte le fasi del lavoro svolto, mirato alla progettazione e alla realizzazione del software per determinare il valore della costante di elasticità K del cantilever, da impiegare nel campo della microscopia a forza atomica.

I punti di forza di questo progetto sono: l'utilizzo di Python, ampiamente usato in ambito scientifico, per l'analisi e il calcolo matematico, per ottenere i risultati ricavati dai dati sperimentali per il calcolo della costante k; l'uso di software open source per lo sviluppo, abbattendo i costi e sfruttando la conoscenza diffusa dell'utenza dei vari tools; Il disaccoppiamento tra il sistema di acquisizione e processamento dei dati dall'interfaccia utente che può girare su un dispositivo grafico mobile come un tablet o uno smartphone; l'architettura hardware particolarmente leggera, portabile e di bassa potenza richiesta. . Il sistema può essere usato ovunque e da qualsiasi utente che desidera effettuare operazioni di calibrazione. Questo studio getta le basi per lo sviluppo di un sistema integrato e mobile di ausilio per dispositivi di microscopia a forza atomica,

Il sistema prototipo realizzato nel corso del progetto è basato sul Raspberry Pi, modello B, che è sufficiente ai nostri scopi ma che mostra i suoi limiti in termini di calcolo intensivo. Il sistema, inoltre, presenta un altro piccolo limite che sarebbe l'attesa che trascorre tra una richiesta dell'utente (client) e la risposta del Raspberry Pi (server), nonostante si è utilizzato Ajax, in cui lo scambio dei dati (tra client e server) avviene in background tramite una chiamata asincrona in modo da non ricaricare l'intera pagina.

Nulla vieta di effettuare più di una richiesta simultanea al server per operazioni differenti con o senza controllo da parte dell'utente.

Bisogna prestare attenzione alle richieste da inviare, l'utente potrebbe trovarsi in difficoltà qualora si dovesse aspettare una risposta che ritarda ad arrivare, e potrebbe non avere idea di cosa stia accadendo alla pagina, spesso inviando una nuova richiesta ignari di aver già richiesto un'informazione e sovraccaricando il server. Questo comportamento dovrebbe essere evitato, in quanto aggiunge richieste ad un server già saturo, che non riuscirà a smaltire il suo carico, continuando a lasciare l'utente in attesa di una risposta.

Nell'ottica di eventuali sviluppi futuri, sicuramente rivestirà una notevole importanza l'adozione del Raspberry Pi 2, introdotto all'inizio del 2015, che offre un sostanziale aumento delle performance rispetto alle versioni precedenti a parità di costi. Il nuovo modello monta infatti un quad-core da 900 MHz con 1GB di RAM rispetto a quello utilizzato in questo progetto che monta un single-core da 700 MHz e con 512MB di RAM. Ovviamente, con quest'ultimo modello, migliorerà anche il tempo di attesa per lo scambio dei dati tra Client e Server senza sovraccaricare troppo il Raspberry Pi.
Vista la velocità con cui la tecnologia si è evoluta nell'ultimo decennio, è arduo fare previsioni su quello che ci aspetterà fra altri 10 anni. Qualcosa è possibile immaginare. E' prevedibile che ci saranno nuovi modelli di Raspberry Pi sempre più potenti, più veloci e consumeranno meno energia.

Bibliografia e sitografia

G. Binnig, C.F. Quate, Ch. Gerber. *Atomic Force Microscope*. Physical Review Letters vol 56, n. 9, 3 Marzo 1986.

Wikipedia., *Microscopio effetto tunnel*. https://it.wikipedia.org/wiki/Microscopio_a_effetto_tunnel.

Bharat Bhusham. *Handbook of nanotechnology*. Springer, 2007. Pag. 591-608.

Victor L. Mironov. *Fondamenti di microscopia a scansione di sonda*. Accademia russa delle scienze, Istituto per la fisica delle microstrutture, 2004. (trad. it. [a cura di Giacomo Torzo]).

J. Hutter and J. Bechhoefer. *Calibration of atomic-force microscope tips*. Rev. Sci. Instrum. 64, 3342 (1993).

Butt and Jaschke M 1995, *Nanotechnology 6*. 1-7.

N.A. Burnham, X Chen, C.S. Hodges, G.A. Matei, E.J. Thoreson, C.L. Roberts, M.C. Davies and S.J.B. Tendler. *Comparison of Calibration methods for atomic-force microscopy cantilevers*. 2002.

Wikipedia, *Raspberry Pi*. https://it.wikipedia.org/wiki/Raspberry_Pi

Raspberry Pi, *Installazione S.O*. http://www.raspberrypi.org/wp-content/uploads/2012/12/Raspberry_Pi_-_Guida_veloce.pdf

Mark Lutz, *Learning Python*. O'Reilly, 2009.

Wikipedia, *Python*. https://it.wikipedia.org/wiki/Python

Wikipedia, *Nascita e Storia di Python*. 2015. https://it.wikipedia.org/wiki/Guido_van_Rossum

Wikipedia, *Numpy*. Wikipedia. http://www.numpy.org/

JSON. http://json.org/

Wikipedia, *Tornado Web Server* - https://en.wikipedia.org/wiki/Tornado_%28web_server%29

Applicazioni web-based. http://www.prometheo.it/a_applicazione_web.htm

Architetture software, http://www.dis.uniroma1.it/~prgapplsoft/lezioni/2-ArchitettureSoftware.pdf

Wikipedia, *Rich Internet.* https://it.wikipedia.org/wiki/Rich_Internet_application

Wikipedia, *Jean Baptiste Joseph Fourier*. 2015.

