A school intercom system that filters inappropriate phrases and plays emails as announcements in the voice of the sender.

Final Report

I. Ramedies
16023405

Submitted as partial fulfilment of the requirements of Project EPR402
in the Department of Electrical, Electronic and Computer Engineering
University of Pretoria

November 2022

Study leader: Mr. A. Oloo

Part 1. Preamble

This report describes the work I did in designing an intercom system that receives emails and plays the text out as speech, using speech theory and digital signal processing.

Project proposal and technical documentation
This main report contains an unaltered copy of the approved Project Proposal (as Part 2 of the report).

Technical documentation appears in Part 4 (Appendix).

All the code that I developed appears as a separate submission on the AMS.

Project history
This project adapted the formant synthesizer developed by Klatt (1980). The properties of speech acoustics were obtained through the use of Praat recording software and Klatt parameters. Software for using an email application programming interface (API) was taken from PythonCode (2022). The databases used for parts of speech (POS) tagging and phoneme transcription was provided by the NLTK database and the CMU institution. The rest of the work reported on here, is entirely my own.

Language editing
This document has been language edited by a knowledgeable person. By submitting this document in its present form, I declare that this is the written material that I wish to be examined on.

My language editor was mrs. Noeroenniesa Ramedies .

_____	07/11/2022
Language editor signature	*Date*

Declaration

I, Ishaque Ramedies understand what plagiarism is and have carefully studied the plagiarism policy of the University. I hereby declare that all the work described in this report is my own, except where explicitly indicated otherwise. Although I may have discussed the design and investigation with my study leader, fellow students or consulted various books, articles or the internet, the design/investigative work is my own. I have mastered the design and I have made all the required calculations in my lab book (and/or they are reflected in this report) to authenticate this. I am not presenting a complete solution of someone else.

Wherever I have used information from other sources, I have given credit by proper and complete referencing of the source material so that it can be clearly discerned what is my own work and what was quoted from other sources. I acknowledge that failure to comply with the instructions regarding referencing will regarded as plagiarism. If there

is any doubt about the authenticity of my work, I am willing to attend an oral ancillary examination/evaluation about the work.

I certify that the Project Proposal appearing as the Introduction section of the report is a verbatim copy of the approved Project Proposal.

I. Ramedies

07/11/2022

Date

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

LA	Linguistic analysis
NLP	Natural language processing
DSP	Digital signal processing
TTS	Text-to-speech
STT	Speech-to-text
API	Application programming interface
POS	Part(s) of Speech
HMM	Hidden Markov model
WER	Word error rate
IIR	Infinite impulse response
LUT	Lookup Table
DNN	Deep Neural Network
PSOLA	Pitch Synchronous Overlap and Add
MFCC	Mel-frequency cepstral coefficients
SBC	Single board computer
PCB	Printed circuit board
Email	Electronic mail
AM	Amplitude Modulation
LPF	Low-pass filter
HPF	High-pass filter
BPF	Band-pass filter
UML	Unified Markup Language

Part 2. Project definition: approved Project Proposal

This section contains the problem identification in the form of the complete approved Project Proposal, unaltered from the final approved version that appears on the AMS.

For use by the Project lecturer	Approved		Revision required	
Feedback				

To be completed by the student							
PROJECT PROPOSAL 2022				Project no	AO5	Revision no	2
Title	Surname	Initials	Student no	Study leader (title, initials, surname)			
Mr	**Ramedies**	**I**	16023405	Mr A Oloo			
Project title							
A school intercom system that filters inappropriate phrases and plays emails as announcements in the voice of the sender.							

Language editor name	Language editor signature
N. Ramedies	
Student declaration I understand what plagiarism is and that I have to complete my project on my own.	Study leader declaration This is a clear and unambiguous description of what is required in this project. Approved for submission (Yes/No)
Student signature	Study leader signature and date

1. Project description

What is your project about? What does your system have to do? What is the problem to be solved?

The problem addressed in this project, is the lack of physical contact between teachers and students in schools, due to traveling between classrooms after periods or possibly teacher unavailability. Teachers that make announcements and students that hear them, may not able to be in the same physical location. Employing an individual, to collate announcements would not be an efficient solution to this problem. Thus, the problem addressed in this project is to develop a school intercom system that will make announcements on behalf of the teacher, eliminating the need for announcers or any human intervention.

The concept is to have a system that receives a text-based email from a teacher, then announces this email to the students, in the voice of the teacher that sent the email. This may be accomplished by integrating natural language processing techniques, speech synthesis, and digital signal processing into a single intercom system. The system should also account for any inappropriate phrases received from the email and should be able to play the announcement over an integrated speaker, so that multiple students in a classroom are able to hear it.

2. Technical challenges in this project

Describe the technical challenges that are *beyond* those encountered up to the end of third year and in other final year modules.

2.1 Primary *design* challenges

A main design challenge would be accounting for multiple user's voices. Another is hardware allocation of the system resources. The overall system should be designed to work in real time, so all training of artificial intelligence systems, if used, will have to be conducted, prior to final product compilation. The linguistic analysis technique used will have an effect on the processing time of the overall implementation, and each technique has benefits over others. Selection of a specific technique, or development thereof, will be seen as a design challenge.

2.2 Primary *implementation* challenges

A system implementation challenge, is managing waveforms and assigning it to analyzed text, correctly. Testing results of synthesized audio, when compared to natural language audio, might be difficult to achieve, unless extra functional components are developed. It also might be challenging to develop bandpass filters with very narrow passbands, depending on the support that the development platform provides. Another implementation challenge would be within the natural language processing and linguistics analysis algorithms. The English language contains many rules and it might prove challenging to combine multiple rules without making design errors within the algorithms.

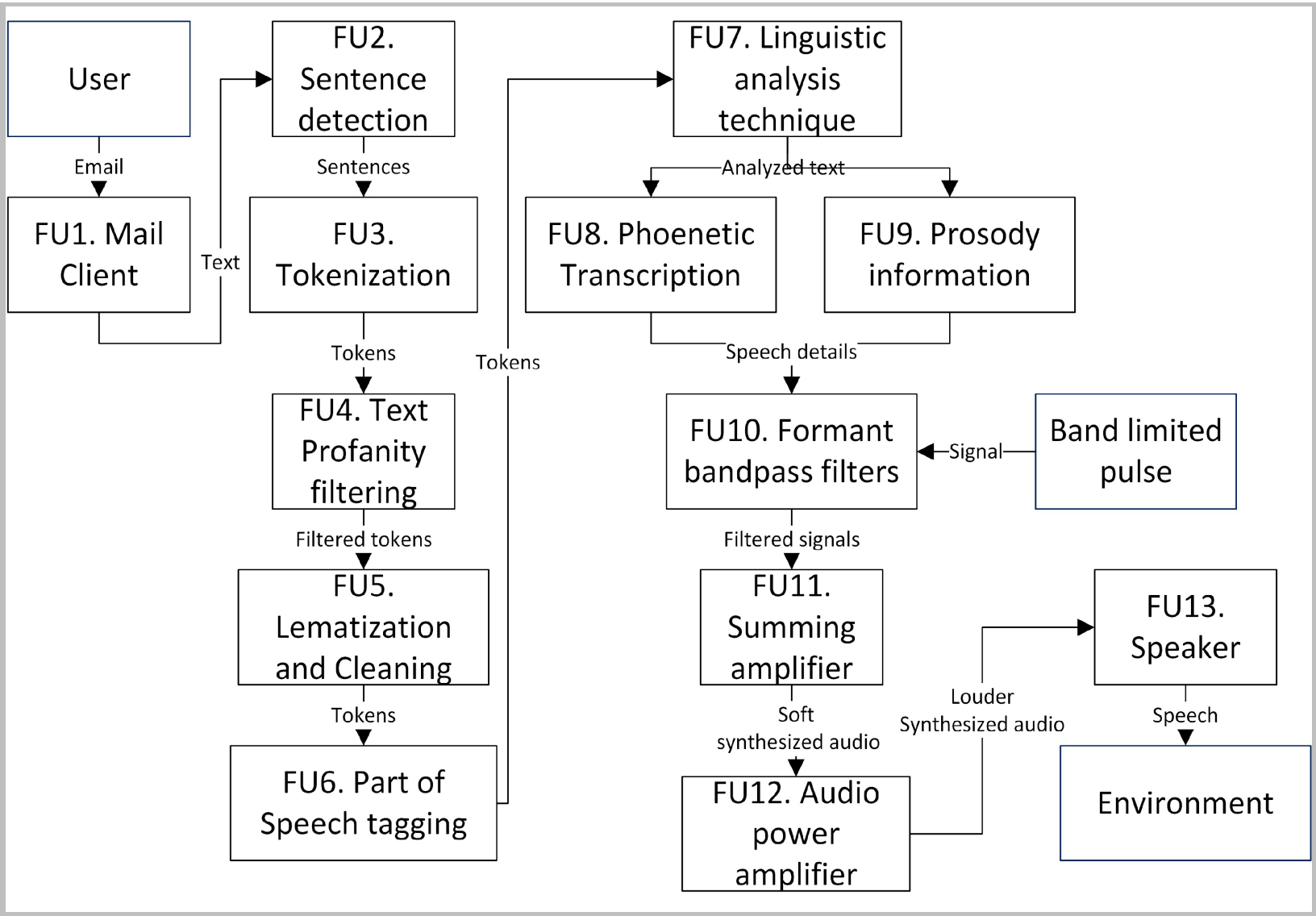
3. Functional analysis

3.1 Functional description

Describe the design in terms of system functions as shown on the functional block diagram in section 3.2. This description should be in *narrative format*.

The user input will be an email text announcement that is sent to the system via the internet. FU1 is a mail client that will receive these emails. The text is extracted from the email whereafter, natural language processing (NLP) and linguistics analysis is performed on the text from FU2 to FU7. The process involves detecting sentences (FU2), and creating tokens from those sentences (FU3). These tokens are run through a profanity filter (FU4) that removes all inappropriate phrases contained within a set of tokens, by checking the tokens against a database of profane words. The output of the profanity filter will be a set of tokens with some marked to be inappropriate, if applicable. The set of marked tokens will then go through a lemmatization and cleaning process, in FU5, where sentences are correctly formatted. Each token is then tagged with its part of speech (FU6), relevant to the whole sentence. A linguistics analysis technique, such as a rule-based technique, is then performed on the output, in FU7, to determine the meaning of the tagged sentences. The analyzed text is sent to FU8, a phonetic transcription function, that will append phonemes to every part of the word, representing what it will sound like. The analyzed text is simultaneously sent to a prosody generator (FU9), which determines the prosody information, i.e., the pitch, loudness, and duration of the words. The information generated from the combination of FU8 and FU9 are then used in FU10 and FU11, that make up a waveform generator or synthesizer. The synthesizer artificially creates a sound signal from the retrieved prosody information using formant synthesis. Formant synthesis combines 5 different frequencies (formants), of a user's voice, utilizing digital bandpass filters (FU10), into a single voice signal, by using a summing amplifier (FU11). Depending on the strength of the synthesized signal, it may be amplified through an audio power amplifier (FU12). The synthesized speech would then be output as an announcement through FU13, a speaker. The output should be heard in the classroom environment by students.

3.2 Functional block diagram



4. System requirements and specifications

These are the core requirements of the system or product (the mission-critical requirements) in table format IN ORDER OF IMPORTANCE. Requirement 1 is the most fundamental requirement.

	Requirement 1: the fundamental functional and performance requirement of your project	Requirement 2	Requirement 3
1. Core mission requirements of the system or product. Focus on requirements that are core to solving the engineering problem. These will reflect the solution to the problem.	The announcement has to be broadcast in a synthetic voice that sounds similar to the sender's voice, in terms of the pitch, loudness, length, and strength.	The system should be able to convert an arbitrary text input to speech information in natural language.	The message that is announced, should be filtered of all explicit inappropriate phrases. The system should consist of a comparative database with which to deal with all profanity.
2. What is the target specification (in *measurable* terms) to be met in order to achieve this requirement?	The fundamental frequencies, amplitude, duration, and amplitude dynamics of the formants in a speaker's voice should be band limited. Formant peaks should not differentiate more than 10% of the original voice.	A linguistics analysis module produces word error rates (WER) of 40% or below, after processing text. Processing for short non-complex sentences comprising of around 15-20 words should provide speech information to the synthesizer within 2 seconds.	The accuracy of a large sample set, should be at 100%, to ensure that the profanity filter is operable for explicit profanity.
3. Motivation: *how* or *why* will meeting the specification given in point 2 above *solve the problem*? (Motivate the *specific* target specification selected)	Adults can differentiate a 1-2% change in 4000Hz tones. At lower voice frequencies reaching maximum ranges of 155Hz - 255Hz, people have less sensitive hearing. A 5-10% change in this metric will yield a similar sounding synthetic voice.	Developed STT transcribers of human voices yield WERs between 15 to 40%. It is expected that synthesized speech will lie within this range. Performance measure is a non-deterministic polynomial (NP) problem, so a cubic bound complexity is assumed.	A 100% profanity filtering margin ensures that evident profanity is removed. This relies on the assumption that teachers are responsible in what they announce.
4. How will you demonstrate at the examination that this requirement (point 1 above) and specification (point 2 above) has been met?	Recordings of multiple announcers' voices will be played, in conjunction with the synthesized speech for auditory comparison. Waveforms of the speech will also be displayed and may be compared.	Text data will be fed into the system, wherein a linguistics analysis function will provide a synthesizer with speech information. A STT transcriber will interpret the perceived announcement after synthesis. WER and execution time will be displayed.	Text data sets, containing profanity, will be input into a profanity filter. The filter will identify explicit profanity and the corresponding specification may be observed.
5. Your own design contribution: what are the aspects that *you will design and implement yourself* to meet the requirement in point 2? If none, *remove this requirement.*	A formant text-to-speech (TTS) synthesizer, consisting of DSP techniques, will be designed and implemented. A prosody generation function will be implemented.	A linguistics analysis module, will be implemented to generate speech information. Natural language processing functionality will be implemented to reduce possibility of error for the linguistics analysis module.	A text-based profanity filter will be developed and implemented.
6. What are the aspects to be taken off the shelf to meet this requirement? If none, indicate "none"	A microcontroller board, capable of digital signal processing, will be purchased. Software to record user audio will be used. A speaker will be used for audio playback of the processed signal.	A microcontroller board with adequate processor speed. A speech-to-text (STT) software platform to evaluate the system will be used. An audio recorder will be used to store system output. A database of English words will be used.	None.

