

Department of Information Engineering and Computer Science

Bachelor’s Degree in

COMPUTER SCIENCE

final dissertation

Development of a gui for the

packet loss concealment testbench tool

|  |  |
| --- | --- |
| Supervisor  Prof. Luca Turchet  Ph.D Luca Vignati | Student  Stefano Dallona |

Academic year 22/23

**Ringraziamenti**

A mia moglie e alle mie figlie

che mi hanno sostenuto costantemente in questo percorso

**INDEX**

[**1** **Packet Loss Concealing (PLC)** 7](#_Toc147240521)

[**1.1** **Digital audio fundamental concepts** 7](#_Toc147240522)

[**1.2** **Domain (Real-time streaming of audio data excluding speech)** 7](#_Toc147240523)

[**1.3** **Simulazione della perdita di pacchetti** 7](#_Toc147240524)

[1.3.1 Modello basato sulla distribuzione di probabilità binomiale 8](#_Toc147240525)

[1.3.2 Modello di Gilbert-Elliot 8](#_Toc147240526)

[**1.4** **Packet loss concealment** 9](#_Toc147240527)

[**1.5** **Metriche per la valutazione dei file ricostruiti** 9](#_Toc147240528)

[**2** **PLC Testbench UI** 10](#_Toc147240529)

[**2.1** **Funzioni principali dell’interfaccia grafica** 10](#_Toc147240530)

[**2.1.1** **Configurazione dei dati di input** 10](#_Toc147240531)

[**2.1.2** **Gestione e monitoraggio delle elaborazioni** 12](#_Toc147240532)

[**2.1.3** **Analisi dei risultati** 13](#_Toc147240533)

[**2.1.4** **Ricerca e visualizzazione delle elaborazioni** 16](#_Toc147240534)

[**2.2** **User Interaction and Experience** 16](#_Toc147240535)

[**2.3** **Scelte tecniche** 16](#_Toc147240536)

[**2.3.1** **Architettura dell’applicazione** 16](#_Toc147240537)

[**2.3.2** **Linguaggi di programmazione utilizzati** 16](#_Toc147240538)

[**2.3.3** **Frameworks** 17](#_Toc147240539)

[**2.3.4** **Packaging e distribuzione** 18](#_Toc147240540)

[**3** **Risultati** 19](#_Toc147240541)

[**3.1** **Supporto nella validazione e nel debug dell’applicazione PLC-TestBench** 19](#_Toc147240542)

[**3.2** **Ampliamento dei potenziali utilizzatori dell’applicazione PLC-TestBench** 19](#_Toc147240543)

[**3.3** **Miglioramento della user experience** 19](#_Toc147240544)

[**3.4** **Sviluppi futuri** 19](#_Toc147240545)

[**4** **Conclusione** 20](#_Toc147240546)

**Abstract**

*Lo scopo di questa tesi era quello di implementare un'interfaccia utente grafica per il Testbench Tool PLC (Packet Loss Concealment) sviluppato dal dottorando Luca Vignati, rendendo così lo strumento più facile da usare e allo stesso tempo potenziando l'interpretazione e l'analisi dei dati di output dello strumento.*

*Prima dell'implementazione di questa interfaccia utente, l'unica interazione disponibile avveniva direttamente in un Jupyter Notebook, dove gli input dovevano essere immessi direttamente modificando il codice. Ciò richiedeva che l'utente disponesse di un ambiente di sviluppo completamente funzionale, con tutta la complessità che ciò implica, e richiedeva anche che l'utente avesse familiarità con la programmazione Python. Inoltre, non vi era alcun aiuto nell'interpretazione dei risultati, quindi l'unica opzione era consultare i file audio e le immagini grezze di output. Questo tipo di interazione non è ottimale, soprattutto in termini di consultazione dei risultati, in quanto si è costretti a considerare un elemento alla volta, precludendosi la possibilità di aumentare la qualità dell'analisi considerando più aspetti contemporaneamente.*

*Gli algoritmi PLC sono generalmente necessari in contesti di trasmissione dati in tempo reale e il loro ruolo è quello di mascherare ("nascondere") qualsiasi perdita che potrebbe verificarsi in uno o più pacchetti in modo trasparente per l'applicazione che utilizza il flusso di dati.*

*Lo strumento PLC Testbench mira a fornire un framework modulare per supportare un confronto qualitativo e quantitativo delle prestazioni di diversi algoritmi PLC, applicandone due o più a una serie di file audio forniti come input allo strumento. La perdita di pacchetti viene simulata applicando modelli matematici stocastici con parametrizzazione configurabile. Il pacchetto software include un'implementazione degli algoritmi di simulazione della perdita di pacchetti utilizzati più frequentemente come il Binomial PLS, basato sulla distribuzione binomiale della probabilità di perdita e il Gilbert-Elliott PLS, basato su una catena di Markov a due stati.*

*L’analisi qualitativa è supportata da numerose rappresentazioni visive delle caratteristiche dei segnali audio (maschere dei pacchetti persi, forme d’onda, spettrogrammi) e dalla possibilità di riprodurre qualsiasi sottoinsieme di ciascuna traccia audio prodotta per valutarne la qualità percepita.*