Wikipedia, *Joseph Louis Lagrange* . 2015. https://it.wikipedia.org/wiki/Joseph-Louis_Lagrange

Wikipedia, *Fenomeno di Gibbs*. 2015. https://it.wikipedia.org/wiki/Fenomeno_di_Gibbs

Density Spectral. https://en.wikipedia.org/wiki/Spectral_density

Curve fitting. https://it.wikipedia.org/wiki/Curve_fitting

Installazione S.O. Raspbian. http://www.raspberrypi.org/downloads/

Win32diskimager. https://launchpad.net/win32-image-writer/+download

Appendice A

Le seguenti istruzioni, per la configurazione della scheda SD, sono per utenti Windows.

1) Scaricare il sistema operativo per il Raspberry Pi: è possibile scegliere la distribuzione software che si desidera usare: sono presenti le tre principali sopraelencate e alcune altre scelte. Nel nostro caso scarichiamo il s.o. **Raspbian** – Debian Wheezy.[21]
2) Scompattare il file scaricato, del S.O. Raspbian – Debian Wheezy, contenente un file con estensione .img
3) Scaricare e decomprimere il file win32diskimager-binary.zip[22], nella stessa directory in cui è stato scompattato il file del S.O. Raspbian.
4) Collegare la scheda SD al PC, ed avviare il file Win32DiskImager.exe

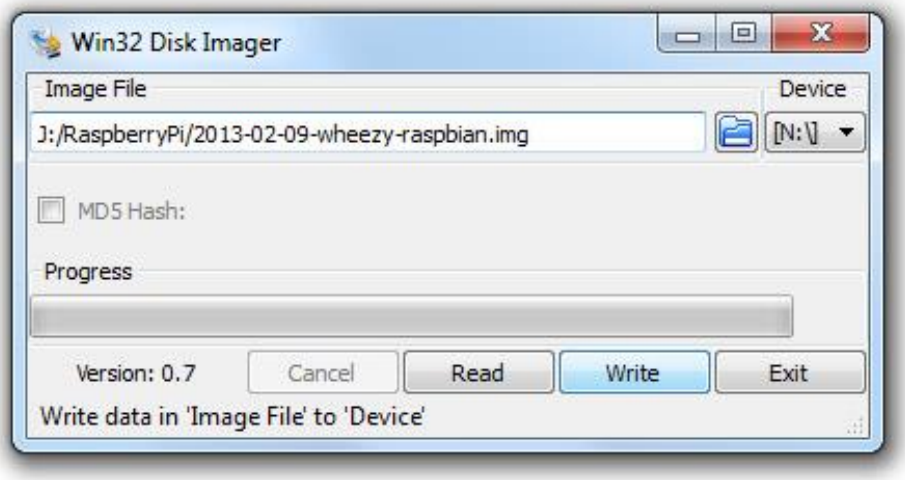

Figura 49: Win32DiskImager.exe per installare il S.O. Raspbian.

[21] Installazione S.O. Raspbian. http://www.raspberrypi.org/downloads/
[22] Win32diskimager. https://launchpad.net/win32-image-writer/+download

A questo punto andiamo a cercare il file .img appena estratto, controlliamo che la lettera sotto *Device* corrisponda a quella del lettore di SD, e iniziamo la scrittura con *Write*. Dopo qualche minuto la nostra scheda SD sarà pronta per essere inserita nello slot del Raspberry per poterlo avviare.

Già dal primo avvio viene visualizzata la schermata di configurazione "raspi-config" e vengono installati di default i driver essenziali per le periferiche usb di tastiera e mouse.
Dopo il riavvio del Rapsberry e le dovute configurazioni iniziali, è possibile attivare la modalità grafica LXDE, impostando il boot automatico.

LXDE (Lightweight X11 Desktop Environment) è un ambiente grafico libero, fig. 50, per il sistema operativo GNU/Linux creato con l'obiettivo di essere estremamente leggero.

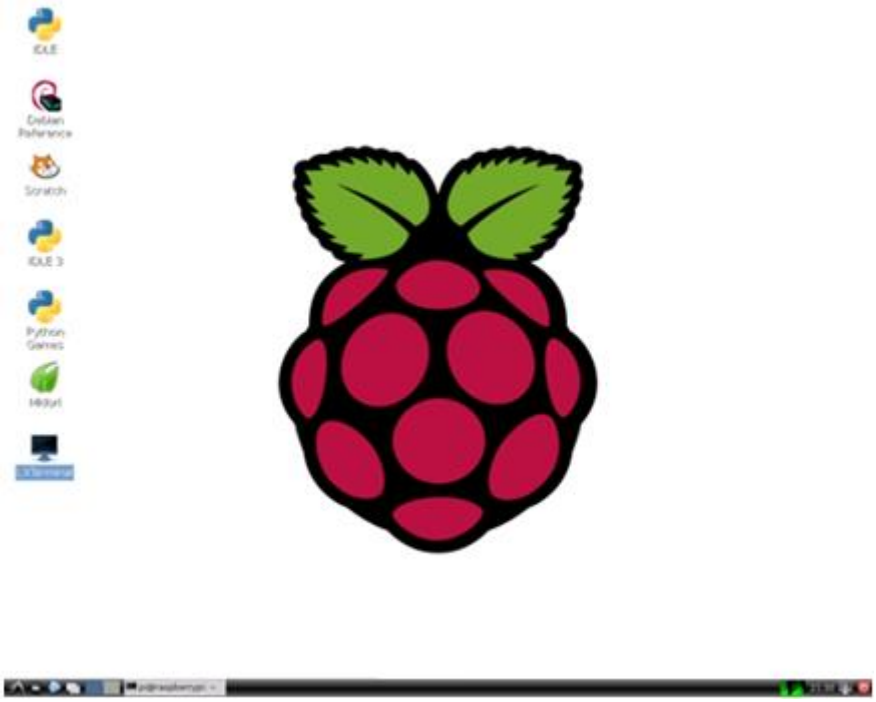

Figura 50: ambiente grafico LXDE.