System requirements and specifications page 2			
	Requirement 4	Requirement 5	Requirement 6
1. Core mission requirements of the system or product. Focus on requirements that are core to solving the engineering problem. These will reflect the solution to the problem.	A mail system should receive emails from a user.		
2. What is the target specification (in *measurable* terms) to be met in order to achieve this requirement?	The text data that is sent by the user in the email, should be correctly reflected by the receiving end of the system, with no changes made to the input.		
3. Motivation: *how* or *why* will meeting the specification given in point 2 above *solve the problem*? (Motivate the *specific* target specification selected)	To ensure that the mail system receives emails correctly, it must prohibit the receiving end from making any changes to the user's input.		
4. How will you demonstrate at the examination that this requirement (point 1 above) and specification (point 2 above) has been met?	A display screen should indicate notification upon receiving the email. The output of the text should be displayed and confirmed by the user.		
5. Your own design contribution: what are the aspects that *you will design and implement yourself* to meet the requirement in point 2? If none, *remove this requirement.*	Implementation of an email text display feature.		
6. What are the aspects to be taken off the shelf to meet this requirement? If none, indicate "none"	A Wi-Fi module, capable of Internet connection, and an LCD screen, for debugging and display purposes, will be purchased. An email library will be used for the implementation of the mail client.		

5. Field conditions

These are the REAL-WORLD CONDITIONS under which your project has to work and has to be demonstrated.

	Field condition 1	Field condition 2	Field condition 3
Field condition requirement. In which field conditions does the system have to operate? Indicate the one, two or three most important field conditions.	The intercom system has to be audible enough for a noisy indoor room environment. It should be operable from any location with internet access.		
Field condition specification. What is the specification (in measurable terms) for this field condition?	A standard noisy environment for a classroom of students ranges between 65-70 dB.		

6. Student tasks

6.1 Design and implementation tasks

List your primary design and implementation tasks in bullet list format (5-10 bullets). These are *not* product requirements, but *your* tasks.

- The intercom system must be designed and implemented.
- An appropriate linguistics analysis technique has to be selected.
- Audio data from multiple people will have to be collected.
- Large sentence text data sets, with profanity, have to be collected and sorted.
- A physical cover for the system will have to be designed and built.
- The system should be integrated onto a PCB board.
- Simulated versions of the functions should be developed for debugging and demonstration purposes.

6.2 New knowledge to be acquired

Describe what the theoretical foundation to the project is, and which new knowledge you will acquire (*beyond* that covered in any other undergraduate modules).

- The student will be required to master theoretical background knowledge on Natural Language Processing and Speech Synthesis.
- The student will be required to acquire knowledge on Linguistics Analysis techniques.
- The student will need to learn how to work with audio data sets.
- The student will be required to learn about rulings of speech in the English language.
- The student will have to learn how various aspects of digital signals affect the auditory output of speech.

Part 3. Main report

1. Literature study

A literature study is presented in the sections to follow.

1.1. Background and context of the problem

Speech is the expression of thoughts and feelings by articulate sounds, and it is the most natural and convenient form of communication for a human. Speech technologies are those that can recognise, analyse and understand spoken speech. The two primary forms of speech technology are speech recognition and speech synthesis technologies, where one form of technology uses input speech to perform text processing, and the other produces speech from a text processing module. The project entails the latter, a text-to-speech (TTS) intercom system. TTS systems benefit the population by assisting those with literacy difficulties, and reduced vision, among others. Therefore, the design and implementation of such a system will positively affect the people that use it. **Figure 1** illustrates the main components involved in a TTS system.

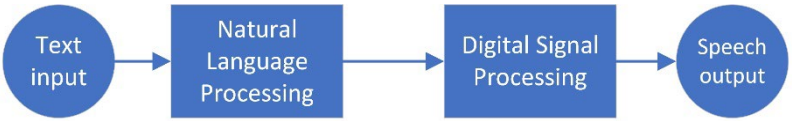


Figure 1 Block diagram of TTS system

Several subsystems have to be considered to execute the intercom system. These main subsystems are the natural language processing (NLP) unit and speech synthesiser. In addition, depending on the application of the TTS system, it may include a variety of other subsystems, such as a linguistics analysis (LA) module, profanity filter, and spell checker, to name a few. Implementing these subsystems requires knowledge of digital signal processing (DSP) techniques, speech processing theory, optimisation algorithms, search algorithms, and signal feature extraction. A literature study is conducted to summarise techniques and work related to the intercom system.

1.1.1. Natural language processing

Natural language processing is a fundamental, essential component of text-to-speech synthesis. The front-end development of TTS systems involves NLP, intending to attach prosody or sound information to the input text. Zhigang provides information on how the components involved in the front-end development work together to provide the necessary output to the DSP unit [1]. The NLP unit consists of a pre-processing block, morphological analysis unit, contextual analysis unit, syntactic analysis unit, phonetisation module, and prosody generation function. Reichel describes some generic NLP modules associated with TTS synthesis [2]. The modules are text normalisation, part-of-speech (POS) tagging, grapheme-to-phoneme conversion and word stress.

(a) Pre-processing

Pre-processing of text rectifies any error or abnormalities in the text received as input. Text such as abbreviations, numbers, acronyms, and dates must be converted into full text. This process is a form of text normalisation, but it really puts all input text into a recognisable format for the computer to perform operations on. Pre-processing also breaks text into sentences by using classifiers to determine where sentences are split. There are several steps involved in pre-processing. However, the type of system dictates which of the steps are necessary to be implemented.

(b) Morphological analysis and Contextual analysis

Morphological analysis and contextual analysis are usually associated together. This process involves taking the sentences from a pre-processing module and dividing it into a sequence of words. The morphological analysis involves this word segmentation, also known as text normalisation, and the contextual analysis determines the context of each segmented word or token about the sentence. So, for example, whereas classifiers may be used to determine how to split up sentences, whitespaces often indicate the end of a word for text normalisation in the English language.

(c) Syntactic analysis

The syntactic analysis aims at predicting the prosody of the input sentence by parsing it. Therefore, an essential aspect of this module is performing POS tagging. This gives information about the grammar of the sentence by determining each word's part of speech. There are several tagging methods to consider, such as rule-based tagging, stochastic tagging, transformation-based tagging, and hidden Markov model (HMM) tagging [3].

(d) Phonetisation

Phonetisation involves the generation of phonetic symbols, or phonemes, for each word in the text. Phonetisation is done by constructing a lexicon, a dictionary of words with their pronunciations. The lexicon would serve as a look-up table (LUT) for the words. For words or special characters that cannot be identified as part of the lexicon, it would be necessary to include an algorithm that would be able to generate the phone sequence for that word.

(e) Prosody generation

Prosody generation is the generation of prosodic information for the synthesiser, and it would specify signal attributes like pitch curve information, duration, and pause information. This model affects the naturalness and intelligibility of the speech output to a certain extent. Generating prosodic information is done if points of emphasis are identified, along with punctuation tokens and phoneme indices in a lexicon.

1.1.2. Linguistic analysis

NLP consists of computational linguistics, the rule-based human language modelling, with statistical and machine learning models. In general, linguistics analysis approaches that should be avoided for this system implementation are; rules-based and neural network implementations, as they are data-driven and may be time-consuming. The book "*Theory and applications of digital speech processing*" [3] provides a better insight into linguistics analysis and the desired output. It also provides knowledge on TTS systems, digital signal processing techniques, and system design considerations. Linguistics analysis is thus the formatting approach for the NLP module.

1.1.3. Speech synthesis methods

Speech generation can be divided into two main categories, namely, rule-driven synthesis techniques and data-driven synthesis techniques. The concept of rule-driven techniques is to generate speech output according to rules developed by simulating the articulation or acoustic process. On the other hand, data-driven techniques rely more on recorded speech data or parameters derived from speech data. Rule-driven methods include articulatory and formant synthesis methods, while data-driven techniques consist of concatenative, unit selection, HMM, and deep neural network (DNN) synthesis. These synthesis methods are described for comparative purposes.

(a) Articulatory synthesis

Articulatory synthesis is performed by collecting observations of physical human pronunciation. This synthesis requires rules for an articulatory model to be developed beforehand so that the model's parameters may be adjusted to change the synthesized sound. The parameters included in the model to be controlled the tongue tip position and height, tongue position and height, lip aperture, and lip protrusion. Unfortunately, due to the difficulty in simulating human pronunciation through these parameters, this synthesis method often leads to inaccuracies in sound quality.

(b) Formant synthesis

Formant synthesis simulates the acoustic process instead of the physical pronunciation, emphasising the output's sound. *Formants* are the main frequencies that distinguish sounds from each other. There are usually five formants visible in a spectrum of an audio signal. In 1979, Dennis H. Klatt published a report on formant synthesis containing a realised speech synthesiser to provide a flexible research tool for studying aspects of speech [4]. The Klatt synthesizer model is based on *the Acoustic Theory of Speech Production*, developed by Fant [6].

The paper demonstrates the result of using digital resonators in parallel or cascade configurations while allowing the user to specify variable control parameter data. The paper also contains synthesiser design descriptions, motivations, computer requirements and strategies for imitating speech utterances. The paper thus provides a reasonable baseline expectancy when considering a formant synthesiser for speech output. Formant synthesisers have the advantages of rule-driven synthesis techniques, and the sound quality of the synthesized speech is better than the articulatory synthesiser, but it does not sound natural.

(c) Concatenative synthesis

Concatenative synthesis was developed in the 1990s to establish more natural-sounding speech. Concatenative synthesis is a data-driven synthesis technique. As the name suggests, it uses various recorded voice units such as words, syllables, phones, and other recordings and concatenates these units to produce speech. The majority of speech features remain unchanged when compared to natural speech, but the technique relies on the speech corpus that is used to produce the output speech for good quality speech. This approach may be detrimental for systems with low memory requirements, and discontinuities around the speech units should be handled correctly for this approach to be stable. DSP techniques, such as the synchronous pitch overlap and add (PSOLA) method, were developed in 1986 to modify the speech units when concatenating them.

(d) Unit Selection synthesis

The unit selection synthesis is an improvement of concatenation synthesis and was first introduced in 1992 by Sagisaka [7]. In concatenation synthesis, the output speech would often sound more unnatural when the speech signals were modified using the PSOLA method. Unit selection stores multiple instances of units and selects the unit that matches prosodic features more accurately so that fewer changes are made to the unit for a more natural-sounding output. The model's introduction made the speech more natural sounding but still has the same problems of selecting error units and storing a large speech corpus. The approach is also not as flexible since the voice quality could not be improved.

(e) Hidden Markov model (HMM) synthesis

An HMM synthesiser makes use of statistical parametric synthesis techniques. The benefit of this approach is the reduced demand for memory for the storage of units. Instead of storing units, speech parameters are stored. The HMM is a model that has to be trained with probabilities initially, and once the probabilities are known, the trained model is used to apply the synthesis. Training is conducted similarly to how it is done in speech recognition systems.

Initially, the features that have to be trained are decided. The most commonly used features are derived from Mel-frequency cepstral coefficients (MFCC) and its dynamic features and the LogF0 model and its dynamic features. The MFCC and LogF0 are algorithms that are used to extract audio features. Features have to be extracted per frame, and have to be stored in feature vectors. The feature vectors are used to train the HMM [8]. Synthesis requires the HMM, a duration model, and a synthesis filter to generate the feature vectors.

Zen developed a statistical speech synthesizer with the HMM as a basis for the model [9] and noted a significant limitation in the quality of the synthesized speech. Although the HMM model has a less memory-demanding corpus and trains in less time than unit-selection synthesis systems, the vocoding aspects, accuracy of acoustic models, and over-smoothing proved to be significant influences on the degrading quality of speech.

(f) Deep Neural Network Model synthesis

HMM-based synthesis systems required more than 50 contextual factors to be considered with acoustic modelling. Furthermore, the more typical top-down decision tree structure was used to ensure that all contextual factors were considered. However, this structure yields limitations. For example, expressing complex context dependencies is inefficient, and dividing the input space by using separate parameters for each region results in fragmenting the data. The limitations of the structure, therefore, degrade the accuracy of acoustic models and, as a result, speech quality.

The DNN uses an alternative structure consisting of a multilayer neural network. In a DNN, unsupervised learning occurs before supervised learning. It uses a weight structure to tweak the connections between layers of the network, often using a large amount of training data. For a DNN, if there is more training data that is reliable, then the training period takes much longer to complete. However, the result of the prediction and classification is improved. DNNs are mainly used to model the features of text and speech sounds and the vectors of phonetic signals.

An investigation into the performance analysis of DNN-based speech synthesis systems was performed by Chen [10]. The result is that the DNN is more beneficial than the HMM during voiced, unvoiced, and quasi-cycle prediction classification. However, with this improvement, it is more costly computation-wise, and often the prediction of the pitch could be better than the HMM-based synthesiser. The DNN is still a topic for continued research in modern applications.