*L'analisi quantitativa del risultato dell'algoritmo PLC è supportata da molteplici metriche prodotte dallo strumento per ciascuno dei file audio ricostruiti, come l’errore quadratico medio, l’energia spettrale, PEAQ (valutazione percettiva della qualità audio).*

*Come framework, lo strumento può essere facilmente esteso fornendo implementazioni aggiuntive dei diversi tipi di oggetti (algoritmo PLS, algoritmo PLC, metrica di analisi dell'output).*

*La GUI può gestire queste estensioni esponendo in modo trasparente le nuove funzionalità senza la necessità di alcuna modifica del codice. Ciò si ottiene sfruttando le capacità di introspezione del linguaggio di programmazione sottostante e scansionando la libreria del banco di prova del PLC per scoprire le funzionalità personalizzate.*

*Ogni elaborazione dello strumento produce un dataset di risultati, convenzionalmente chiamato “Run”, che contiene un'istantanea di tutti i parametri di input e di tutto l'output prodotto (maschere di pacchetti persi, audio ricostruito, file di metriche).*

*L'elaborazione può essere rappresentata come una foresta di alberi il cui nodo radice è associato ad uno dei file audio forniti in input. Ogni file di input dà origine a più simulazioni di perdita, una per ogni istanza di un algoritmo di perdita di pacchetti configurato. Le simulazioni di perdita possono essere mappate sui nodi alla profondità uno negli alberi e le tracce audio corrispondenti vengono a loro volta ricostruite applicando ciascuno degli algoritmi PLC configurati a livello Run. I file audio ricostruiti sono rappresentati da nodi a profondità due negli alberi. Infine, ciascun algoritmo ricostruito viene valutato applicando ciascuna metrica configurata nel Run alla corrispondente traccia audio ricostruita.*

*È possibile impostare più istanze di ciascun oggetto di configurazione su un'esecuzione, purché le impostazioni corrispondenti siano univoche all'interno dell'esecuzione.*

*Durante la fase di progettazione sono stati valutati diversi tipi di GUI e i loro pro e contro sono stati attentamente soppesati. Infine, si è deciso di optare per lo sviluppo di un'applicazione web a causa dei numerosi vantaggi offerti dalla tecnologia web, come un'ampia gamma di modalità di implementazione (che vanno da un ambiente locale autonomo a grandi infrastrutture distribuite), la portabilità dell'applicazione su diverse piattaforme, la facilità di installazione. Tutto ciò al prezzo di uno sforzo leggermente maggiore nell’implementazione.*

*Per rendere la distribuzione del software il più semplice possibile, nonostante l'elevato numero di dipendenze in termini di librerie richieste, si è deciso di sfruttare la tecnologia dei containers.*

*Il design della GUI è stato realizzato con particolare attenzione alla modularità per facilitare la manutenzione e i futuri miglioramenti. La modularità è stata ottenuta costruendo widget grafici che possono essere riutilizzati in tutta l'applicazione.*

*Lo sviluppo dell'applicazione ha presentato diverse sfide, soprattutto a causa della grande quantità di dati intrinsecamente coinvolti nell'elaborazione audio. Anche tracce audio molto brevi sono composte da milioni di campioni che dovevano essere rappresentati visivamente come forme d'onda e/o riprodotti in streaming come suono in tempo reale. Per raggiungere le prestazioni desiderate è stato necessario adottare tecniche di sottocampionamento e ottimizzazioni multiple.*

*Un altro aspetto impegnativo è stato l’ampio spettro di tecnologie e linguaggi di programmazione da apprendere in un periodo di tempo relativamente breve.*

1. **Packet Loss Concealing (PLC)**

Il dominio in cui sono state applicate le tecniche PLC durante questo lavoro è quello dei segnali audio non vocali, poiché la ricostruzione del parlato richiede tecniche peculiari che erano al di fuori dell'ambito di ricerca da cui ha avuto origine questa tesi.

Al momento l’unico formato audio supportato e il WAV.

* 1. **Digital audio fundamental concepts**

La trasmissione del suono è un fenomeno fisico costituito da un'onda di pressione che si propaga attraverso un mezzo gassoso, fluido o solido. La frequenza, l'ampiezza e la fase di quest'onda possono cambiare nel tempo, originando uno spettro di frequenze. Il suono può essere percepito dall'orecchio umano solo se una qualsiasi delle frequenze dello spettro rientra in un determinato intervallo, chiamato campo uditivo umano.

L'audio digitale è una rappresentazione del suono in forma digitale. Tipicamente l'ampiezza dell'onda di pressione viene campionata a intervalli regolari e il suo valore viene mappato su un intervallo discreto di valori, dando così origine ad un flusso di numeri.

Secondo il teorema del campionamento di Nyquist-Shannon, per campionare tutte le frequenze comprese nel campo uditivo umano un segnale audio deve essere campionato ad una frequenza di 44,1 kHz .

.

A diagram of a wave

Description automatically generated with medium confidence A graph with red and blue dots

Description automatically generated

* 1. **Domain (Real-time streaming of audio data excluding speech)**

The packet loss can be simulated by randomly dropping some packets from the original audio stream based on a given probability distribution. The random variable represents a packet transmission whose possible outcomes are successful transmission or packet loss.