Appendice B

Verrà presentata la procedura d'installazione della libreria open source python-ftdi per il Raspberry Pi.

Da terminale eseguire:

$ sudo apt-get install libftdi-dev

Successivamente:

1. Creare un file "/etc/udev/rules.d/99-libftdi.rules". Sarà necessario l'accesso come utente root per creare questo file.
2. Inserire le seguenti righe nel file:

    - SUBSYSTEMS=="usb",ATTRS{idVendor}=="0403", ATTRS{idProduct}=="6001",GROUP="dialout", MODE="0660"
    - SUBSYSTEMS=="usb",ATTRS{idVendor}=="0403", ATTRS{idProduct}=="6014",GROUP="dialout", MODE="0660"

Visualizzare sito web:
http://pylibftdi.readthedocs.org/en/latest/installation.html
per una guida completa all'installazione del pacchetto.

Verifichiamo che la libreria sia stato installata correttamente, connettendo la scheda SunCard al Raspberry Pi. Da terminale, digitiamo:

$ python2.7 -m pylibftdi.examples.list_devices

Se tutto va bene, il programma dovrebbe riferire informazioni su ogni dispositivo collegato.

Dalla figura 51 notiamo che il dispositivo “FTDI: FT245R USB FIFO: AD021LOS” risulta collegato correttamente.

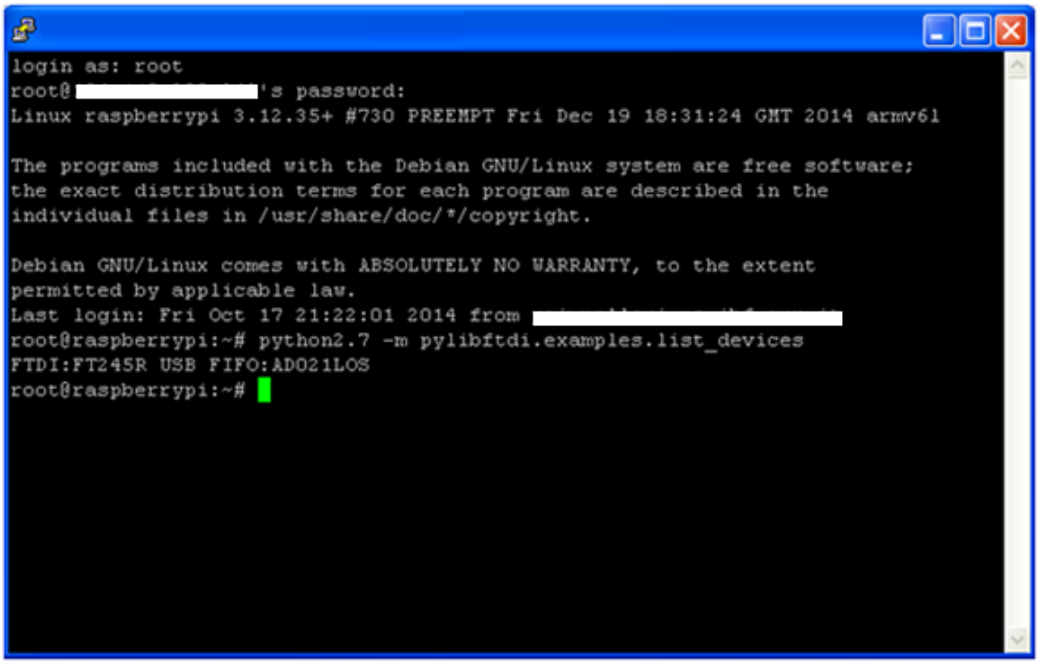

Figura 51: Verifica installazione libreria python-ftdi.

Appendice C

Procedura di installazione:

Dal sito http://www.tornadoweb.org/en/stable/ scaricare l'ultima versione di Tornado, nel nostro caso: tornado-4.1.tar.gz .

Spostarlo nella directory apposita, scompattarlo ed eseguire l'installazione:

```
tar xvzf tornado-4.1.tar.gz
cd tornado-4.1
python setup.py build
sudo python setup.py install
```

Per assicurarsi che tutto funzioni, creiamo un file denominato *web_server.py* che contiene il codice seguente, e lo eseguiamo dal terminale utilizzando il comando *python web_server.py*:

```
import tornado.ioloop
import tornado.web
class MainHandler(tornado.web.RequestHandler):
    def get(self):
        self.write("Hello, world")
application = tornado.web.Application([
    (r"/", MainHandler),
])
if __name__ == "__main__":
    application.listen(8080)
    tornado.ioloop.IOLoop.instance().start()
```

Adesso su un altro computer, collegato sulla stessa rete LAN, colleghiamo il browser a *http: //raspberrypi:8080* (o utilizzando l'indirizzo IP del Raspberry Pi). Se tutto è andato a buon fine, si dovrebbe visualizzare: *Hello, world*.