1.2. Application summary

A summary of the relevant knowledge is presented in the sections to follow.

1.2.1. Overview of NLP components and linguistic analysis

Natural language processing components require a certain amount of coding knowledge. In addition, performance-enhancing algorithms could be utilised to optimise the outcome of the system. Optimisation algorithms are decided on based on the task to be implemented and are dependent on the designer's preferences. Therefore, there is much room for decision-making in NLP, although it depends highly on the input required and the desired outcome.

The components and purpose of the system should be considered individually for design to make decisions on how to approach NLP. For instance, pre-processing involves several choices. These choices revolve around the type of expected input presented to the system. For this system, the input for the pre-processing component is an email announcement received from a teacher for a school environment. If assumptions are made that the teacher will not send emoticons, URLs, and spells correctly, this limits the amount of pre-processing needed to handle casing, punctuations, and other tasks. Similarly, other components will have to be individually considered and what needs to be implemented based on those considerations is merely a design choice affected by the desired system output.

The linguistic analysis applies to the entire process of NLP. The aim is to understand the content of the text in most cases. How well this content should be understood depends on the accuracy of NLP components. The literature indicates that statistical methods are used in most cases for higher accuracy.

1.2.2. Overview of synthesis techniques

As discussed previously, synthesis techniques are either rule-driven or data-driven. Simply measuring the impact of the two categories narrows down the options for selecting a synthesiser. Rule-driven techniques are used less in modern applications due to the unnatural-sounding speech. However, they are advantageous for their low processing requirements, costs, and flexibility. Data-driven methods thus provide a higher-quality speech. If the system has access to higher-powered processors and large memory banks, then data-driven methods are the best approach.

Of the synthesisers, unit selection synthesis model yields the best speech but is also the least memory efficient. The HMM and DNN were developed to overcome the memory requirements of the unit-selection model, and they produce similar-sounding speech. The HMM and DNN provide the best memory-efficient solutions that give good speech output. In the future, rule-driven and data-driven synthesisers may be merged to yield improvements in synthesis technology.

For the complexity of the synthesisers, the articulatory synthesiser is not practical, as it is very easy to make mistakes in obtaining parameters for the physical movements of the mouth. Articulatory synthesisers also require the design of external components, which is not practical for an intercom system. Formant synthesisers are based on formants which are not challenging to locate in signal spectrograms. Digital signal processing is vital in forming signals made up of formants and leaves little room for mistakes.

Concatenative synthesis is simple to implement as it requires little to no knowledge of any signal processing techniques and merely requires a large corpus of speech units. Unit selection synthesis is similar to concatenative synthesis but requires more algorithmic knowledge and a larger corpus. HMM, models are very complex to implement as they require knowledge of feature extraction algorithms and the ability to recognise elements of feature vectors. DNNs require knowledge of neural networks and waveform synthesis, but if implemented correctly with a large enough corpus, they will yield the best-sounding output.

2. Approach

The approach of the project is considered in the sections to follow.

2.1. Project objectives

The project's primary objective is to develop a profanity-filtering intercom system that relays announcements in the voice of an email sender. The project entails designing and implementing several subsystems, including an email client, an NLP module, a profanity-filtering module, and a speech synthesizer. The approach taken to accomplish these subsystems is discussed in the following sections. First, for the system as a whole, each subsystem was addressed individually and then integrated after verification of their operation.

2.2. Email client

The email client is the subsystem that will accept text input to the system. The development of an email client is not a core requirement of the project; however, its implementation is required. For this reason, an external email application programming interface (API) was used to serve as the email client. Things to consider when implementing the email client are the need for internet access and a personal email address so that emails can be received and accessed. The other requirement of the email client is a display feature to ensure that the system receives the most recent email sent to the system. As the only matter of importance is verification of the email text, the display was handled arbitrarily, i.e., just displaying the email in a browser, as well as storing it in a folder on the device. If multiple users are to use the device, it would be necessary to include an email handler, which identifies the person that sends the email and assigns a synthesized voice to that person’s email.

2.3. Text processing and profanity filtering strategy

After the system receives an email, the initial text-processing elements are applied. The synthesiser requires each word to be broken down into its phonemes, attached with prosody information. Assumptions were also made that emails are received with the correct spelling and punctuation. The relevant NLP components were implemented by considering the synthesiser and text input restrictions. This process also aimed at using algorithms and strategies that would yield the fastest execution time while still being within a reasonable accuracy bound.

For text pre-processing, the text undergoes normalisation and tokenisation because, for POS tagging, only tokens are required to perform contextual analysis. When tokens are acquired, profanity filtering is performed by using a self-defined dictionary and performing a one-to-one comparison with tokens. This method handles explicit profanity, not derivatives thereof, as specified by the project requirements. Alternatively, profanity filtering could have been done by performing a deep analysis using Levenshtein automata. This method would handle derivatives of profanity as well as distorted profanity. However, this would result in an increase in processing time and still achieve the exact project requirement.

The syntactic analysis then performs POS tagging on the tokens using an online corpus. The algorithm used to accomplish POS tagging is the statistical HMM method optimised using the Viterbi algorithm. These algorithms were chosen since statistical methods are preferred for increased accuracy in text-processing algorithms. Also, the

contextual analysis only considers the POS of a word so that it can handle homographs (words that are spelt the same but are pronounced differently). By identifying the POS of a homograph, the correct pronunciation for it is assured. Alternatively, learning models could be used for POS tagging. However, these models would require large databases, which are difficult to authenticate.

Tokens assigned with parts of speech are then compared with a lexicon, a dictionary containing words and their phoneme structure. For words that do not exist in the lexicon, a learning model may be used to predict which phonemes should be assigned to the tokens. This learning model might help pronounce names. The method used was an assignment of phonemes through the lexicon with no added learning models. This approach assumes that announcements do not involve words not included in the lexicon. It is also the preferred method as the lexicon is extensive and covers almost every spoken word phoneme description.

Prosody information is then assigned manually according to tested individual phoneme outputs, i.e., prosody information is pre-defined for individual phonemes. The synthesiser's tuning parameters were considered when assigning prosody information. Alternatively, prosody information could be assigned by considering not only phonemes individually but the phonemes they are grouped with. Hence, it could lead to a more natural-sounding output. However, the former method chosen could be more accurate and, for future work, should preferably be modified using the latter method for better output.

2.4. Synthesizer and synthesis strategy

The selected synthesizer was the formant synthesizer, as it is a rule-based synthesizer that depends on the formants of a signal spectrum. The synthesizer involved the design of resonators and the development of code to connect the resonators in a way that simulates the acoustic process. Additional digital components must be designed, such as amplifiers, mixers, low-pass filters, and others. The validity of the synthesizer was tested with generic digital input signals by plotting the time domain and frequency domain of the resulting signals after developing every component. Other synthesizer structures could be used for different qualities of speech output. However, the other models involve few DSP techniques and require large memory capacities in hardware and large speech corpora.

The synthesis strategy involved a form of concatenation synthesis since the formant synthesizer is only capable of synthesizing audio that uses a single phoneme. Therefore, the phonemes that are produced by the synthesizer are concatenated to produce output words. Alternatively, a word could be generated with the formant synthesizer, but that would require varying amounts of parameter inputs for individual words. As it is not a consistent solution, concatenation of phonemes is the only reliable option.

3. Design and implementation

The design and implementation of the project are presented in the following sections.

3.1. Design summary

This section summarises the project tasks and their implementation (see ***Table 1*** and ***Table 2***).

Deliverable or task	Implementation	Completion of deliverable or task, and section in the report
A formant text-to-speech (TTS) synthesizer, consisting of DSP techniques, will be designed and implemented. A prosody generation function will be implemented.	A parallel/cascade formant synthesiser was designed using several digital components designed from first principles in Python. In addition, prosody generation was implemented based on prosodic features extracted from analysed recordings.	Completed Section 3.3 of the report.
A linguistics analysis module, will be implemented to generate speech information. Natural language processing functionality will be implemented to reduce possibility of error for the linguistics analysis module.	Linguistics analysis consisted of algorithms implemented from first principles. Natural language processing encompassed generic text processing using a corpus obtained from a Python module.	Completed Section 3.2 of the report.
A text-based profanity filter will be developed and implemented.	A text-based profanity filter was developed successfully.	Completed Section 3.2 of the report.
Implementation of an email text display feature.	Emails can be displayed in a browser using the web browser library.	Completed Section 3.4 of the report.

Table 1 Design Summary

Deliverable or task	Implementation	Completion of deliverable or task, and section in the report
The intercom system must be designed and implemented.	The design of multiple intercom subsystems has been achieved. However, integration is not yet completed.	Incomplete Section 3.5 of the report.
An appropriate linguistics analysis technique has to be selected.	Linguistics analysis consists of a POS tagger using an HMM optimised with a Viterbi algorithm.	Completed Section 3.2 of the report.
Audio data from multiple people will have to be collected.	Audio data from two people has been collected using Praat software. However, only one has been integrated into the system.	Completed Section 3.3.5 of the report.
Large sentence text data sets, with profanity, have to be collected and sorted.	The text data was provided by an NLTK corpus, and profanity was self-defined based on this corpus.	Completed Section 3.2 of the report.
A physical cover for the system will have to be designed and built.	A cover was purchased off the shelf for the system to reduce the time spent on this task.	Completed Section 3.5 of the report.
The system should be integrated onto a PCB board.	No PCB designs were required for the system. Instead, the system is integrated into a single-board computer (SBC).	Completed Section 3.5 of the report.
Simulated versions of the functions should be developed for debugging and demonstration purposes.	Each subsystem was individually tested, and simulations were documented and plotted using Python.	Completed Section 3.2.3 and 3.3.7 of the report.

Table 2 Design Summary (continued)

In the following sections, the design implementation details will be described. The subsystems shown are the NLP and linguistic analysis unit, which consists of profanity filtering, and the synthesizer, which includes prosody generation. The design implementation consists of theory, analysis, modelling, simulations, software designs and implementations, and evaluation by statistical analysis.

3.2. Natural Language Processing

The design and implementation of NLP will be presented in the following sections.

3.2.1. Theory and Analysis

NLP is the computer's understanding of human language that is spoken or written. NLP aims to form a structured text from some unstructured text or, conversely, form an

unstructured text from a structured text. The former is called natural language understanding (NLU). It is used for systems that only require the computer to understand an area of the text. In contrast, the latter is called natural language generation (NLG) and is used for systems that try to generate speech with some given background knowledge, such as in chatbots. The system has to be evaluated to determine which components of NLP are required.

In TTS synthesis, the initial stage comprises of pre-processing text, which is used to eliminate any ambiguity included in the text. The main ambiguous form of any text is homonyms, homophones, and homographs. However, where TTS conversion is concerned, homographs (words spelt the same but sound different) form the central area of ambiguity. The next stage of speech synthesis is phonetization, which uses phonemes to convert text into a sequence of sounds. The final stage of speech synthesis is the generation of these sounds to form speech.

From this system evaluation, it can be seen that NLP is concerned with the first two stages of TTS synthesis, i.e., pre-processing text and phonetization. TTS also requires several intermediate steps of pre-processing and phonetization (see ***Figure 2***). Morphological, contextual, and syntactic analysis constitutes refinement of the pre-processing stage, whereas prosody generation may be seen as the refinement of the phonetization stage. The NLP module of a TTS system is now known. However, the models involved in the stages are not. The selection of models to be used are design choices that involve consideration of the type of synthesis system that will be designed (see section 3.2.2).

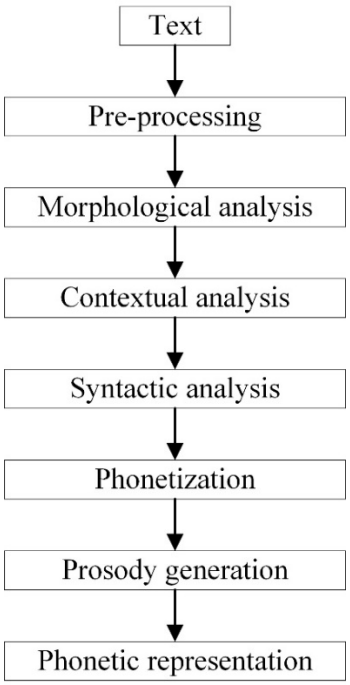


Figure 2 The Natural Language Processing Module, involving pre-processing, morphological analysis, contextual analysis, syntactic analysis, phonetization, and prosody generation.

3.2.2. Modelling

Models have to be selected based on design criteria and project requirements. Models for each of the following procedures are required for basic TTS synthesis:

- Tokenization,
- parts-of-speech tagging, and

- phonetisation.

In addition to these models, a lookup table will be used to deal with profanity after POS tagging. Additional models may be considered to assist the correctness of the synthesized output, however, requires large corpora and lexicons.

(a) Tokenization

To tokenize a sentence is to break the sentence up into individual tokens associated with a phoneme structure. For example, to better understand this, one could consider the sentence: "In 1998, Dr Wick said that his son Robert was a good student.". In this sentence, there are many words that are easy for people to understand. However, a computer does not know how to interpret the year "1998" or the abbreviation "Dr". The goal of tokenization is to convert these to computer-understood terms, i.e., "nineteen ninety-eight" and "doctor".

One way of approaching this problem is to develop a lookup table for known abbreviations and acronyms. However, this only works in some cases. For instance, the abbreviation "Dr" can be interpreted as "doctor" or "drive". It would also be unrealistic to list every number and have a lookup table to interpret each one. Numbers can also be ambiguous in specific contexts. A general approach to tokenization is thus considered.