* 1. **Simulazione della perdita di pacchetti**

The packet loss can be simulated by randomly dropping some packets from the original audio stream based on a given probability distribution. The random variable represents a packet transmission whose possible outcomes are successful transmission or packet loss.

La perdita di pacchetti può essere simulata eliminando casualmente alcuni pacchetti dal flusso audio originale in base a una determinata distribuzione di probabilità. La variabile casuale associata alla distribuzione di probabilità rappresenta la trasmissione di un pacchetto, i cui possibili esiti sono la trasmissione riuscita o la perdita del pacchetto.

### Modello basato sulla distribuzione di probabilità binomiale

In questo modello la probabilità che la variabile casuale assuma il valore corrispondente a un pacchetto perso segue una distribuzione di probabilità binomiale, che rappresenta la somma di una serie di più prove Bernoulliane indipendenti e identicamente distribuite. Una prova di Bernoulli è un esperimento casuale con esattamente due possibili esiti (successo/fallimento), le cui probabilità sono p [0, 1] e q = 1 – p [0, 1].

La probabilità risultante di ottenere N eventi consecutivi di successo/fallimento è descritta dalla seguente formula basata sui coefficienti binomiali.

A math equations on a white background

Description automatically generated

### Modello di Gilbert-Elliot

Rispetto ai modelli basati su un’unica distribuzione di probabilità, il modello di Gilbert-Elliot presenta il grande vantaggio di tenere conto della correlazione tra gli eventi di perdita. Mentre nel modello di perdita con distribuzione binomiale ogni evento di perdita non è correlato al precedente e quindi la sua probabilità è indipendente, nel modello di Gilbert-Elliot la probabilità di ogni evento di perdita è condizionata da quello precedente.

Tale comportamento è in linea con le ipotesi delle catene di Markov ed infatti il ​​modello di Gilbert-Elliot può essere considerato una catena di Markov a due stati il ​​cui grafico è quello rappresentato sotto.

A diagram of a circular object

Description automatically generated

Questo modello è molto più accurato rispetto ai modelli a distribuzione singola, perché nel mondo reale la perdita di pacchetti è solitamente causata da guasti o congestione nella rete di trasmissione e quindi tende a verificarsi a raffiche.

Pertanto, quando si verifica una prima perdita, è più probabile che i pacchetti successivi vengano persi rispetto a uno scenario di trasmissione riuscita, perché il sistema è solitamente passato da uno stato funzionante a uno non funzionante. Allo stesso modo, quando l'ultimo pacchetto è stato trasmesso con successo ciò è indice che il sistema funziona correttamente e quindi la probabilità che il pacchetto successivo non vada perso è maggiore rispetto a quando il sistema è in uno stato non funzionante.

* 1. **Packet loss concealment**

PLC algorithms are typically needed in real-time data transmission contexts, and their job is to mask (“conceal”) any loss that may occur to one or more packets in a manner that is transparent to the application consuming the data stream. Any PLC algorithm needs to satisfy the following requirements:

* Its execution time needs to be smaller than the period between the consumption of packets.
* Its output doesn’t need to match exactly the content of the corresponding packet. However, when put back in place of the lost packet, it needs to be perceptually indistinguishable from the original audio stream.

There are many ways to perform PLC on audio signals. From simpler waveform-based solutions such as waveform repetition and PSOLA to more advanced autoregressive models. Recently, Deep Learning based PLC has begun to arise in the literature with some noticeable examples both in speech and music.

Regardless of the type, they all use the audio data received immediately before the lost packet as the input to the algorithm, which will output the number of samples contained in the lost packet.

In this paper, we present a software tool to study packet loss in audio streams and study, compare, and evaluate PLC algorithms for audio.

* 1. **Metriche per la valutazione dei file ricostruiti**

The metrics implemented by the last version PLC Testbench tool at the time of writing this thesis are:

**Mean Squared Error (MSE):** is a measure of the error in the measurement of a the same phenomenon across paired observation. The observation pair can be represented by an estimatated versus an

**Mean Absolute Error (MAE):**

A mathematical equation with black text

Description automatically generated

**Spectral Energy: ???**

**Perceptual Evaluation of Audio Quality:** is a standardized algorithm for objectively measuring perceived [audio quality](https://en.wikipedia.org/wiki/Audio_quality). Among all the metrics implemented out-of-the-box in the PLC Testbench tool this is certainly the most specialized and most effective one for the domain of audio signals processing because it is based on a [psychoacoustic model](https://en.wikipedia.org/wiki/Psychoacoustics) of the human ear and brain. The model takes into consideration human auditory field and its limits and compares a reference signal with a test signal, ignoring differences that are not considered as perceptible according to the model’s rules. PEAQ transforms the inputs by applying filter banks to the Discrete Fourier Transform (DFT) of the signals, producing an intermediate output in the form of a set of variables, each one capturing a different psychoacoustic dimension.

Finally the output variables are used to feed a neural network that simulates the cognitive processes happening inside the human brain which translates the input variables into an overall quality score named Objective Difference Grade (ODG).