Appendice D1: codice sorgente *pysun.py*

```
# -*- coding: utf-8 -*-
import pylibftdi as ftdi
import struct
import sys
import time
import numpy as np
#suncard specific definitions
#FTDI:FT245R USB FIFO:AD021LOS
CMD_QUERY_FIRMWARE = 0x00
CMD_GET_NBYTES     = 0x01
CMD_CONVERT        = 0x02
CMD_SETNDATA       = 0x03
CMD_ACQUIRE_NDATA  = 0x04
CMD_SEND_NDATA     = 0x05
READ_TIMEOUT       = 3.0
HWRANGE            = 2.048
BITSTEP            = HWRANGE/65536.0
logging = False

class suncard():
    def __init__(self):
        try:
            self.device = ftdi.Device(device_id='AD021LOS')
            #print "Apro collegamento con la Scheda"
        except:
             #print "Errore di Collegamento con la Scheda"
            if logging:
                print sys.exc_info()[0]

    def close(self):
        return self.device.close()

    def __del__(self):
        self.close()

    def get_firmware(self):
        r = self.device.write(chr(CMD_QUERY_FIRMWARE))
        buf = struct.unpack('<BB',self.device.read(2))

        if buf[0] != CMD_QUERY_FIRMWARE:
            print('Suncard error')
            return False
        major   = int( float(buf[1]) / 16.0)
        minor = buf[1] - major * 16;
        print ('Firmware version {0}.{1}'.format(major,minor))
        return [major,minor]

    def _getVolt(self,msb,lsb=None):
        if lsb == None:
            (msb,lsb)=msb
```

```
        word = msb * 256 + lsb
        if (msb >= 0x80) :
            volt = (HWRANGE/2.0) - (BITSTEP * (word - 32768))
        else:
            volt = -BITSTEP * word
        return volt

    def get_value(self):
        r = self.device.write(chr(CMD_CONVERT))
        buf = struct.unpack('<BBB',self.device.read(3))
        if buf[0] != CMD_CONVERT:
            print('Suncard error')
            return False
        return self._getVolt(buf[1],buf[2])

    def flush(self):
        allbuf=''
        s = self.device.read(1)
        while (s!='') :
            allbuf+=s
            s = self.device.read(1)
        print "Flushed {0} bytes from the
       board".format(len(allbuf))
        return allbuf

    def get_values(self,nData):

        msb = (nData & 0xFF00) >> 8;
        lsb = nData & 0xFF;

        # 1-Tell the board how many data we are interested in
        # --> Indicare i dati di nostro interesse
        r = self.device.write(chr(CMD_SETNDATA ))
        r = self.device.write(chr(lsb ))
        r = self.device.write(chr(msb ))
        buf = struct.unpack('<B',self.device.read(1))

        if buf[0] != CMD_SETNDATA:
            print('Suncard error')
            return False

        # 2-Send the start acquisition command
        # --> Inviare il comando di acquisizione
        r = self.device.write(chr(CMD_ACQUIRE_NDATA ))
        buf = struct.unpack('<B',self.device.read(1))

        if buf[0] != CMD_ACQUIRE_NDATA:
            print('Suncard error')
            return False

        # 3-retrieve the data from the board
        # --> recuperare i dati dalla scheda
        r = self.device.write(chr(CMD_SEND_NDATA ))

        i = 0
        voltarray = []
        start = time.time()
        while(len(voltarray)<nData):
            buf = self.device.read(2)
```

```
            if buf != '':
                if len(buf)==1:
                    print 'Suncard error! Requested buffer was too
high'
                    break
                try:
                    buf = struct.unpack('<BB',buf)
                    voltarray.append(self._getVolt(buf[0],buf[1]))
                except:
                    print 'Risultato inatteso at
N={0}'.format(len(voltarray))
                    break
            elif len(voltarray)>0:
                ora = time.time()
                if (ora-start)>READ_TIMEOUT:
                    print "Timeout of the acquisition at {0}
data".format(len(voltarray))
                    break
        voltarray= np.array(voltarray, float)
        ##Effettuo la correzione del Volt
        ##esatto, con la seguente formula:
        voltarray = (voltarray - 0.0283 )/ 0.903
        #print np.median(voltarray)
        return voltarray
```