The general approach to dealing with non-standard words (NSWs) is to detect the words, classify the words, and then expand those words. Detection is often done with hand-crafted rules, the classification may be done with basic decision trees, and expansion can then be done with lookup tables or rules. The project specification, however, states that NLP will handle non-complex sentences. This means that in most cases, the NSW scenario will not be considered, and tokenization can be performed by simply splitting up the words in the most common way, which is by spacing in English. A system should thus be considered that consists of a word and punctuation mark separator (see ***Figure 3***), which can be done with simple coding schemes.

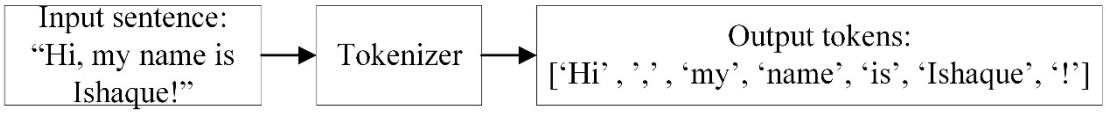


Figure 3 Tokenizer system, the input is a sentence, and the expected output is a list tokens.

(b) POS tagging

As discussed previously (see section 3.2.1), homographs are the primary source of sentence ambiguity for computer understanding. It is, therefore, important that an accurate POS tagger model be implemented. As discussed in section 2.3, the tagging method decided on was the HMM optimised with the Viterbi algorithm, as statistical approaches to handling POS tagging are proven to be the most efficient and the most accurate.

Consider a sequence of tokenised words, $w_1, w_2, \ldots, w_i$, in a sentence, and the desired output is the tag sequence of these words, $t_1, t_2, \ldots, t_i$. The system to be modelled is shown in ***Figure 4***. An HMM can be used to formulate the relationship between tokenised words and their respective POS tag because the system is assumed to be a Markov process with hidden states (tags) and observable outputs (words).

Figure 4 POS tagger system, where the input are tokens and the expected output are tagged tokens.

In Bayesian inference, all possible sequences of classes must be considered. For this problem, the sequences of classes are simply every POS that may be assigned to a word based on the available POS tags in a corpus. Of these tags, the most probable tag sequence, t_1^n, has to be decided upon, given a sequence of words, w_1^n, seen in Equation 1,

$$\hat{t}_1^n = arg\max_{t_1^n} P(t_1^n|w_1^n), \tag{1}$$

where *argmax* is a function that selects the highest probability of a tag sequence given a word sequence depicted as $P(t_1^n|w_1^n)$. The above probability expression can be solved by applying Bayes theorem to the problem (Eq. 2) and simplifying the expression (Eq. 3) as

$$\hat{t}_1^n = arg\max_{t_1^n} \frac{P(w_1^n|t_1^n)P(t_1^n)}{P(w_1^n)}, \tag{2}$$

$$\hat{t}_1^n = arg\max_{t_1^n} P(w_1^n|t_1^n)P(t_1^n). \tag{3}$$

As seen in Equation 3, the expression is still challenging to compute, so the HMM assumes that this expression is a conditional probability problem. In a conditional probability problem, two assumptions can be made about this problem. The first assumption is that the probability of a word appearing is only dependent on its own POS tag. The second assumption is that the probability of a tag appearing depends only on the previous tag. The first assumption is depicted in Eq. 4, and the second assumption is depicted in Eq. 5 as

$$P(w_1^n|t_1^n) \approx \prod_{i=1}^{n} P(w_i|t_i), and \tag{4}$$

$$P(t_1^n) \approx \prod_{i=1}^{n} P(t_i|t_{i-1}). \tag{5}$$

Equation 3 can then be re-written in terms of the assumptions made in Eq. 4 and Eq. 5 to yield the HMM for POS tagging in Eq. 6 as

$$\hat{t}_1^n = arg\max_{t_1^n} \prod_{i=1}^{n} P(w_i|t_i)P(t_i|t_{i-1}). \tag{6}$$

where $P(w_i|t_i)$, is the emission probability, i.e., the probability that a particular word contains a specific tag, and $P(t_i|t_{i-1})$ is the transition probability, i.e., the probability that a particular tag occurs in a specific sequence.

This model now provides a solution to POS tagging. However, applying this model yields two possible solutions. The first solution is the naïve approach which is an exhaustive search, in which case k^n possibilities will have to considered for some constant k. This could take a long time and quickly becomes an infeasible solution for maximising $\hat{t}_1^n$. The other approach that can be taken is a dynamic approach which can be accomplished using a Viterbi algorithm.

The Viterbi algorithm is an algorithm that calculates a Viterbi value in each time step or iteration for states. Only the state that yields the highest Viterbi value for that time step will be considered for the following time step. For example, if Eq. 6 is considered, the Viterbi algorithm would be applied by calculating all emission and transition probabilities. Instead of multiplying all of the probabilities out, the Viterbi value is determined by considering a single word iteration (say the first one, i.e., w_1). Each emission probability and transition probability relating to the combination of that single word iteration and each tag index, t_i, would be tested for that word. The highest value is the Viterbi value, and only that tag index will have to be considered in the next state as a previous index.

Using the Viterbi algorithm requires markers for start and stop transitions signifying the beginnings and ends of sentences. This way, the number of computations required significantly decreases, and the model's accuracy should not deteriorate. Also, in practicality, when multiplying lots of smaller values, underflow could occur, depending on the type of processor. The logarithm of Viterbi values could be taken to address the underflow issue. The space complexity of the Viterbi algorithm is $O(KN)$ because a $K*N$ matrix is being stored, and the time complexity is reduced to $O(K^2N)$, since only K work is done for each KN cell.

(c) Phonetization

Phonetization is the process where phonemes are assigned to words. For phonetization, the use of a lexicon is required which contains these phoneme entries. Lexicons are often structured with entries that contain:

- The head word
- The POS of the word
- The pronunciation (in terms of phonemes)
- The lexical stress

The only lexicon that is accessible is the lexicon from Carnegie Mellon University. The lexicon contains the head word, the pronunciation, and the lexical stress. Unfortunately, there is no POS available for this particular lexicon, so this will affect the system's overall accuracy. Modelling for the system consists of search algorithms to access phonetic transcription information as fast as possible. The binary search algorithm can be used as it has $O(\log n)$ time complexity. Another technique to reduce search times is to separate the lexicon into various files to store the phonemes of certain alphabets. However, in most cases, simply applying a search algorithm is efficient enough. The prosody generation function should be implemented after modelling control parameters for the synthesizer. The phonetization model produces the output, as seen in ***Figure 5***. If there is lexical stress, then it will be added as well. However, lexical stress must be more accurate and should not be relied upon.

Figure 5 Phonetization system, where the input is tagged tokens and the expected output is a phonetic representation of the tokens.

3.2.3. Software design and implementation

The complete overview of the natural language processing unit, including external entities, is shown in ***Figure 6***. The input provided to the subsystem is text, and the output provided is prosodic text, i.e., text marked with phonemes. The POS tagger requires an external corpus of sentences marked with their POS. The profanity filtering unit requires a pre-defined dictionary, and the phonetization unit requires a lexicon with words that contain phoneme structure.

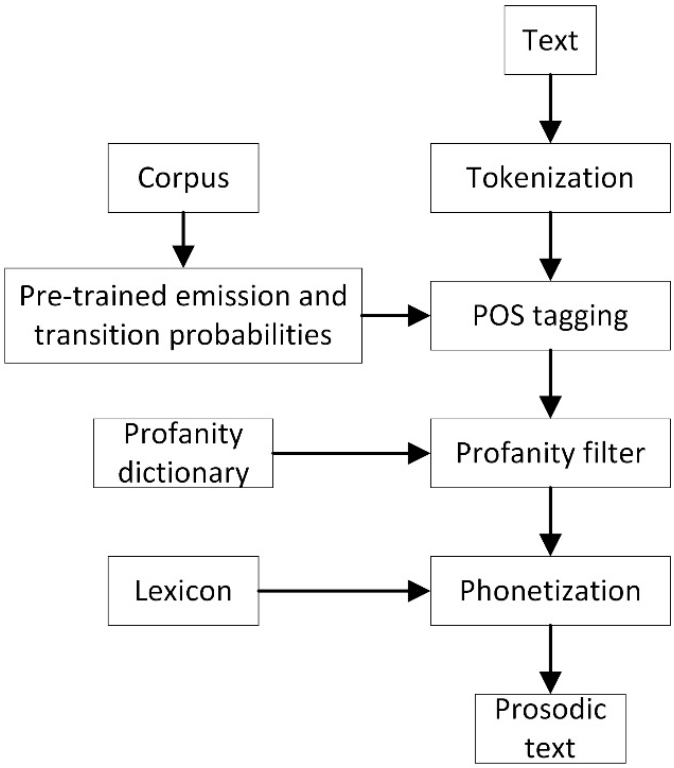


Figure 6 Natural Language Processing System Architecture, the input is text and the output is prosodic text.

The remaining software design sections contain the pseudocode of each process in the NLP unit and the outputs of the implemented system.

(a) Tokenization pseudocode

Figure 7 contains the pseudocode for the tokenization process. The tokenizer is given an input sentence received from the email inbox and tokenizes it by first performing a split where there is a whitespace character and then performing a split where there is a punctuation mark. ***Figure 8*** is an example output of token output from the tokenizer model. In an actual application, more simple sentences will be considered.

```
#Tokenization model
Input = sentence
Output = tokens

for char in sentence:
    if char == " ":
        split the word

for char in word:
    if char == punctuation mark:
        split the word
```

Figure 7 Tokenization Pseudocode.

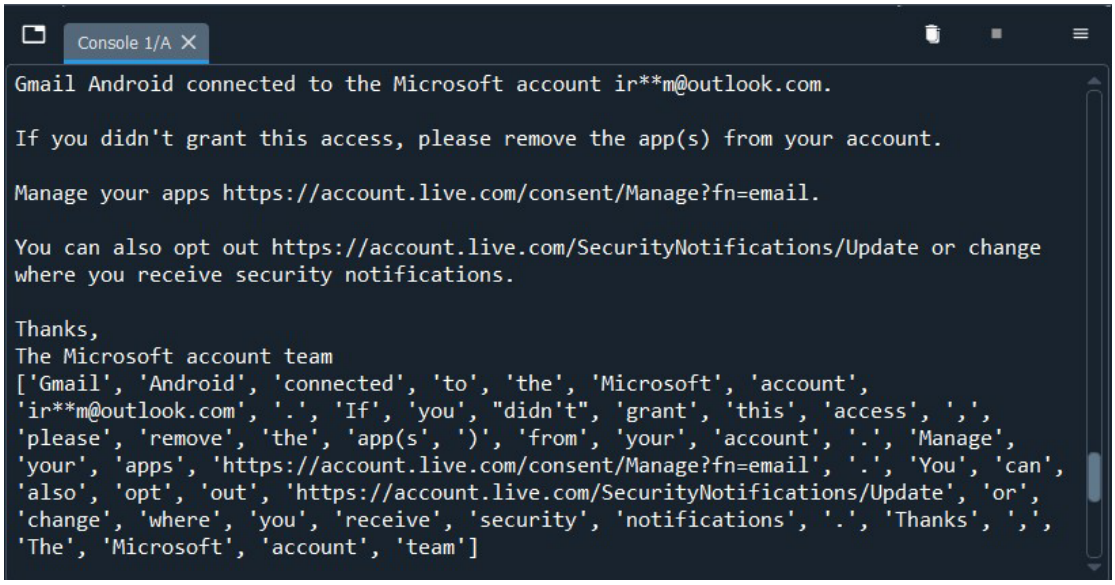


Figure 8 Tokenizer example output of a few complex sentences.

(b) POS tagger pseudocode

Figure 9 contains the pseudocode for the POS tagging process. Pseudocode was also included for the Viterbi algorithm optimization. It is worth noting that only the transition matrix can be pre-determined using the available corpus. This data may then be stored and accessed when performing POS tagging. The emission probabilities can also be stored if only words from the corpus are used. However, if not, it will have to be calculated during the Viterbi algorithm run time.