The algorithm comes in two different versions: the basic one, less computationally expensive, and the advanced one, more accurate but computationally heavier and therefore not always applicable to real-time signals.

A diagram of a process

Description automatically generated

1. **PLC Testbench UI**
   1. **Funzioni principali dell’interfaccia grafica**

Lo scopo dell'interfaccia utente del PLC Testbench è quello di rendere lo strumento più facile da usare, ridurre il tempo necessario per consultare i risultati e aumentare la quantità e la qualità delle informazioni ottenute dall'analisi dei risultati.

Prima dell'implementazione di questa interfaccia utente, l'unica interazione disponibile avveniva direttamente in un Jupyter Notebook, dove gli input dovevano essere forniti direttamente modificando il codice. Inoltre, non vi era alcun aiuto nell'interpretazione dei risultati, quindi l'unico modo era consultare i file audio e le immagini grezze di output. Questo tipo di interazione non è ottimale, soprattutto in termini di consultazione dei risultati, in quanto si è costretti a considerare un elemento alla volta, precludendosi la possibilità di aumentare la qualità dell'analisi considerando più aspetti contemporaneamente.

L’interfaccia utente può essere concettualmente divisa in quattro parti:

1. Configurazione dei dati di ingresso

2. Monitoraggio dei progressi

3. Analisi dei risultati

* + 1. **Configurazione dei dati di input**

Lo scopo di questo modulo della GUI è di consentire la selezione dei file da elaborare mediante un'esecuzione dello strumento PLC Testbench e consentire la configurazione dell’insieme degli algoritmi di perdita di pacchetti, degli algoritmi di ricostruzione e delle metriche di valutazione da applicare a ciascun file di input con le relative parametrizzazioni.

Data la natura multifase dell'attività e l'elevato numero di selezioni e scelte coinvolte nel processo supportato da questa parte dell'interfaccia, durante la progettazione l'attenzione è stata posta sul rendere l'interazione quanto più semplice e intuitiva possibile.

Per raggiungere questo obiettivo, è stato deciso di modellare i passaggi principali del processo attraverso un'interazione basata su procedure guidate, in cui ogni passaggio ha uno scopo chiaro e limitato e l'utente segue la sequenza dei passaggi in modo logico. Suddividere un'attività complessa in più attività più semplici presenta diversi vantaggi, come diminuire la probabilità di errori, fornire un contesto più chiaro per la segnalazione degli stessi, rendere più semplice per l'utente sviluppare un modello mentale più accurato del software che sta utilizzando, fornire all'utente un aggiornamento sullo stato di avanzamento nella procedura, ridurre il carico cognitivo.

Ad ogni passaggio viene garantita la coerenza dei dati segnalando eventuali errori all'utente prima che venga intrapreso il passaggio successivo. In questo modo eventuali informazioni errate o mancanti vengono segnalate in un ambito limitato e quindi il problema è più facile da comprendere e risolvere.

Dal punto di vista tecnico, questa è stata individuata fin dall’inizio come la parte della GUI che ha maggiori probabilità di essere influenzata da estensioni o sviluppi futuri, quindi sembrava cruciale renderla il più flessibile e auto-adattante possibile.

Ciò significava che l’interfaccia doveva essere in grado di gestire l’aggiunta di nuovi algoritmi con parametri arbitrari in modo trasparente, senza richiedere alcuna modifica del codice.

Il soddisfacimento di questo requisito è stato ottenuto sfruttando le potenti funzioni introspettive del linguaggio di programmazione Python, che consentono di analizzare la struttura del codice in fase di runtime. Attraverso queste API è possibile scoprire le relazioni tra i componenti software.

Nell'interfaccia utente di PLC Testbench è sufficiente che qualsiasi implementazione di un nuovo algoritmo erediti dalla classe base della propria categoria, sia essa una simulazione di perdita o un occultamento di perdita di pacchetti o un'analisi di output, affinché venga automaticamente esposto nelle schermate corrispondenti con tutte le sue proprietà.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* + 1. **Gestione e monitoraggio delle elaborazioni**

Questo modulo della GUI ha lo scopo di consentire all'utente di monitorare lo stato di un'elaborazione in corso, fornendo una sorta di barra di avanzamento e informazioni come l'ETA (Tempo stimato di arrivo) per le sottofasi.

Ogni elaborazione dello strumento PLC Testbench è denominata "Run" e contiene un'istantanea di tutti i dati di input e tutti i risultati di output generati come le maschere di simulazione delle perdite, i file ricostruiti e le metriche valutate su ciascun file ricostruito. Ciascuno dei passaggi dell'elaborazione può essere mappato sul nodo di un albero dove il nodo radice rappresenta un file di input. Il Run nel suo complesso è quindi rappresentabile come una foresta di alberi.

Poiché l'elaborazione è identica per tutti i file di input in termini di algoritmi che vengono eseguiti sulla traccia originale ed è sequenziale, visualizzare continuamente l'intera struttura di una Run implicherebbe appesantire l'interfaccia con informazioni ridondanti. Si è quindi deciso di rappresentare l’avanzamento a due livelli diversi, ovvero all'interno di ciascun albero di un file di input e a livello globale.