Appendice D2: codice sorgente *web_server.py*

```
import time
import os
import tornado.ioloop
import tornado.web
import numpy as np
#from scipy.fftpack import rfft, irfft, fftfreq, fft , fftshift
from scipy.optimize import curve_fit
from pysun import *
import json

def func(xdata2, nu_k, A, B, x2_nu_k, Q):
    y = A/xdata2
    y = y+ B
    y1=x2_nu_k / (Q*Q)
    y2=1- (xdata2/nu_k) * (xdata2/nu_k)
    y3=xdata2/(nu_k * Q)
    y=y+y1/(y2*y2+y3*y3)
    return y

class resetData(tornado.web.RequestHandler):

    def get(self):
        self.render("index.html")

    def post(self):
        global counter, in_nu_k, in_x2_nu_k
        reset = self.get_argument("reset", strip = True)
        if reset=='0':
            counter = 0.
            in_nu_k = -1.
            in_x2_nu_k = -1.

class initialiteParameters(tornado.web.RequestHandler):

    def get(self):
        self.render("index.html")

    def post(self):
        global counter, numPoints, A, B, Q, in_nu_k, in_x2_nu_k,
A_bis, B_bis, Q_bis
        reset = self.get_argument("reset", strip = True)
        if reset=='0':
            counter = 0.
        numPoints = int(self.get_argument("numPoints", strip =
True))
        A = float(self.get_argument("a", strip = True))
        B = float(self.get_argument("b", strip = True))
        Q = float(self.get_argument("q", strip = True))
        in_nu_k = float(self.get_argument("in_nu_k", strip =
True))
        in_x2_nu_k = (float(self.get_argument("in_x2_nu_k", strip
```

```
= True)))
class setParameters(tornado.web.RequestHandler):

    def get(self):
        self.render("index.html")

    def post(self):
        global T, theta, delta_zm_vm, fattore
        reset = self.get_argument("reset", strip = True)
        if reset=='0':
            counter = 0.
        T = float(self.get_argument("t", strip = True))
        # Prelevo theta e lo converto in radianti per il coseno
        theta = float(self.get_argument("theta", strip = True)) *
(np.pi/180)
        delta_zm_vm = float(self.get_argument("delta_zm_vm", strip
= True))
        fattore = 0.75 * ((delta_zm_vm*delta_zm_vm) *
((1/np.cos(theta))*(1/np.cos(theta))))
        print T, theta, delta_zm_vm, fattore

class MainHandler(tornado.web.RequestHandler):

    def get(self):
        self.render("index.html")

class SendData(tornado.web.RequestHandler):

    def get(self):
        global p1, numPoints, xs, yf, counter, A, B, Q, T,
in_nu_k, in_x2_nu_k
        if counter < 1:
            yf = np.zeros(numPoints, 'complex')
        print "Richiesta JSON"
        #Data Signal
        x = np.array(p1.get_values(numPoints), dtype=float)
        ys = x
        ys=np.roll(ys,-1)
        xs=np.arange(numPoints)
        data = [ list(a) for a in zip(xs[:numPoints-1],
ys[:numPoints-1])]

        #Scrivo i dati su file
        os.remove("/home/pi/Desktop/tornado/flot/file_log.txt")

        filelog =
open("/home/pi/Desktop/tornado/flot/file_log.txt", "a")
        bb = []
        for aa in xs:
            bb.append(aa)

        filelog.write("Data Signal - xs:\n"+str(bb)+"\n")

        cc = []
        for dd in ys:
            cc.append(dd)

        filelog.write("Data Signal - ys:\n"+str(cc)+"\n")
```

```
        #FFT
        media = (sum(x) / len(x))
        x = x - media
        yf_current = np.fft.fft(x)
        yf = yf + np.fft.fftshift(yf_current)
        counter = counter +1.
        yplot = yf/counter

        #Power Spectrum
        yplot = (np.abs(yplot)**2) /(numPoints **2)

        xf = np.fft.fftfreq(numPoints, 1.)
        xf = np.fft.fftshift(xf) * 161764.
        k = yplot.argmax()

        valmax_x = np.abs(xf[k])  #frequenza
        y_max = (yplot[k])

        xdata = xf
        ydata = yplot

        x2_nu_k = y_max * fattore   #Fluttuazione
        nu_k = valmax_x             #Frequenza

        #Test
        test = np.trapz(yf_current)
        #print "Test " + str(test)


        dataFFT = [ list(a) for a in zip(xdata[numPoints/2:],
ydata[numPoints/2:])]

        #filelog.close()

        # Curve FIT
        if (in_nu_k > 0. and in_nu_k != nu_k):
            x2_nu_k = in_x2_nu_k
            nu_k = in_nu_k

        elimino_zero = [numPoints/2]
        xdata2 = np.delete(xdata, elimino_zero)
        ydata2 = np.delete(ydata, elimino_zero)

        if((theta != 0) or (delta_zm_vm != 1) ):
            ydata2 = 0.75 * ydata2 * ((delta_zm_vm*delta_zm_vm) *
((1/np.cos(theta))*(1/np.cos(theta))))
            dataFFT = [ list(a) for a in zip(xdata2[numPoints/2:],
ydata2[numPoints/2:])]

        #ricavo il delta_nu, passo della Freq
        #serve per il calcolo del metodo k
        delta_nu = np.abs(xdata2[2] - xdata2[1])
        #Memorizzo i valori del range selezionato
        xaxis_from = float(self.get_argument("xaxis_from", strip =
True))
        xaxis_to = float(self.get_argument("xaxis_to", strip =
True))
        range_fit_x = []
        range_fit_y = []
```

```
        ErrorFIT = ""
        if((np.isnan(xaxis_from) != True) and (np.isnan(xaxis_to)
!= True)):
            # Range selezionato - Valori presenti
            for i in range(len(xdata2)):
                #Memorizzo in due nuovi vettori i valori presenti
all'interno
                # del range selezionato, per x e y, min < xdata2 <
max
                if((xdata2[i] > xaxis_from) and (xdata2[i] <
xaxis_to)):
                    range_fit_x.append(xdata2[i])
                    range_fit_y.append(ydata2[i])
        try:
            if((len(range_fit_x) == 0) and (len(range_fit_y) ==
0)):
                #Se non ci sono nuovi valori nel range: min <
xdata2 < max
                #fai il fit su xdata2, ydata2
                ypopt = func(xdata2, nu_k, A, B, x2_nu_k, Q)
                popt, pcov = curve_fit(func, xdata2,
ydata2,p0=[nu_k, A, B, x2_nu_k, Q])
                ypopt = func(xdata2, *popt)
                dataFIT = [ list(a) for a in
zip(xdata2[numPoints/2:], ypopt[numPoints/2:])]
                nu_k_fit = popt[0]
                A = popt[1]
                B = popt[2]
                x2_nu_k_fit = popt[3]
                Q = np.abs(popt[4])
                method_k = ((2*kb*T*Q*delta_nu) / (np.pi *
nu_k_fit * x2_nu_k_fit))
                #method_k = (method_k * np.power(10,+9))
                method_k = float(method_k)* (10**+18)

            else:
                #Se i nuovi valori sono presenti, nel range: min <
xdata2 < max,
                #fai il fit su  range_fit_x, range_fit_y
                range_fit_x = np.array(range_fit_x, dtype=float)
                range_fit_y = np.array(range_fit_y, dtype=float)
                ypopt = func(range_fit_x, nu_k, A, B, x2_nu_k, Q)
                popt, pcov = curve_fit(func, range_fit_x,
range_fit_y,p0=[nu_k, A, B, x2_nu_k, Q])
                ypopt = func(range_fit_x, *popt)
                dataFIT = [ list(a) for a in
zip(range_fit_x[numPoints/2:], ypopt[numPoints/2:])]
                nu_k_fit = popt[0]
                A = popt[1]
                B = popt[2]
                x2_nu_k_fit = popt[3]
                Q = np.abs(popt[4])
                method_k = ((2*kb*T*Q*delta_nu) / (np.pi *
nu_k_fit * x2_nu_k_fit))
                #method_k = (method_k * np.power(10,+9))
                method_k = (method_k * (10**18))
        except:
            dataA = []
            dataB = []
```

```
            dataQ = []
            dataFIT = []
            dataCurveFIT = []
            nu_k_fit = 0.
            dataNu_k_fit = []
            x2_nu_k_fit = 0.
            dataX2_nu_k_fit =[]
            method_k = 0.
            dataMethod_k = []
            ErrorFIT = "Error FIT. Optimal parameters not found."
            print ErrorFIT

        dataSignal = {
                        "label" : "Signal",
                        "data" : data,
                        "color": "#0000FF"
                    }

        dataSignalFFT = {
                            "label" : "FFT",
                            "lines" : {"show": "true"},
                            "points" : {"show": "true"},
                            "data" : dataFFT,
                            "color": "#D2691E"
                        }
        dataCurveFIT = {
                            "label" : "FIT Curve",
                            "data" : dataFIT,
                            "color": "#DC143C"
                        }

        dataValMax_x = {
                            "label" : "Valore Max X",
                            "data"  : [("%10.0f" % nu_k) ]
                        }
        dataValMax_y = {
                            "label" : "Valore Max Y",
                            "data"  : [("%10.3f" % x2_nu_k) ]
                        }
        dataA = {
                    "label" : "A",
                    "data"  : [("%10.4f" % A)]
                }
        dataB = {
                    "label" : "B",
                    "data"  : [("%10.4f" % B)]
                }
        dataQ = {
                    "label" : "Q",
                    "data"  : [("%10.4G" % Q )]
                }
        dataNu_k_fit = {
                            "label" : "Nu_K FIT",
                            "data"  : [("%10.4f" % nu_k_fit)]
                        }
        dataX2_nu_k_fit = {
                                "label" : "X2_nu_k FIT",
                                "data"  : [("%10.4e" %
x2_nu_k_fit)]
```

```
                        }
        dataMethod_k = {
                            "label" : "Method K",
                            "data"  : [("%10.4e" % method_k)]
                        }

        dataErrorFIT = {
                            "label" : "Error FIT",
                            "data" : [ErrorFIT],
                            "color": "#DC143C"
                        }

        dataOBJ = [ dataSignal, dataSignalFFT, dataCurveFIT,
dataValMax_x, dataValMax_y, dataA, dataB, dataQ, dataNu_k_fit,
dataX2_nu_k_fit, dataMethod_k, dataErrorFIT ]

        data_json = json.dumps(dataOBJ)
        self.write(data_json)

root = os.path.dirname('__file__')
port = 8888

application = tornado.web.Application([
    (r"/", MainHandler),
    (r"/salvodata.json", SendData),
    (r'/ws', resetData),
    (r'/initialite', initialiteParameters),
    (r'/setParameters', setParameters),
    (r"/(.*)", tornado.web.StaticFileHandler, {"path": root,
"default_filename": "index.html"})
])

if __name__ == '__main__':
    global counter, numPoints, A, B, Q, T, kb, in_nu_k,
in_x2_nu_k, theta, delta_zm_vm, fattore
    p1 = suncard()
    counter = 0.
    numPoints=500
    A= 0.
    B= 0.
    Q = 70.
    T = 297. #Kelvin
    kb = 1.38e-23   #Costante di Boltzmann J/K
    in_nu_k = -1.
    in_x2_nu_k = -1.
    theta = 0
    delta_zm_vm = 1.
    fattore = 1.

    application.listen(port)
    tornado.ioloop.IOLoop.instance().start()
```