Figure 10 contains an example output for the sentence: “John likes the blue house at the end of the street.” The system's output may be verified by manually evaluating the sentence or alternatively using a reliable POS tagging source. This sentence contains non-complex language, so it is easier for non-professionals to evaluate it. However, the output does provide the correct tagging prediction.

```
#POS tagger
Input = tokens
Output = POS of tokens

#First download corpus if not downloaded
train data = get sentences from corpus bank
tags = get number of unique tags in train words
words = get number of unique words in train words

t = number of tags
#Calculate the transition probabilities
transition matrix = create (t x t) matrix

#For unknown tags rules may be specified
rules = #create rules list, e.g., if ending in 'ed', POS = 'VERB'

#Calculate the most probable tags
predicted tags = Viterbi (tokens, train data, transition matrix)

#Viterbi function
function Viterbi:
 for word in words:
    v = #probability list for current word (Viterbi values)
    for tag in train data:
        #Make special case for first tag
        current transition = transition matrix [last state, tag]

        emission = #compute emission probability
        state = #compute state probability
        v list = #append state to list

    #Get max Viterbi value from list
    value = max(v)

    #If the max v value is a probability of 0, then apply the rule set
    if value = 0:
        POS = #accroding to rule set
    #Otherwise just assign the POS relating to the highest state
    else:
        POS = #highest state probability

    POS list = #append to a POS list

 return POS list
```

Figure 9 POS tagger Pseudocode

Figure 10 POS Tagger example output

(c) Profanity filter pseudocode

Figure 11 contains the pseudocode for the profanity filter loop. It simply checks whether or not an explicitly defined profane word is in a pre-defined dictionary, and if it is, then it is marked with a profanity tag. The synthesiser should handle profanity by replacing the profane word with a short duration of silence.

```
#Profanity filtering
Input = tokens
Output = filtered tokens

for token in tokens:
    if token in profanity dictionary:
        mark token as profanity
```

Figure 11 Profanity filtering Pseudocode

(d) Phonetization pseudocode

Figure 13 contains the pseudocode for the phonetic transcription of the system. Separate code can transfer the lexicon text file to a file optimized with Python, i.e., a .npy file. The format of the lexicon can be used to determine which phonetic information to attach to certain words. A search algorithm could also be used to speed up search time. ***Figure 14*** contains an example output of what to expect in terms of prosodic text for the words 'composing', 'perpetuates, and 'underscoring'.

```
#Phonetic transcription
Input: POS tagged words
Output: Phonetic representation and lexical stress (optional)
#Lexicon text file is initially written to .npy file
lexicon = load('file')
#First index of every element in lexicon can be compared with the token

#Return all instances of the word
for element in lexicon:
    possible words = #search lexicon index 0 and append if it matches token

for word in possible words:
    #Check the second index of word to match POS
    if POS matches word POS:
        phonemes = remaining indices of word
        stress = third index of word

return phonemes, stress
```

Figure 12 Phonetic transcription Pseudocode

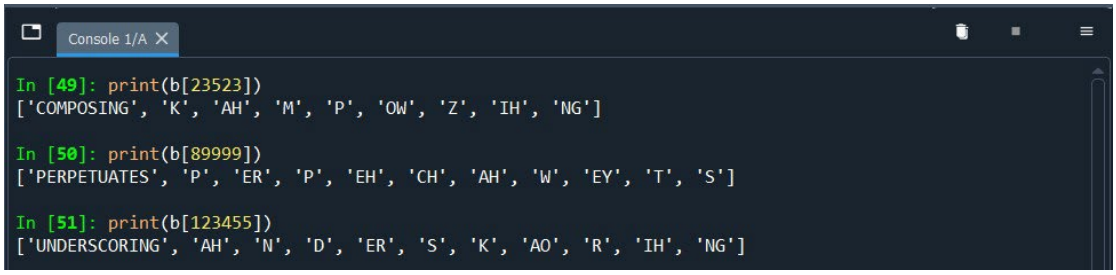


Figure 13 Phonetic transcription example output

3.3. Speech synthesis

The design and implementation of a speech synthesis subsystem are discussed in the following sections.

3.3.1. Theoretical analysis and System modelling

The theoretical analysis and system modelling for a speech synthesizer are considered.

(a) Formant synthesis

As discussed in section 1.1.3, speech synthesis systems are split into two styles. The first is rule-based synthesis, and the other is data-driven synthesis. Rule-based synthesis methods are preferable for reasons surrounding hardware, data access, and time limitations. Of the rule-based techniques, the formant synthesizer was selected, as it offers fewer computationally intensive operations, and the modelling of the system is less complex and yields a more accurate synthesized speech. The concept of formants has been briefly touched on in section 1.1.3(b), and the concept will be further explored during modelling and analysis.

The design of the synthesizer is derived from the formant synthesizer developed by Dennis Klatt in 1979 [5]. Klatt bases the design on an acoustic theory of speech production presented by Fant [6], and the theory is summarized in ***Figure 15***. According to this theory, one or many sound sources are activated by the build-up of lung pressure. Each sound source may be characterized separately in the frequency domain by a source spectrum *S(f)*, where *f* is the frequency in Hz. The sound source then excites a vocal tract transfer function *T(f)*, which is the ratio of lip volume velocity *U(f)*, to source volume velocity *S(f)*. A radiation characteristic *R(f)* then describes the output sound pressure, *P(f)*, recorded at some distance away from the lips of the talker.

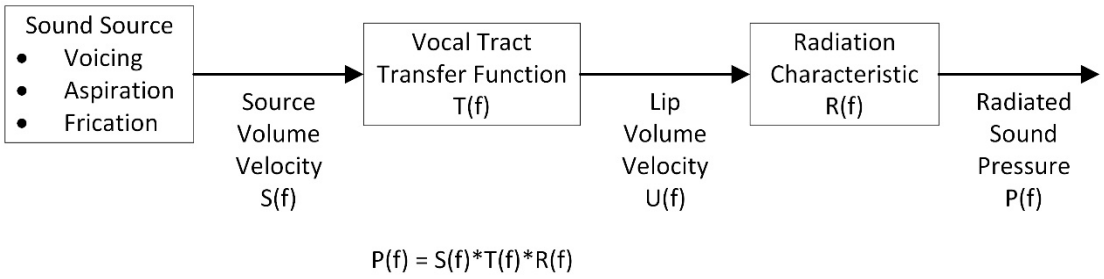


Figure 14 The output spectrum of a speech sound, P(f), can be represented as a product of a source spectrum S(f), a vocal tract transfer function, T(f), and a radiation characteristic, R(f).

The synthesizer includes multiple components to simulate the generation of sound sources, the vocal tract transfer function, and the radiation characteristic. This process will be discussed in the sections to follow.

(b) Cascade and Parallel configurations

To simulate the vocal tract transfer functions, researchers have explored multiple configurations to obtain a high-quality approximation to human speech. Of the numerous configurations, two standard configurations are parallel and cascade.

In the parallel configuration, formant resonators that stimulate the vocal tract transfer function are connected in parallel, as seen in ***Figure 16***. As seen in the figure, all resonators are preceded by an amplitude control which determines the amplitude of the

relevant formant peak in the output spectrum. This configuration may apply to both voiced and voiceless speech sounds.

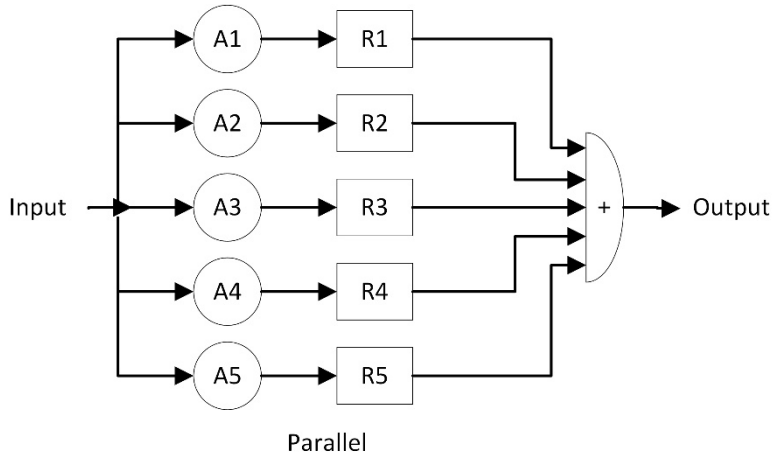


Figure 15 The vocal tract transfer function simulated in a parallel configuration. A set of digital resonators, labelled R, are each preceded by an amplitude control, labelled A.

In the cascade configuration, formant resonators are connected in cascade, as seen in ***Figure 17***, to synthesize sonorant sounds. The advantage of the cascade connection is that amplitudes for formant peaks are automatically adjusted, reducing the need for individual amplitude control of each formant. However, this only applies to sonorants, so the disadvantage is that it still requires a parallel configuration to handle other sounds that are made above the larynx, located between the throat and lungs. Although the parallel synthesizer is not optimal, in that it requires amplitude control, it is still necessary for sounds that the cascade configuration will find difficult to produce.

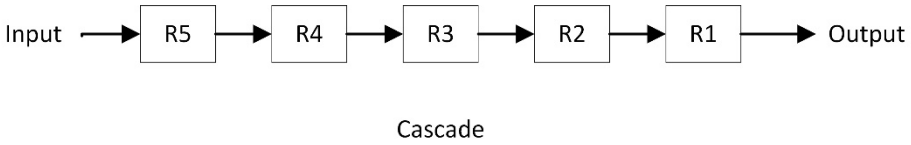


Figure 16 The vocal tract transfer function simulated in a cascade configuration. A set of digital resonators, labelled R, are connected in cascade.

As discussed, both configurations must be utilised in a hybrid connection to simulate the vocal tract transfer function. This cascade and parallel combination is shown in ***Figure 18***. Alternatively, a single parallel configuration can be used if amplitude control over all formants is required. A digital switch may be implemented to set up the configuration beforehand. For this project implementation, the combination configuration is used, considering the outlined benefits of both.

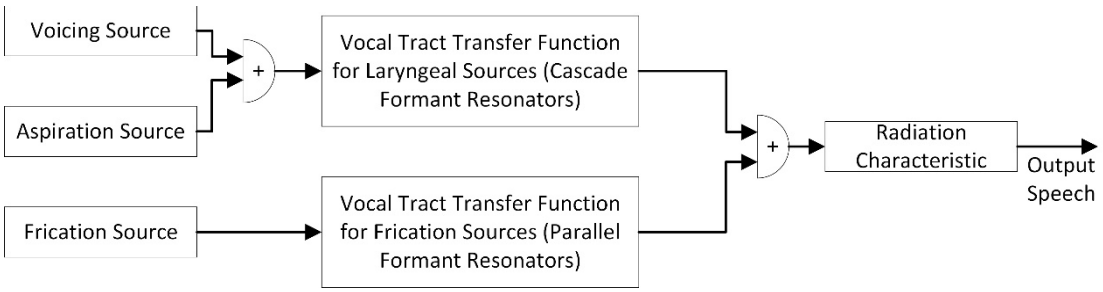


Figure 17 The synthesizer may be used in a Cascade/Parallel Formant configuration, if amplitude control over sonorant sounds is not desired.

(c) Waveform sample rate

The majority of sound energy in speech is contained within the frequency ranges between 80 Hz and 8000 Hz [11], although these values are debatable. However, intelligibility tests of band-pass filtered speech indicate that intelligibility is not measurably changed if frequencies above 5000 Hz is removed [12]. It will still sound natural if an external low-pass filter is used to filter speech above 5000 Hz.

Due to software improvements, audio scaling, and very high sample rates accessible with generally low computing power, reducing the sample rate for external audio sources is not required. These insights, however, provide information about the expected sampling rate of the system (5000 Hz) if external low-pass filtering is considered. Furthermore, due to the maximum sample rate for recognising sound energy (8000 Hz), a sample rate of 10000 samples per second should be reasonable for modern micro-controller processors.

(d) Digital resonators

Digital resonators are the building blocks of the synthesizer. The purpose of the digital resonator is to simulate the resonant (formant) frequency, F, and the resonance bandwidth, BW, of the individual formants. The model of the digital resonator is based on an infinite impulse response (IIR) filter configuration scheme. The resonator configuration is provided in ***Figure 19***. Samples of the output sequence *y(nT)* are computed from the input sequence *x(nT)* in Eq. 7, as

$$y(nT) = Ax(nT) + By(nT - T) + Cy(nT - 2T), \tag{7}$$

where *y(nT-T)* and *y(nT-2T)* are the previous two samples of the output sequence *y(nT)*, affected by a unit delay. The variable T is the sample period of the signal, which is calculated as the inverse of the sample rate. T for all cases will be 0.0001 seconds. The constants A, B, and C are related to the resonant frequency and resonance bandwidth by the impulse invariant transformation shown in Eq. 8, as

$$\begin{aligned} C &= -\exp(-2\pi\, BW\, T), \\ B &= 2\exp(-\pi\, BW\, T)\cos(2\,\pi\, FT), \\ A &= 1 - B - C, \end{aligned} \tag{8}$$

where F, BW, and T are the same variables as previously defined.

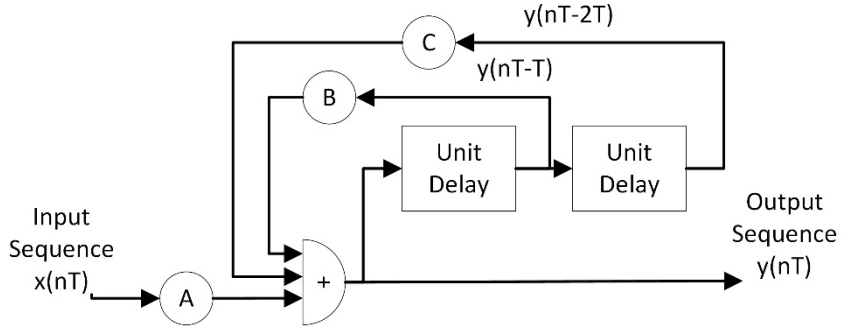


Figure 18 The digital resonator block diagram. It translates an input sequence x(nT) to an output sequence y(nT), through a combination of Unit delays (buffers), and coefficients (A, B, and C).

The digital resonator is modelled as a second-order difference equation (Eq. 7). And the transfer function for the sampled frequency response is given by Eq. 9 as

$$T(f) = \frac{A}{1 - Bz^{1} - Cz^{-2}}, \tag{9}$$

where $z = \exp(2\pi jfT)$. This transfer function can be obtained by performing a matched z-transform using the digital resonator block diagram. The transfer function will be simulated to verify the validity of the digital resonator.

(e) Digital anti-resonator and low-pass resonator

The digital anti-resonator simulates antiresonance or an anti-formant. The frequency response of the anti-resonator is thus expected to yield a frequency response that is the exact mirror image of the digital resonator's response. Antiresonance is important for shaping the voicing source's spectrum and simulating nasalization effects in the cascade model of the vocal tract transfer function. Antiresonance can be accomplished by modifying the equations of the digital resonator, such that the samples of the output sequence *y(nT)* are related to the input sequence *x(nT)* by Eq. 10 as

$$y(nT) = A'x(nT) + B'x(nT - T) + C'x(nT - 2T), \tag{10}$$

where *x(nT-T)* and *x(nT-2T)* are the previous two samples of the input *x(nT)*. The new constants *A'*, *B'*, and *C'* are defined in Eq. 11 as

$$\begin{aligned} A' &= 1/A, \\ B' &= -B/A, \\ C' &= -CA, \end{aligned} \tag{11}$$

where A, B, and C are obtained with Eq. 8, where F, and BW, are the antiresonance frequency and bandwidth, respectively.