A computer screen shot of a network

Description automatically generated

* + 1. **Analisi dei risultati**

Questo modulo è stato considerato fin dall'inizio come il più importante, quindi gran parte del tempo dedicato all'analisi e allo sviluppo dell'interfaccia utente del PLC Testbench è stato dedicato ad esso.

In questo caso l'obiettivo era fornire un modo semplice per analizzare i risultati sia dal punto di vista qualitativo che dal punto di vista quantitativo.

L'analisi qualitativa è rappresentata principalmente dalla navigazione e dal confronto visivo delle forme d'onda nelle regioni in cui i pacchetti sono andati persi e dall'ascolto dei file audio generati dagli algoritmi di PLC.

L'analisi quantitativa è rappresentata principalmente dal confronto numerico delle prestazioni raggiunte dagli algoritmi PLC secondo una determinata metrica come l'Errore Quadrato Medio, l'Errore Assoluto Medio, l'Energia Spettrale, la Valutazione Percettiva della Qualità Audio (PEAQ) rispetto alla parametri di input specificati.

Un altro aspetto estremamente importante è stato fornire la possibilità di integrare e correlare il più possibile i diversi tipi di informazioni per ottenere una visione più approfondita del comportamento degli algoritmi testati.

I requisiti principali in questo caso erano:

• mostrare chiaramente le parti perdute del file originale

• consentire un facile confronto delle forme d'onda spaziando da un livello molto alto al livello di dettaglio del singolo campione

• dare la possibilità di visualizzare ogni combinazione di forme d'onda nel confronto

• dare la possibilità di riprodurre qualsiasi traccia audio, originale o ricostruita posizionando la testina in qualsiasi posizione semplicemente cliccando sulla forma d'onda

• supportare una gamma molto ampia di livelli di zoom incluso uno a livello di campione

• visualizzare visivamente le metriche calcolate sui file audio ricostruiti

• visualizzare uno spettrogramma del file audio, sia esso originale o ricostruito

La necessità di visualizzare più informazioni contemporaneamente ha reso l'interfaccia dashboard più appropriata e accattivante in questo caso. La dashboard è stata organizzata in 4 sezioni, di cui la prima è focalizzata sulla rappresentazione delle forme d'onda di tutte le versioni audio di un file di input e sulla navigazione tra i file e sulle funzioni di zoom. Le regioni dei campioni perduti vengono visualizzate insieme alla forma d'onda e, quando si fa clic su di esse, i relativi dati di dettaglio vengono caricati nella sezione sottostante.

La seconda sezione offre la possibilità di ispezionare la forma d'onda a livello di campione in qualsiasi regione in cui alcuni campioni sono andati persi. Ciò è estremamente utile per valutare da un punto di vista qualitativo il comportamento dell’algoritmo e per verificare la correttezza dell’implementazione poiché la rappresentazione visiva rende molto più evidente qualsiasi comportamento scorretto.  
La terza sezione ha lo scopo di visualizzare le metriche calcolate sulle versioni ricostruite del file analizzato ed è composta da due diversi grafici, uno contenente i dati provenienti da metriche che producono serie storiche, l'altro che rappresenta i dati provenienti da metriche che producono valori singoli. Infine la quarta sezione mostra lo spettrogramma del file audio corrente, che rappresenta sotto forma di mappa di colori la distribuzione delle frequenze che lo compongono e la loro intensità relativa e assoluta.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated



* + 1. **Ricerca e visualizzazione delle elaborazioni**
  1. **User Interaction e User eXperience**

https://marvelapp.com/blog/design-principles-reducing-cognitive-load/

* 1. **Scelte tecniche**

During the design phase different types of GUI were evaluated and their pros and cons were carefully weighted up. Finally, the decision was made to go for the development of a web application because of the many advantages offered by the web technology, like a wide range of deployment modes (ranging from a local standalone environment to big, distributed infrastructures), the portability of the application on different platforms, the ease of installation.

* + 1. **Architettura dell’applicazione**

The user interface application is made of a Web application composed of two layers: one managing the user interface representation and the interaction with the user, the other providing the backend services to be consumed by the graphical components.

The application therefore can be used both as a standalone application by deploying it on the user machine or in a multi-user environment where the application is deployed on one or more remote servers.

The two layers exchange information via a RESTFul API which, if needed, can be exploited also by third-party services. The API is secured by using OAuth2 protocol.

In the presentation layer, modularity has been achieved through the creation of specialized but highly customizable components that can easily be reused across different pages.

In the backend layer, modularity is provided by the extensibility of the RESTFul API where new functions can be “plugged in” by simply adding new endpoints. This layer is also hiding the details of the interaction with the PLC Testbench from the presentation layer, providing an abstract and stable view of the library functions.

Persistence is based on a NoSQL database, where the data are stored in JSON format in order to provide higher flexibility to accommodate future evolution of the schema. Both on-premise and cloud databases are supported.

The GUI adapts automatically to the PLC Testbench library by using introspection to retrieve the list of the algorithms used for packet loss, PLC, and output analysis. This way any extension made to the underlying tool does not require any manual change to the UI code.