Appendice D3: codice sorgente *index.html*

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
	<meta http-equiv="Content-Type" content="text/html;
charset=utf-8">
	<title>Impallari</title>
	<link href="examples.css" rel="stylesheet" type="text/css">
	<!--[if lte IE 8]><script language="javascript"
type="text/javascript" src="excanvas.min.js"></script><![endif]-->
	  <script type="text/javascript"
src="tooltips/bubble.js"></script>
	<script language="javascript" type="text/javascript"
src="jquery.js"></script>
	<script language="javascript" type="text/javascript"
src="jquery.flot.js"></script>
	<script language="javascript" type="text/javascript"
src="jquery.flot.selection.js"></script>
	<script type="text/javascript">
	$(function() {
		// setup plot
		var myVar;
		var plot;
		var overview;
		var data = [];
		var dataFFT = [];
		var dataFIT = [];
		var valmax_x = [];
		var y_max = [];
		var A, B, Q, nu_k_fit, x2_nu_k_fit, method_k, ErrorFIT
		    = [];
		var xaxis_from, xaxis_to, yaxis_from, yaxis_to;

		    var options = {
			    legend: {
				    show: true
			    },
			    series: {
				    lines: {
					    show: true
				    },
				    points: {
					    show: false
				    }
			    },
			    yaxis: {
				    ticks: 5,

			    },
			    selection: {
```

```
                        mode: "xy"
                    }
                };
            $("button.start").click(function(){
             myVar = setInterval(function () {fetchData()}, 1000);
            });
            $("button.stop").click(function(){
                        clearInterval(myVar);
                         myVar = null;
                          $.xhrPool.abortAll();
                   });

                  // Rimuovo le richieste in "Pending"
                  $.xhrPool = [];
                  $.xhrPool.abortAll = function() {
                        $(this).each(function(idx, jqXHR) {
                              jqXHR.abort();  //aborts connection
                              $.xhrPool.splice(idx, 1);
                              //removes from list
                        });
                  };
                  $.ajaxSetup({
                        beforeSend: function(jqXHR) {
                              $.xhrPool.push(jqXHR);
                        },
                        complete: function(jqXHR) {
                              var index =
                              $.xhrPool.indexOf(jqXHR);
                              if (index > -1) {
                                    $.xhrPool.splice(index, 1);
                              }
                        }
                  });