A digital low-pass filter can be produced as a special case when setting the frequency *F* of a digital resonator to zero. The cut-off frequency of a low-pass resonator designed in this way should be equal to half of the specified resonator bandwidth, i.e., $F_c = BW/2$. Low-pass filters are required for voicing sources and will be described when discussing the control parameters of the synthesizer. The simulation of this special case of low-pass filter will be provided.

(f) Formant Synthesizer block diagram and control parameters

Figure 20 is the block diagram of the cascade and parallel formant synthesizer introduced by Klatt [5]. The project utilises this model as a basis for synthesising vowel and consonant sounds. The model was modified to reduce the number of control parameters required to determine the characteristics of the output signals. As a result, there are 20 available control parameters of 37 that may be altered for the desired outcome. These parameters are found in **Table *5*** and are marked as a variable (V), or constant (C). The table also contains each control parameter's abbreviated forms, which will be referred to when discussing those parameters and how to model them. Furthermore, the synthesizer includes ranges for parameters and their typical values.

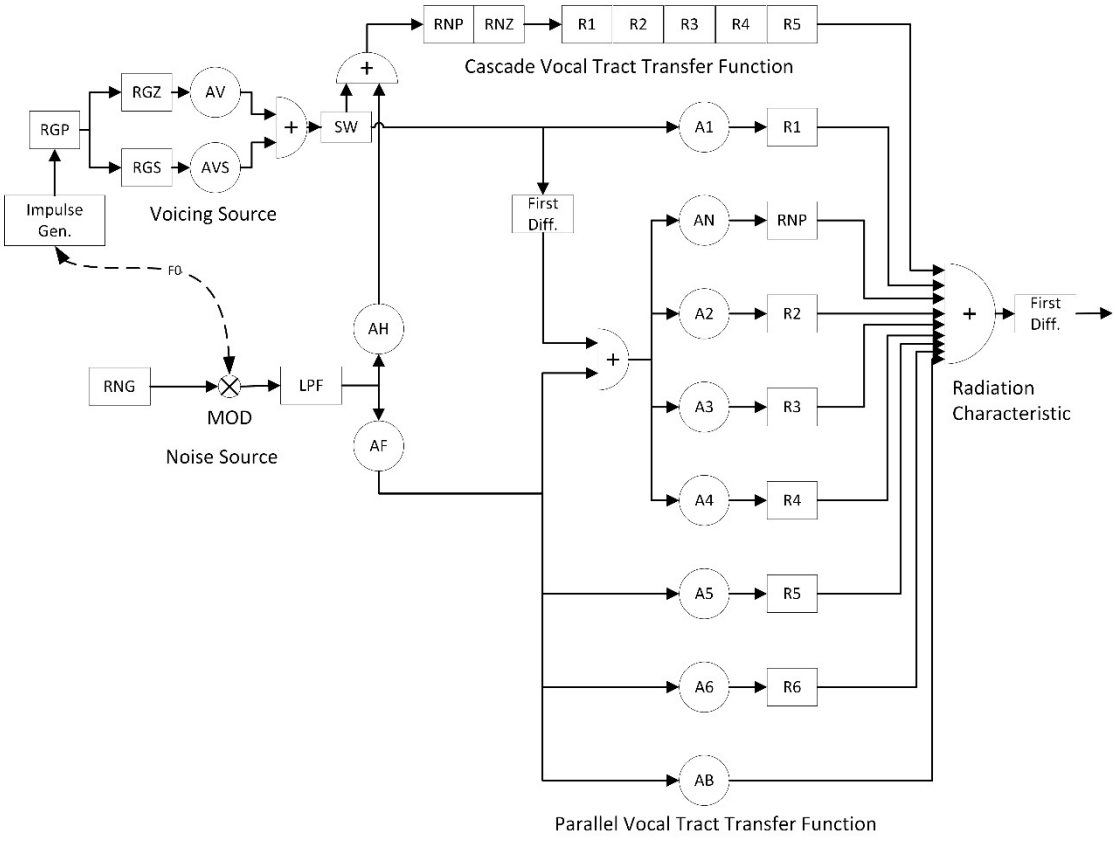


Figure 19 Block diagram of cascade and parallel formant synthesizer. R indicates a digital resonator, and A indicates an amplification control. The sound sources, vocal tract transfer functions and radiation characteristics are all labelled on the diagram.

N	V/C	Sym.	Name	Min	Max	Typ.
1	V	AV	Amp of voicing (dB)	0	80	0
2	V	AF	Amp of frication (dB)	0	80	0
3	V	AH	Amp of aspiration (dB)	0	80	0
4	V	AVS	Amp of sinusoidal voicing (dB)	0	80	0
5	V	F0	Fundamental freq. of voicing (Hz)	0	500	0
6	V	F1	Formant 1 freq. (Hz)	150	900	450
7	V	F2	Formant 2 freq. (Hz)	500	2500	1450
8	V	F3	Formant 3 freq. (Hz)	1300	3500	2450
9	V	F4	Formant 4 freq. (Hz)	2500	4500	3300
10	V	FNZ	Nasal zero freq. (Hz)	200	700	250
11	C	AN	Nasal formant amp (dB)	0	80	0
12	C	A1	Formant 1 amp (dB)	0	80	0
13	V	A2	Formant 2 amp (dB)	0	80	0
14	V	A3	Formant 3 amp (dB)	0	80	0
15	V	A4	Formant 4 amp (dB)	0	80	0
16	V	A5	Formant 5 amp (dB)	0	80	0
17	V	A6	Formant 6 amp (dB)	0	80	0
18	V	AB	Bypass path amp (dB)	0	80	0
19	V	B1	Formant 1 bw (Hz)	40	500	50
20	V	B2	Formant 2 bw (Hz)	40	500	70
21	V	B3	Formant 3 bw (Hz)	40	500	110
22	C	SW	Switch, 0=CASC, 1=PARA	0	1	0
23	C	FGP	Glottal resonator 1 freq. (Hz)	0	600	0
24	C	BGP	Glottal resonator 1 bw (Hz)	100	2000	100
25	C	FGZ	Glottal zero freq. (Hz)	0	5000	1500
26	C	BGZ	Glottal zero bw (Hz)	100	9000	6000
27	C	B4	Formant 4 bw (Hz)	100	500	250
28	V	F5	Formant 5 freq. (Hz)	3500	4900	3750
29	C	B5	Formant 5 bw (Hz)	150	700	200
30	C	F6	Formant 6 freq. (Hz)	4000	4999	4900
31	C	B6	Formant 6 bw (Hz)	200	2000	1000
32	C	FNP	Nasal pole freq. (Hz)	200	500	250
33	C	BNP	Nasal pole bw (Hz)	50	500	100
34	C	BNZ	Nasal zero freq. (Hz)	50	500	100
35	C	BGS	Glottal resonator 2 bw (Hz)	100	1000	200
36	C	SR	Sampling rate	5000	20000	10000
37	C	NFC	Number of cascaded formants	4	6	5

Table 3 A list of control parameters for the cascade and parallel synthesizer. For each parameter, V/C specifies whether that parameter is variable or constant. The specifier and name of each parameter is included along with their ranges, and typical values.

3.3.2. Sources of Sound

Two primary sources of sound occur during speech production [13]. The first source involves quasi-periodic vibrations of the vocal folds. The vibration of the vocal folds is called voicing and is the most relevant source of sound for the English language.

The second source involves a generation of turbulent noise by the rapid flow of air past a narrow constriction. This noise source is called aspiration if that constriction is located at the vocal folds. Aspiration is used to produce sounds like the 'h' sound. If the noise source is located above the larynx, for producing sounds like 's', then the noise source is called frication. Plosive sounds also mainly consist of frication.

When voicing and turbulent noise are both evident in a sound, usually for fricatives ('z' or voiced 'h'), the noise seems to be presented in a way where it is amplitude modulated by the vocal folds. For this reason, the synthesizer should be capable of generating two voicing waveforms (normal voicing waveforms and quasi-sinusoidal waveforms) and two aspirations (normal aspiration and amplitude modulated aspiration). These are the only required sound sources for English.

(a) Voicing source - the Impulse train generator

The voicing source can be seen in the top left of ***Figure 20***. The control parameters that are variable for this source are the fundamental frequency of voicing (*F0*), the amplitude of normal voicing (*AV*), and the amplitude of quasi-sinusoidal voicing (*AVS*).

An impulse train generator activates the voicing source when a fundamental frequency value above 0 Hz is specified. *F0* can be extracted by analysing Praat spectrograms and waveforms of the individual's voice. *AV* determines the amplitude of each pulse of the pulse train. Typical ranges of *AV* is from 0 dB when off, to an average of 60 dB, in a strong vowel. Furthermore, the number of samples between pulses (*T0)* is determined by Eq. 12 as

$$T0 = SR/F0, \tag{12}$$

where *SR* is a sampling rate previously defined as 10 000, and *F0* is the fundamental frequency of the person's voice. The functionality of the impulse train generator will be simulated.

(b) Normal voicing

As discussed in section 3.3.1(e), low-pass resonator filters are required for voicing sources, which are *RGP* and *RGS*. The resonator concerned with normal voicing is *RGP*. The output of the impulse train generator has to be sent through *RGP* to produce a smooth waveform representing a typical glottal volume velocity waveform [5]. Therefore, the frequency and bandwidth (*FGP* and *BGP*) are constants set to 0 Hz and 100 Hz, respectively, to produce a -12 dB per octave above 50 Hz, thus accomplishing this requirement.

The anti-resonator *RGZ* may be used to modify the shape of the spectrum of the voicing source for certain people even further. The frequency and bandwidth (*FGZ* and *BGZ*) are set as constants but, if so desired, may be modified for a particular person if the normal voicing output is not adequate.

(c) Quasi-sinusoidal voicing

The control parameter *AVS* determines the amount of smoothed voicing generated during voiced fricatives, voiced aspirates, and voiced plosives. Once again, the impulse train generator has to be sent through the second low-pass filter, but this time for quasi-sinusoidal voicing. *RGS* is used. Its frequency is set to zero, and its bandwidth (*BGS*) determines the cut-off frequency for signals to be attenuated. A bandwidth of 200 Hz is adequate for this implementation. The exact ranges apply to *AVS* as it applies to *AV*.

(d) Frication source and the Low pass filter

The frication source is located at the bottom left of ***Figure 20***. It consists of a random number generator, a modulator, a -6 dB/octave low-pass filter, and two variable control parameters. The control parameters are the amplitude of frication (*AF*) and aspiration (*AH*). The spectrum of the frication source should be flat, and the amplitude distribution should be Gaussian [5]. A random number generator can accomplish this.

An assumption was also made that the output (source volume velocity) is proportional to the integral of source pressure [5] and that it can be approximated with the low-pass filter. The output samples of the filter, *y(nT),* is related to the input, *x(nT)*, by Eq. 13, as

$$y(nT) = x(nT) + y(nT - T). \tag{13}$$

The output of the random number generator is amplitude modulated by the *MOD* component when *F0* and *AV* are specified. AF determines the amplitude of frication, and a value of 60 dB will generate a strong frication noise, while a zero value turns off frication.

(e) Aspiration source

Aspiration noise is the same as frication noise. However, it generates noise from the larynx. A separate control is needed for aspiration amplitude as the cascade tract is designed for laryngeal sound sources. The control parameter *AH* is used to control the amplitude of aspiration. Like frication, a value of 60 dB will generate strong aspiration, while a zero value will turn off aspiration.

3.3.3. Vocal Tract Transfer functions

As described in section 3.3.1(a), the vocal tract transfer function is a ratio of lip volume velocity to source volume velocity. The synthesizer uses two configurations to model this transfer function. The cascade configuration models sound within the larynx, and the parallel configuration models sound properties of frication noise (see section 3.3.1(b)). The following sections will discuss the parameters contained within these two configurations.

(a) Cascade vocal tract model

Five resonators are used to simulate the vocal tract of a male speaker. This is due to their average vocal tract length being 17 cm long. A longer vocal tract relates to the number of formants in the speech waveforms produced by people. Females generally have vocal tracts that are 15 – 20% shorter than a male vocal tract, suggesting that having a cascade model with four resonators for a female is suitable. For males with longer vocal tracts, it is recommended to increase the number of formant resonators to 6.

The cascade model of ***Figure 20***, consisting of five formant resonators, has a transfer function represented in the frequency domain as just the product of resonator transfer functions, shown in Eq. 14 as

$$T(f) = \prod_{n=1}^{5} \frac{A(n)}{1 - B(n)z^{1} - C(n)z^{-2}}, \tag{14}$$

where constants *A(n)*, *B(n)*, and *C(n)* is determined by the *n*th formant frequency *F(n)* and *n*th formant bandwidth *BW(n)*, described by the relationships previously derived in Eq. 8.

(b) Formant frequencies and bandwidths

Each formant resonator introduces a peak in the magnitude spectra of a waveform. This will be evident when observing digital resonator simulations. The *n*th formant peak is specified with the variable control parameter relating to *Fn*. The bandwidths of the formants are similarly specified with the variable control parameter relating to *BWn*.

The shape of the vocal tract theoretically determines formant frequency values. Different formant frequency values produce different sounds of speech, which are caused by the different shapes of the vocal tract. The first three formants provide the most impactful formant frequencies that affect human perception in terms of articulation. The fourth and fifth formants do not vary often and could be held constant to a certain extent. These last formants assist with obtaining the shape of the waveform but provide little impact on sound perception.