The application architecture has been designed to be as stateless as possible and thus supports very well horizontal scalability, which means it is possible to split the workload among as many instances of the application as it is need by providing proper load balancing in front of them.

Scalability together with containerized distribution makes the application suitable for deployment in an orchestrated environment.

* + 1. **Linguaggi di programmazione utilizzati**

The criteria that drove the selection of the programming languages to be used in the project were the following in order of importance:

1. More portable language are to be preferred over less portable language.
2. More largely adopted and supported language are to be preferred over less widespread language.
3. The languages already used by the PLC Testbench are to be preferred in order to reduce the project maintenance complexity.
4. Languages with better support for scalability are to be preferred over less scalable languages.

Based on this criteria Javascript popped up as the best choice for the core UI layer because of its out-of-the-box integration with all browsers, its being the de-facto standard for the web and the huge number of GUI widgets libraries freely available. On the other side Python seemed like the most natural choice for the Rest API and services’ layer, given its tighter integration with the PLC Testbench tool and the availability of several libraries and frameworks inherently designed for that purpose.

Adopting the same language used by the PLC Testbench tool helped a lot in keeping the project complexity as low as possible, both in terms of languages and in terms of final artifact’s distribution.

I criteri che hanno guidato la scelta dei linguaggi di programmazione da utilizzare nel progetto sono stati in ordine di importanza:

1) È preferibile un linguaggio più portabile rispetto a un linguaggio meno portabile.

2) È preferibile un linguaggio maggiormente adottato e supportato rispetto a un linguaggio meno diffuso.

3) I linguaggi già utilizzati dal PLC Testbench sono da preferire al fine di ridurre la complessità di manutenzione del progetto.

4) Lingue con un migliore supporto per la scalabilità da preferire rispetto a lingue meno scalabili.

Sulla base di questi criteri, Javascript è emerso come la scelta migliore per il layer principale dell'interfaccia utente grazie alla sua integrazione nativa con tutti i browser, al suo essere lo standard di fatto per il web e all'enorme numero di librerie di widget GUI liberamente disponibili. D'altro canto Python è sembrato la scelta più naturale per l'API Rest e il layer dei servizi, data la sua più stretta integrazione con lo strumento PLC Testbench e la disponibilità di numerose librerie e framework intrinsecamente progettati per tale scopo.

L'utilizzo dello stesso linguaggio utilizzato dallo strumento PLC Testbench inoltre ha aiutato molto a mantenere la complessità del progetto quanto più bassa possibile, sia in termini di linguaggi utilizzati che in termini di distribuzione degli artefatti finali.

* + 1. **Frameworks**

Although frameworks in some cases tend to increase the complexity of software projects, in this case it was judged to be worth adopting some. In fact, leveraging available solutions to common problems was seen in this case not only as a way to avoid “reinventing the wheel” and speed up the project delivery, but also as a mean to standardize the code organization and incorporate some important best practices and design-patterns.

The frameworks used in the PLC Testbench UI codebase are the following:

**React JS:** probably the most important one; the library makes declarative, component base development of GUIs much easier, by providing support for components’ lifecycle management, state management, UI rendering management and optimization.

**Node js:** used essentially to manage javascript dependencies through the npm package manager.

**D3js:** for the highly specialized tasks related to visual representation of data in most cases D3js library was used behind the scene by wrapping the d3js code into brand new custom React components.

**Wavejs:** a flexible graphical library to represent layered audio waveforms by SVG (Scalable Vector Graphics).

**Flask:** a lightweight framework providing all the functionalities needed by a REST API based web application.

**MongoDB:** persistence of the data was achieved by means of MongoDB, which is one of the most widespread opensource no-SQL databases. The choice of a no SQL database was mainly influenced by the fact that this technology does not impose a rigid schema on the data and therefore looked more flexible and adaptable in case of future extensions of the software.

**PyMongo:** a library for Python programming language that makes interacting with MongoDB much more convenient by providing higher level abstractions over the data layer.

Sebbene in alcuni casi i framework tendano ad aumentare la complessità dei progetti software, in questo caso si è ritenuto che valesse la pena adottarne alcuni. Infatti, sfruttare le soluzioni disponibili ai problemi comuni è stato visto in questo caso non solo come un modo per evitare di “reinventare la ruota” e accelerare la realizzazione del progetto, ma anche come un mezzo per standardizzare l’organizzazione del codice e incorporare alcune importanti best-practice e design-patterns.

I framework utilizzati nel codice dell'interfaccia utente PLC Testbench sono i seguenti:

React JS: probabilmente il più importante; la libreria semplifica notevolmente lo sviluppo dichiarativo basato su componenti della GUI, fornendo supporto per la gestione del ciclo di vita dei componenti, la gestione dello stato, la gestione e l'ottimizzazione del rendering dell'interfaccia utente.

Node js: utilizzato essenzialmente per gestire le dipendenze javascript tramite il gestore pacchetti npm.

D3js: per le attività altamente specializzate legate alla rappresentazione visiva dei dati, nella maggior parte dei casi la libreria D3js è stata utilizzata dietro le quinte racchiudendo il codice d3js in nuovissimi componenti React personalizzati.