                // Zoom
                $("#placeholderFFT").bind("plotselected", function
                      (event, ranges) {
                    plot = $.plot("#placeholderFFT", [ dataFFT,
                           dataFIT ],
                          $.extend(true, {}, options, {
                                  xaxis: { min: ranges.xaxis.from,
                                           max: ranges.xaxis.to },
                                  yaxis: { min: ranges.yaxis.from,
                                           max: ranges.yaxis.to }
                                  })
                           );
                        xaxis_from = ranges.xaxis.from;
                        xaxis_to   = ranges.xaxis.to;
                        yaxis_from = ranges.yaxis.from;
                        yaxis_to   = ranges.yaxis.to;


$("#selection_x_from").text(ranges.xaxis.from.toFixed(2));

$("#selection_x_to").text(ranges.xaxis.to.toFixed(2));

                        overview.setSelection(ranges, true);
                        });
```

```
                    $("#overview").bind("plotselected",
function (event,ranges) {
                              plot.setSelection(ranges);
                            });
                    function fetchData() {
                            function onDataReceived(series) {
                                  data        =  series[0] ;
                                  dataFFT     =  series[1] ;
                                  dataFIT     =  series[2] ;
                                  valmax_x    =  series[3] ;
                                   y_max       =  series[4] ;
                                   A           =  series[5] ;
                                  B           =  series[6] ;
                                  Q           =  series[7] ;
                                  nu_k_fit    =  series[8] ;
                                  x2_nu_k_fit =  series[9] ;
                                  method_k    =  series[10] ;
                                   ErrorFIT    =  series[11] ;

                    $.plot("#placeholder",[ data ], options);

                      plot = $.plot("#placeholderFFT",[
                          dataFFT, dataFIT ], {
                               legend: {
                                         show: true
                                         },
                                         xaxis: {
                                                min: xaxis_from,
                                                max: xaxis_to
                                                //min: 2000
                                                //max: 40000
                                                 },
                                          yaxis: {
                                                    ticks: 10
                                               //min: yaxis_from,
                                               //max: yaxis_to
                                                      },
                                           selection: {
                                                   mode: "xy"
                                                      }
                                              });

                    overview = $.plot("#overview", [
                                        dataFFT, dataFIT ], {
                                         legend: { show: false
                                                },grid: {
                                                    color: "#999"
                                                 },xaxis: {
                                                    //min: 2000,
                                                    //max: 40000
                                                    size: 5
                                                  },
                                                  yaxis: {
                                                    size: 5,
                                                    lineHeight: 8

                                                    },selection:
                                                         {
                                                    mode: "xy"
```

```
                                                        }
                                                    });
document.getElementById('in_nu_k').value =
String(valmax_x.data[0]) ;
document.getElementById('in_x2_nu_k').value =
String(y_max.data[0]) ;
document.getElementById('A').innerHTML =  String(A.data[0]) ;
document.getElementById('B').innerHTML =  String(B.data[0]) ;
document.getElementById('Q').innerHTML =  String(Q.data[0]) ;
document.getElementById('nu_k_fit').innerHTML =
String(nu_k_fit.data[0]) ;
document.getElementById('x2_nu_k_fit').innerHTML
=String(x2_nu_k_fit.data[0]) ;
document.getElementById('K').innerHTML =  String(method_k.data[0])
;
document.getElementById('ErrorFIT').innerHTML =
String(ErrorFIT.data[0]) ;

                                    }

                                    $.ajax({
                                            url: "salvodata.json",
                                            type: "GET",
                                            data: {
                                               "xaxis_from" :
                                             parseFloat(xaxis_from),
                                               "xaxis_to" :
                                             parseFloat(xaxis_to)
                                            },
                                            dataType: "json",
                                            success: onDataReceived
                                    });

                                }

                                $("#autoscale").click(function() {
                                        xaxis_from = null;
                                        xaxis_to = null;
                                        yaxis_from = null;
                                        yaxis_to = null;

        $("#selection_x_from").text("");
                $("#selection_x_to").text("");

                plot = $.plot("#placeholderFFT",[ dataFFT, dataFIT ],
                                {
                                  legend: {
                                      show: true
                                            },
                                  xaxis: {
                                            min: xaxis_from,
                                            max: xaxis_to
                                            //min: 2000,
                                            //max: 40000
                                            },
                                  yaxis: {
                                           ticks: 10,
                                            min: yaxis_from,
                                            max: yaxis_to
```

```
                                },
                            selection: {
                                        mode: "xy"
                                         }
                                    });
                        });

                        $("#initialize").click(function() {
                        $.ajax({
                          url: "initialite",
                          type: "POST",
                          data: {
                             "reset" : "0",
                             "numPoints" :
document.getElementById('numPoints').value,
                            "a" : document.getElementById('a').value,
                            "b" : document.getElementById('b').value,
                            "q" : document.getElementById('q').value,
                            "in_nu_k" :
document.getElementById('in_nu_k').value,
                            "in_x2_nu_k" :
document.getElementById('in_x2_nu_k').value
                                              },
                           success: function(msg){
                                     console.log(msg);
                                           }
                                    });
                           });

                            $("#setParam").click(function() {
                            $.ajax({
                               url: "setParameters",
                               type: "POST",
                               data: {
                                     "reset" : "0",
                                     "t" :
document.getElementById('t').value,
                                     "theta" :
document.getElementById('theta').value,
                                     "delta_zm_vm" :

document.getElementById('delta_zm_vm').value,
                                              },
                                  success: function(msg){
                                             console.log(msg);
                                             }
                                     });
                                });

                                $("#clearSelection").click(function() {
                                       $.ajax({
                                              url: "ws",
                                              type: "POST",
                                              data: {
                                                      "reset" : "0"
                                              },
                                              success: function(msg){
                                                      console.log(msg);
                                              }
```

```
                                });
                            });

                fetchData();
                });
            function fitFormula(){
                var w = 400;
                var h = 250;
                var l = Math.floor((screen.width-w)/2);
                var t = Math.floor((screen.height-h)/2);
                window.open("fitFormula.html","","width=" + w +
",height=" + h + ",top=" + t + ",left=" + l);