Formant bandwidths are a function of energy loss in sound. These bandwidths are often difficult to deduce from spectrograms where natural speech is concerned due to multiple irregularities in the glottal waveform. There are techniques to determine formant bandwidths. However, formant bandwidth collection and frequency can be handled by evaluating spectrograms using Praat software. Similar to some formant frequencies, formant bandwidths often yield minor variations, so they could be held constant to an extent.

(c) Nasals and nasalization of vowels

It is not possible to approximate nasal sounds in the cascade structure with only the five formant resonators. More than five formants are often present in the nasals, and formant amplitudes do not relate to the output of nasal sounds in the cascade structure. Nasalization introduces poles and zeros into the cascade structure by introducing a side-branch resonator. This resonator pair (*RNP* and *RNZ*) represents the oral cavity in case of murmurs, and it is used to approximate these nasal murmurs. The nasal pole frequency (*FNP*) can be set to 270 Hz for all times, while the nasal zero frequency (*FNZ*) should be set to 270 Hz for all times that no nasal sounds occur. In this way, the side branch is removed for non-nasalized speech sounds since they would be equal to each other and cancel out.

(d) Parallel vocal tract model for frication sources

The parallel vocal tract model is used mainly during fricative excitation. Formant amplitudes are set to provide excitation for selected formants. There are six formant resonators in the parallel branch for the synthesis of high-frequency noise (s and z sounds). Adding a sixth formant resonator is better for simulating high-frequency noise than to increasing the value of the fifth formant frequency.

In addition to amplitude control and control over fricatives, a bypass path is included in the parallel configuration (*AB*). This variable control parameter is present because the transfer functions for some sounds, like f and v, contain no prominent resonant peaks. This control parameter is enabled to bypass the resonators so that the summation produces a flat transfer function.

3.3.4. Radiation Characteristic

The radiation characteristic on the right side of ***Figure 20*** models the effect of directivity patterns, which is the degree to which sound radiation is emitted in a single direction from the head. The sound pressure measured approximately a meter away from the lips is proportional to the temporal derivative of the lip-plus-nose volume velocity and inversely proportional to the distance from the lips [6]. The transformation can be simulated by taking the first difference in lip-nose volume velocity, as shown in Eq. 15 as

$$p(nT) = u(nT) - u(nT - T). \tag{15}$$

A simulation of the first difference component showing the radiation characteristic will be shown.

3.3.5. Synthesis strategy and Prosody generation

The synthesis strategy involves data collection using Praat, developing a prosody generation function, synthesising vowels and consonants, scaling the sounds, and concatenating them. This procedure will be discussed in the sections to follow.

(a) Audio data collection and feature extraction

Praat software offers a variety of tools that are used to record data for a person and analyse the waveforms so that features can be extracted. The features of interest are the amplitude of voicing (*AV*), the fundamental frequency (*F0*), the formant frequencies (*Fn*), the formant bandwidths (*BWn*), and the shape of the spectral waveform. ***Figure 21*** shows a waveform and spectrogram of the utterance 'odd'. The dark patches in the spectrogram indicate the energy of the speech output, and it can be used to determine the formant frequencies. The bandwidths are obtained by evaluating the range (vertically) of the energies.

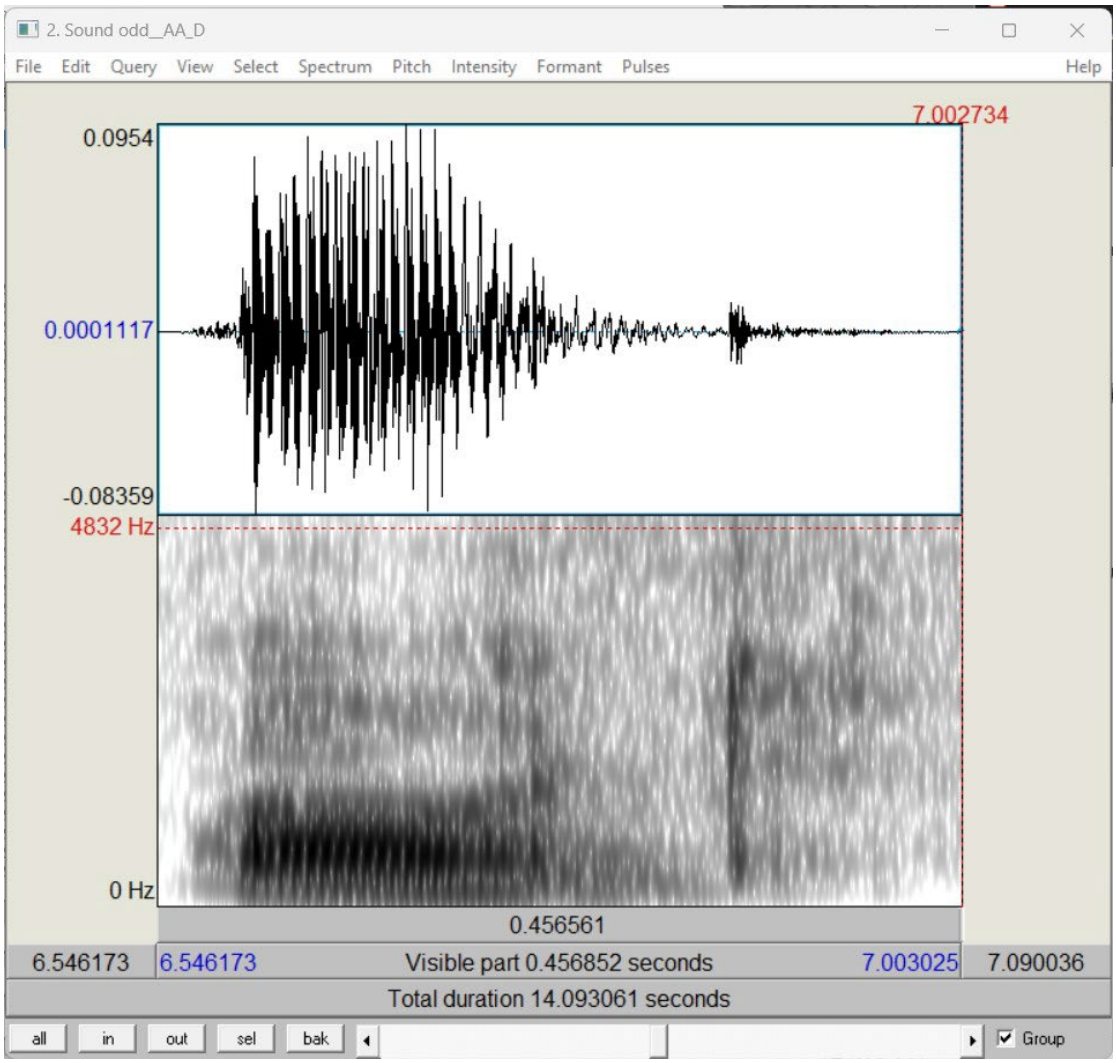


Figure 20 Praat window showing the waveform and spectrogram of the word utterance 'ODD'. This utterance is used to extract features for the 'AA' phoneme.

Furthermore, the software allows easier retrieval of these parameters, as seen in ***Figure 22***. In the figure, the pitch (blue), intensity (yellow), and formant frequencies (red) are approximated for easier readability. At the point of the cursor, the first formant frequency is approximately 681.2 Hz, the intensity or fundamental frequency is 132.9 Hz, and the intensity or amplitude of voicing is 63 dB. Therefore, for finding the value of the second formant the following red line above the first one may be used as a guideline. Bandwidths of formants are obtained by selecting a region and running a script that determines the spread of the formant samples for that region. This is a feature of the software and could be used for efficiency. Standard bandwidth calculations entail selecting the high-frequency margin and subtracting the lower-end frequency margin. However, this method would not be efficient and is not reliable for analysing spectrograms.

For data collection involving consonant sounds, formant amplitudes are required. Unfortunately, Praat software does not have any feature to extract amplitude information. However, as a general rule for the formant synthesizer, consonants are formed with the same approximate frequencies and amplitudes for all people. The data used for consonant sounds are thus taken from Klatt [5], which results from trial-and-error adjustments.

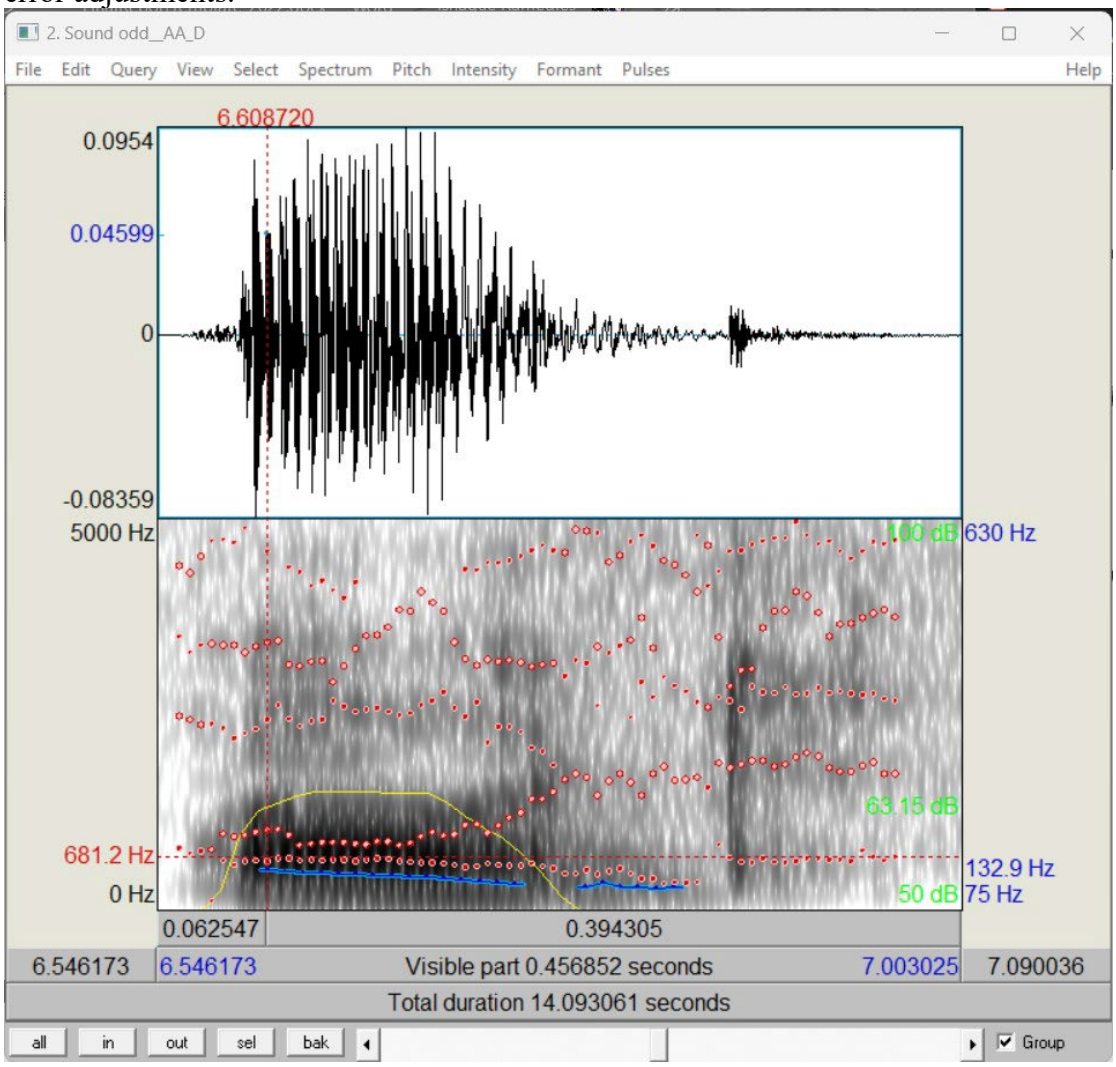


Figure 21 Praat window for the utterance 'odd'. The spectrogram contains extra information pertaining to formant frequencies, bandwidths, fundamental frequency, and amplitude of voicing.

(b) Synthesis of vowels

Vowels can be synthesized by using the data collection strategy for multiple sounds. The critical control parameters to be varied are *AV*, *F0*, and the three lowest formant frequencies and bandwidths. In addition, *AH* and *AVS* can be activated for natural breathy sounds. The result of collecting data for all vowel phonemes present in the lexicon used for the TTS synthesizer is shown in ***Table 6***. The first row of each result indicates the start value, and the second row of each result indicates the end value. Similarly, other parameters are captured as a result of a start and end value which is evenly divided.

Phoneme	F1	F2	F3	B1	B2	B3
'AA'	625	920	2499	130	248	451
	610	1100	2666			
'AE'	770	1861	2513	138	144	78
	640	1788	2691			
'AH'	718	1234	2488	146	265	248
	644	1308	2636			
'AO'	460	644	2710	149	30	661
	490	773	2710			
'AW'	773	1200	2150	219	204	256
	590	1013	2710			
'AY'	737	866	2322	121	435	435
	460	1900	2600			
'EH'	550	2673	2986	48	428	271
	387	1972	2673			
'ER'	552	1474	2654	66	67	84
	350	1548	2654			
'EY'	600	2000	2670	121	556	599
	350	2270	3000			
'IH'	440	2110	2750	26	377	346
	300	2220	2750			
'IY'	310	2020	2960	45	200	400
	290	2070	2960			
'OW'	660	1220	2575	120	66	93
	450	1330	2750			
'OY'	535	900	2513	105	105	260
	410	1900	2513			
'UH'	450	1200	2200	65	110	140
	375	850	2200			
'UW'	420	1680	3185	52	133	547
	310	1270	3185			

Table 4 Parameter values for the synthesis of vowel phonemes present in the lexicon. The first row of each result indicates the start value, and the second row of each result indicates the end value.