Wavejs: una libreria grafica flessibile per rappresentare forme d'onda audio stratificate tramite SVG (Scalable Vector Graphics).

Flask: un framework leggero che fornisce tutte le funzionalità necessarie a un'applicazione web basata su API REST.

MongoDB: la persistenza dei dati è stata ottenuta mediante MongoDB, che è uno dei database opensource no-SQL più diffusi. La scelta di un database no SQL è stata influenzata principalmente dal fatto che questa tecnologia non impone uno schema rigido sui dati e quindi appariva più flessibile ed adattabile in caso di future estensioni del software.

PyMongo: una libreria per il linguaggio di programmazione Python che rende l'interazione con MongoDB molto più conveniente fornendo astrazioni di livello superiore sul livello dati.

* + 1. **Packaging e distribuzione**

With regard to packaging and distribution, the objective was a zero-setup installation of the tool and the proper technology to achieve it was considered to be Docker containers.

The PLC Testbench UI is available as a docker public image built from the master branch of the publicly available GitHub repository of the project. The only prerequisite to get the tool running in standalone mode is having a Docker installation on the target machine.

In order to start the application two containers have to be created, one to host a MongoDB instance and one to host the Flask web application. The template for command lines to launch each container are provided in the README.md file of the PLC Testbench UI’s GitHub project and must be customized in the parts which are necessarily specific to each user’s environment, like volume or SSL certificates setup.

This means that the PLC Testbench UI can run on any platform where Docker can run, like all Linux distributions, Windows and MacOS.

The Docker containerization, together with the packages installer tool from Python also helped in managing the huge amount of transitive dependencies brought into the project by specialized libraries for audio manipulation like gstreamer, librosa or by others for more general purposes like tensorflow.

Per quanto riguarda la pacchettizzazione e la distribuzione, l'obiettivo era un'installazione dello strumento che prevedesse un setup estremamente semplice e la tecnologia adeguata per realizzarla è stata quella dei contenitori Docker.

L'interfaccia utente PLC Testbench è disponibile come immagine pubblica docker creata dal ramo master del repository GitHub disponibile pubblicamente del progetto. L'unico prerequisito per far funzionare lo strumento in modalità autonoma è avere un'installazione Docker sul computer di destinazione.

Per avviare l'applicazione è necessario creare due contenitori, uno per ospitare un'istanza MongoDB e uno per ospitare l'applicazione web Flask. Il modello per le righe di comando per avviare ciascun contenitore è fornito nel file README.md del progetto GitHub di PLC Testbench UI e deve essere personalizzato nelle parti che sono necessariamente specifiche per l'ambiente di ciascun utente, come l'impostazione del volume o dei certificati SSL.

Ciò significa che l'interfaccia utente di PLC Testbench può essere eseguita su qualsiasi piattaforma su cui è possibile eseguire Docker, come tutte le distribuzioni Linux, Windows e MacOS.

La containerizzazione Docker, insieme allo strumento di installazione dei pacchetti Python, ha anche aiutato a gestire l'enorme quantità di dipendenze transitive introdotte nel progetto da librerie specializzate per la manipolazione audio come gstreamer, librosa o da altre più generalistiche e vaste come tensorflow

1. **Risultati**

The most important results achieved throught the PLC Testbench UI development can be considered the following:

* Support to the validation and debugging of the PLC Testbench tool
* Extension of the tool usability
* Improved user experience for the PLC Testbench tool
  1. **Supporto nella validazione e nel debug dell’applicazione PLC-TestBench**

During the development of the Graphical User Interface, it has become evident how the GUI can support and speed-up the debugging of the PLC algorithm, thus acting also as a validation tool for the PLC Testbench itself.

* 1. **Ampliamento dei potenziali utilizzatori dell’applicazione PLC-TestBench**

Providing a GUI for the Testbench Tool improves the overall user experience, makes the learning curve of the tool less steep and extends its usability to users not having a programming background.

* 1. **Miglioramento della user experience**

Compared to the out of the box representations provided by the PLC Testbench tool which are essentially static images, the PLC Testbench UI allows real-time exploration of the data at different level of detail. The detail level can be selected interactively, depending on the needs and the data are more integrated by being displayed, when comparable, as multiple time-series on the same graph.

In order to improve the user experience the GUI has been made responsive so that it can automatically adjust to different devices (PC, tablet, mobile).

The interface has been designed to be user-friendly, intuitive, and as fast as possible, considering the large dataset inherently involved in audio processing applications and the overhead related to charting functions. Whenever possible subsampling or caching techniques have been applied to minimize latency and network bandwidth waste.