            }
           function method_k(){
                    var w = 400;
                    var h = 250;
                    var l = Math.floor((screen.width-w)/2);
                    var t = Math.floor((screen.height-h)/2);
                    window.open("method_k.html","","width=" + w +
",height=" + h + ",top=" + t + ",left=" + l);

            }
          </script>

</head>
<body>
<div id="header">
      <h2><center>Cantilever Calibration</center></h2>
</div>
<div id="content">
      <div class="demo-container">
      <div id="placeholder" class="demo-placeholder"></div>
      </div>
      <div id="col1">
      <p><center><b>Initialize Parameters</b></center></p>
      <p><input id="numPoints" type="text" value="500"
style="text-align: right; width:5em"> <a href="#"
onmousemove="showToolTip(event,'Enter the number of
points.');return false" onmouseout="hideToolTip()">of data
points</a></p>
      <p><input id="a" type="text" value="0"  style="text-align:
right; width:5em"> <a href="#"
onmousemove="showToolTip(event,' 1/&#957; amplitude.');return
false" onmouseout="hideToolTip()"> A [nm&#178;]</a></p>
       <p><input id="b" type="text" value="0" style="text-align:
right; width:5em"> <a href="#"
onmousemove="showToolTip(event,'Spectrum Offset.');return false"
onmouseout="hideToolTip()"> B [nm&#178;]</a></p>
        <p><input id="q" type="text" value="70" style="text-align:
right; width:5em"> <a href="#"
onmousemove="showToolTip(event,'Quality factor.');return false"
onmouseout="hideToolTip()"> Q</a></p>
        <p><input id="in_nu_k" type="text" value="" style="text-
align: right; width:5em"> <a href="#"
onmousemove="showToolTip(event,'Kinetic resonant
frequency.');return false" onmouseout="hideToolTip()"><img
src=img/nu_k.bmp> [Hz]</a></p>
```

```
        <p><input id="in_x2_nu_k" type="text" value=""
style="text-align: right; width:5em"> <a href="#"
onmousemove="showToolTip(event,'The mean-square amplitude at
kinetic resonance.');return false" onmouseout="hideToolTip()"><img
src=img/x2_nu_k.bmp> [nm&#178;]</a></p>
          <p><center><button id="initialize"> Initialize
Parameters </button></center></p>
               </div>
           <div class="demo-containerFFT">
               <div id="placeholderFFT" class="demo-
placeholderFFT"></div>
           </div>
           <div class="demo-overview">
               <div id="overview" class="overviewSmall"></div>
               </div>
           <div id="col5">
               <p><center><b>Setup Parameters</b></center></p>
               <p><input id="t" type="text" value="297"
style="text-align: right; width:5em"> <a href="#"
onmousemove="showToolTip(event,'Insert temperature of the
cantilever for the calculation of the method k.');return false"
onmouseout="hideToolTip()"> T [Kelvin]</a></p>
               <p><input id="theta" type="text" value="0"
style="text-align: right; width:5em"><a href="#"
onmousemove="showToolTip(event,'The cantilever angle with respect
to the horizontal.');return false" onmouseout="hideToolTip()">
angle &#977; [&#176;]</a></p>
               <p><input id="delta_zm_vm" type="text" value="1"
style="text-align: right; width:5em"><a href="#"
onmousemove="showToolTip(event,'Ratio between the manufacturer
stated z movement &#916;Zm and photodiode voltage
&#916;Vm.');return false" onmouseout="hideToolTip()"><img
src=img/delta_zm_vm.bmp> [nm/V]</a></p>
               <p><center><button id="setParam"> Setup Parameters
</button></center></p>
               </div>
               <div id=colcentral>
               <p><a href="javascript:fitFormula()"><center><font
color="#DC143C">Fitting Parameters</font></center></a></p>
               <p>A: </span></p>
               <p>B: </p>
               <p>Q: </p>
               <p><img src="img/nu_k_fit.bmp"></p>
               <p><img src="img/x2_nu_k_fit.bmp"></p>
               <p><a href="javascript:method_k()"><font
color="#000080">K:</font></a> </p>
<div id=col4>
<p style="padding-top:1px;"><span id="A"></span></p>
<p style="padding-top:1px;"><span id="B"></span></p>
<p><span id="Q"></span></p>
<p><span id="nu_k_fit"></span></p>
<p><span id="x2_nu_k_fit"></span></p>
<p style="padding-top:2px;"><span id="K"></span></p>
</div>
           <div id=unita>
           <p><img src="img/unit2.bmp"></p>
           </div>
               <p></p>
               </div>
```

```
                <div id="col2"><center>
<p><button class="start" style="font-size:18px; padding: 1px 7px;
width:130px; height:50px;" >Start</button></p>
<p><button class="stop" style="font-size:18px; padding: 1px 7px;
width:130px; height:50px;">Stop</button></p>
<p><button id="clearSelection" style="font-size:18px; padding: 1px
7px; width:130px; height:50px;"> Reset Data </button></p>
</center></div>
<div id="col3">
<p><center><b>Selected Frequency Range:</b></center></p>
<p>From: <span id="selection_x_from"></span></p>
<div id=col3_to>
<p>To: <span id="selection_x_to"></span></p>
</div>
 <p><center><button id="autoscale"> Auto Scale
</button></center></p>
</div>
<div id="errorFit">
<p><b><span id="ErrorFIT" style="color: #ff0000"></b></p>
</div>
</div>
<p></p>
<p></p>
      <div id="bubble_tooltip">
      <div class="bubble_top">
      <span></span>
      </div>
      <div class="bubble_middle">
      <span id="bubble_tooltip_content"></span>
      </div>
      <div class="bubble_bottom">
      </div>
</div>
</body>
</html>
```