* 1. **Sviluppi futuri**

Nonostante fornisca un'implementazione per tutti i casi d'uso di base riconosciuti e sia pienamente operativa, l'interfaccia utente di PLC Testbench ha ovviamente ancora molto margine di miglioramento. Futuri miglioramenti della GUI potrebbero essere ovviamente dettati dall'evoluzione del Testbench PLC stesso, ma potrebbero anche essere più strettamente legati all'interazione con l'utente e potrebbero essere rappresentati ad esempio da:

* estendere il supporto ad altri formati di file audio, come mp3
* fornire una funzione per esporre i log del Testbench del PLC
* implementare funzioni di ricerca sui risultati di output
* implementare diversi temi dell'interfaccia utente
* internazionalizzare l'interfaccia utente introducendo il supporto per altre lingue
* implementazione di casi d'uso di collaborazione tra utenti
* estensione dei provider di identità supportati per il login (attualmente solo Google)
* sviluppo di funzioni di export per i formati più diffusi (Excel, CSV, ecc.)

1. **Conclusione**

I would like to thank Pr. Luca Turchet and Ph.D Luca Vignati for constantly supporting me during the development and providing precious suggestions and feedback.

**Bibliography**

[1] Coulouris G. F., Dollimore J. e Kindberg T, Distributed Systems: concepts and Design, 1994

[2] Dalal N., Triggs B., Histograms of Oriented Gradients for Human Detection, 2005

[3] Donoho D. L., Compressed Sensing,

[4] http://www.ictbusiness.it/

***Note***

*In the bibliography, all the sources consulted for the dissertation have to be cited and listed in alphabetical order by the first author's surname.*

*According to the source material, the quotation has to be as follows:*

*BOOKS*

*Surname and initial/s of the name/s of the author/s, date of edition, publishing house and (if applicable) number of edition.*

*JOURNAL ARTICLES*

*Surname and initial/s of the first name/s of the author/s, title of the article, name of the journal, volume number, issue number and page numbers.*

*CONFERENCE PAPERS*

*Surname and initial/s of the name/s of the author/s, year of the conference, title of the article, name of the conference, place of the conference, conference dates, page numbers.*

*CITING WEB RESOURCES*

*The consulted webpages have to be listed in alphabetical order.*

*It is necessary to:*

* *Copy the specific URL (the web address) of the consulted webpage*
* *If available, indicate the surname and first name of the author/s, the title and subtitle of the text*
* *If available, indicate the last date you retrieved the webpage (day/month/year).*

**Attachment A Source code repository**

The source code of the project is available on the following GitHub public repository:

<https://github.com/stefano-dallona/plc-testbench-ui>

**PLC Testbench UI’s GitHub source code repository** ….

**Attachment B Docker image**

**PLC Testbench UI’s Docker image**

A pre-built image of the application is publicly available at the following URL:

<https://hub.docker.com/r/stdallona/plc-testbench-ui>

**Running pre-built image and building custom images from source code**

If the pre-built image does not fit your needs you can build a suitable image by customizing the Dockerfile located in the root directory of the source code project on published on GitHub.

**Attachment C CI/CD Pipelines**

In order to make release cycles faster and more reliable a Continuous Integration/Continuous Deployment pipeline has been setup on the free cloud-based tool Circle CI.

The pipeline has been developed exploiting the pipeline as code feature of Circle CI, based on yaml language and is stored in the ***.circleci/config.yml*** file in the GitHub source code repository of the PLC Testbench UI project.

**CircleCI CI/CD Pipelines**

# Use the latest 2.1 version of CircleCI pipeline process engine.

# See: https://circleci.com/docs/configuration-reference

version: 2.1

# Define a common Docker container and environment for jobs

executors:

  docker-publisher:

    # Define the image tag

    environment:

      IMAGE\_TAG: stdallona/plc-testbench-ui:1.0.3

    # Use `docker:stable` as the Docker image for this executor

    docker:

      - image: docker:stable

# Define a job to be invoked later in a workflow.

# See: https://circleci.com/docs/configuration-reference/#jobs

jobs:

  build:

    # Use docker-publisher from above as the Docker container to run this job in

    executor: docker-publisher

    # Add steps to the job

    # See: https://circleci.com/docs/configuration-reference/#steps

    steps:

      - checkout

      # Set up a separate Docker environment to run `docker` commands in

      - setup\_remote\_docker

      - run:

          name: Build Docker image

          command: docker build --tag "${IMAGE\_TAG}" .

      # Archive and persist the Docker image

      - run:

          name: Archive Docker image

          command: docker save --output image.tar "${IMAGE\_TAG}"

      - persist\_to\_workspace:

          root: .

          paths:

            - ./image.tar

  push:

    # Use docker-publisher from above as the Docker container to run this job in

    executor: docker-publisher

    steps:

      # Set up a separate Docker environment to run `docker` commands in

      - setup\_remote\_docker

      # Load and un-archive the Docker image

      - attach\_workspace:

          at: /tmp/workspace

      - run:

          name: Load Docker image

          command: docker load --input /tmp/workspace/image.tar

      # Log in to Docker Hub and push the image

      - run:

          name: Publish Docker image

          command: |

            echo "${DOCKERHUB\_PASS}" | docker login --username "${DOCKERHUB\_USERNAME}" --password-stdin

            docker push "${IMAGE\_TAG}"

# Run the two different jobs as a sequenced workflow

workflows:

  version: 2

  build-push:

    jobs:

      # Run the build first

      - build

      # Push the image second

      - push:

          # Build needs to finish first

          requires:

            - build

          # Only push images from the master branch

          filters:

            branches:

              only: master