(c) Synthesis of consonants

For the synthesis of consonants, data collection suggests using the values obtained in ***Table 7***, ***Table 8***, ***Table 9***, ***Table 10***, and ***Table 11***. Sonorants are similar to vowels and can be synthesized using the same methodology as vowels. The 'r' and 'l' sounds, however, depend on the following vowel in an utterance to an extent. *AV* is set to 10 dB lower than the proceeding vowel for this case. The 'h' sonorant is synthesized using the frequency and bandwidths of the following vowel, increasing *F1* to 300 Hz, and replacing voicing with aspiration.

Values in the fricative and affricate tables are approximated for consonants preceding vowels. Therefore, *A2* is 0 in all these cases and should be set to 60 dB if this is not true. For plosives, the sound is generally short and should be shorter than other consonant sounds such that it matches up with a plosive burst sound.

Sonor	F1	F2	F3	B1	B2	B3
w	290	610	2150	50	80	60
y	260	2070	3020	40	250	500
r	310	1060	1380	70	100	120
l	310	1050	2880	50	100	280

Table 5 Parameter values for sonor consonants.

Fric.	F1	F2	F3	B1	B2	B3	A2	A3	A4	A5	A6	AB
f	340	1100	2080	200	120	150	0	0	0	0	0	57
v	220	1100	2080	60	90	120	0	0	0	0	0	57
θ	320	1290	2540	200	90	200	0	0	0	0	28	48
δ	270	1290	2540	60	80	170	0	0	0	0	28	48
s	320	1390	2530	200	80	200	0	0	0	0	52	0
z	240	1390	2530	70	60	180	0	0	0	0	52	0
š	300	1840	2750	200	100	300	0	57	48	48	46	0

Table 6 Parameter values for fricative consonants.

Aff.	F1	F2	F3	B1	B2	B3	A2	A3	A4	A5	A6	AB
č	350	1800	2820	200	90	300	0	44	60	53	53	0
ĵ	260	1800	2820	60	80	270	0	44	60	53	53	0

Table 7 Parameter values for affricate consonants.

Plo.	F1	F2	F3	B1	B2	B3	A2	A3	A4	A5	A6	AB
p	400	1100	2150	300	150	220	0	0	0	0	0	63
b	200	1100	2150	60	110	130	0	0	0	0	0	63
t	400	1600	2150	300	120	250	0	30	45	57	63	0
d	200	1600	2600	60	100	170	0	47	60	62	60	0
k	300	1990	2850	250	160	330	0	53	43	45	45	0
g	200	1990	2850	60	150	280	0	53	43	45	45	0

Table 8 Parameter values for plosive consonants.

Nasal	FNP	FNZ	F1	F2	F3	B1	B2	B3
m	270	450	480	1270	2130	40	200	200
n	270	450	480	1340	2470	40	300	300

Table 9 Parameter values for nasal consonants.

(d) Synthesis of utterance

A lexicon is used to extract phonemes to synthesize words (see section 3.2.2(c)). The synthesis of each phoneme of that word is conducted sequentially and concatenated in that order. This approach is somewhat naïve because it does not consider sections of silence between phonemes. A prosody generation function comprised of a loop to synthesise each phoneme according to its position in the word is then employed. For instance, if the phoneme is located at index 0 (the start of a word), it would generate synthesized output according to that position's prosody information.

3.3.6. Software design implementation

(a) Classes

Figure 23 contains the UML class diagram of the implemented synthesizer. A separate function is used to initialise a synthesizer object with a specifications object (the control parameters). In addition, the synthesizer object has a construct function which initialises section objects, such as the cascade tract. Finally, each section object initialises some feature objects (such as resonators) according to that section's requirements.

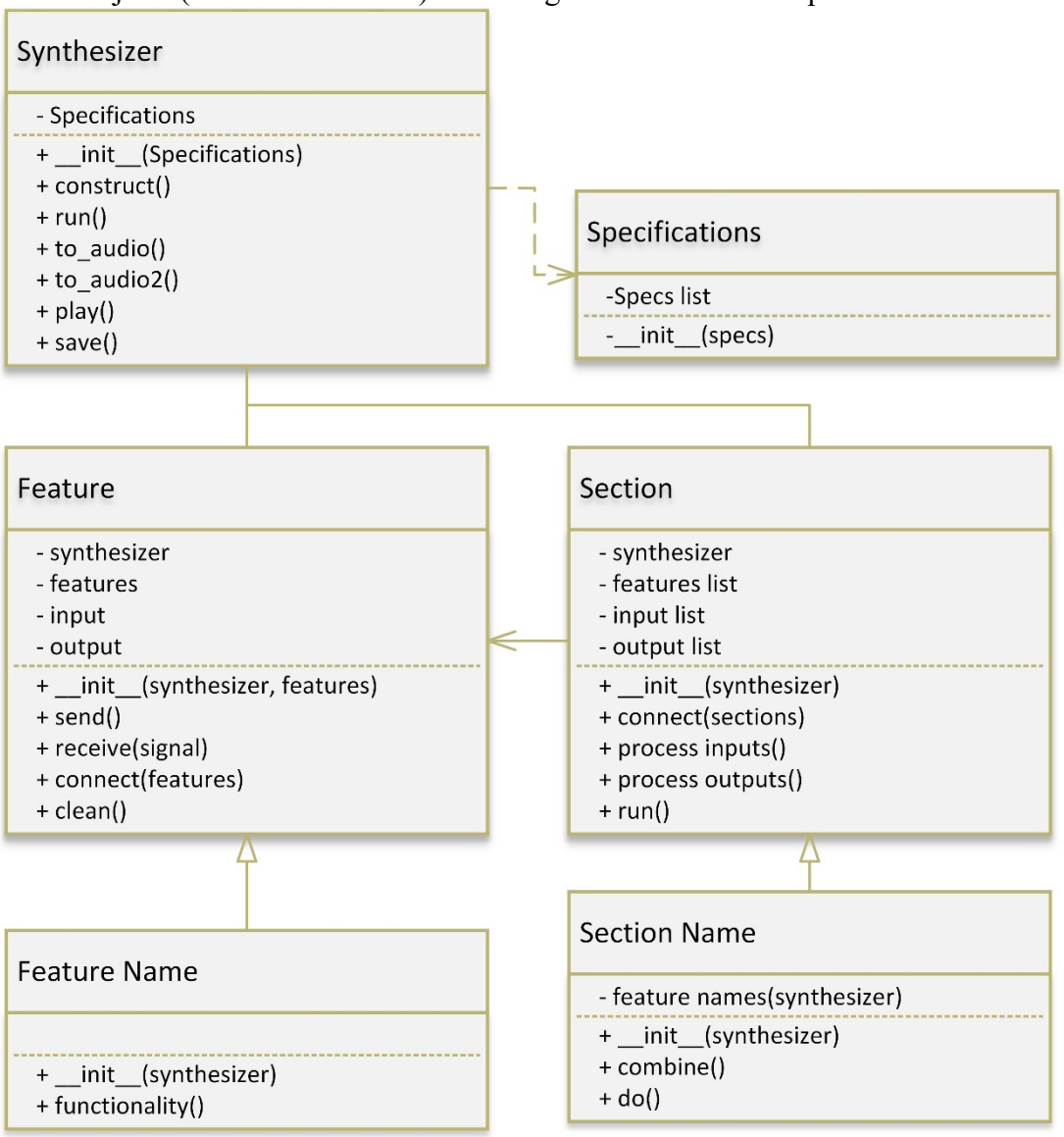


Figure 22 UML Class diagram of synthesizer implementation. A separate function initiates the Synthesizer object with the Specifications object. A synthesizer object contains feature objects and section objects that perform all processing.

Figure 24 contains the UML classes of every feature dependent on the feature class.

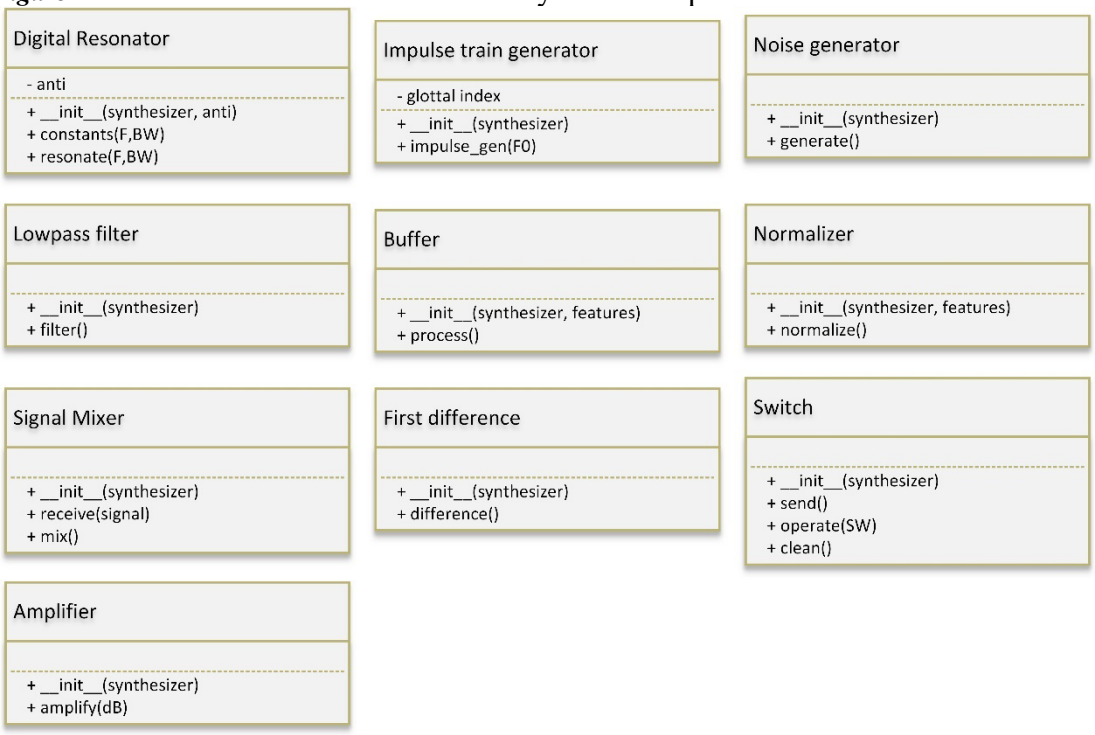


Figure 23 The UML class representation of every feature implemented for the synthesizer.

Figure 25 contains the UML classes of every section dependent on the section class.

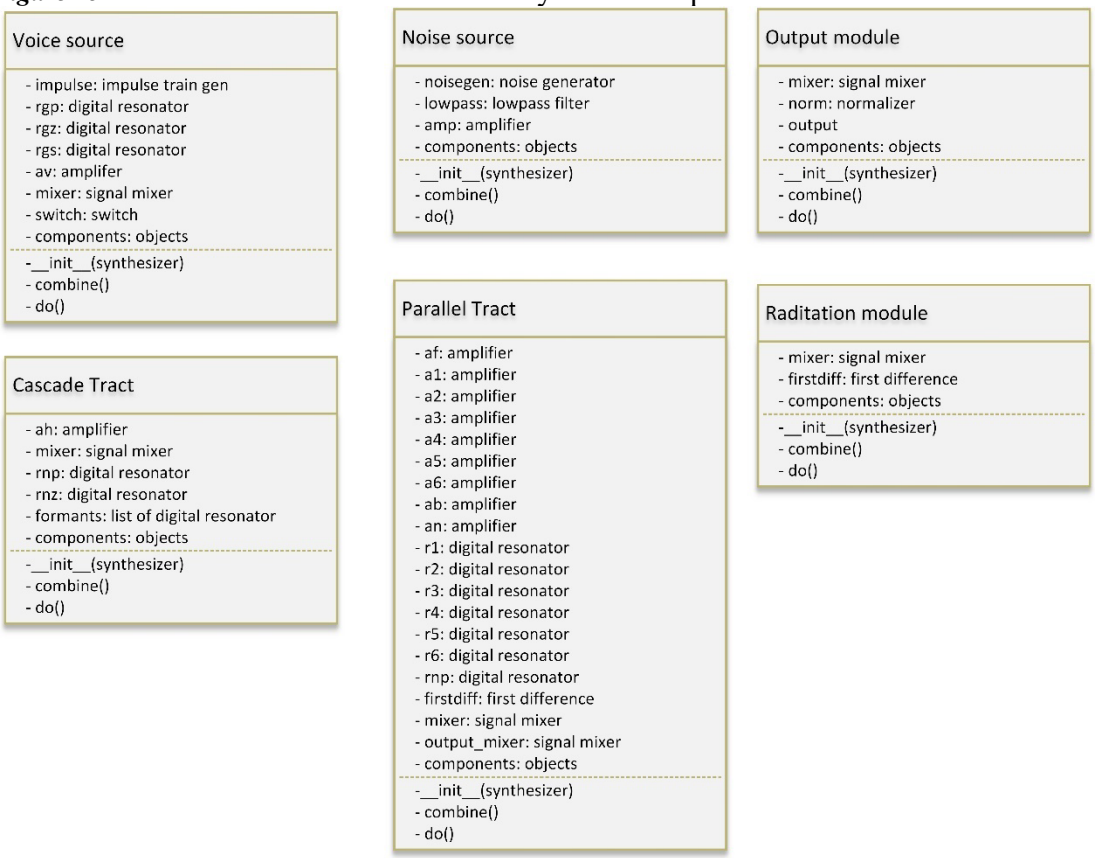


Figure 24 The UML class representation of every section implemented for the synthesizer.

(b) Pseudocode for the synthesizer object creation

Figure 26 contains a pseudocode for creating a synthesizer object. The specifications have to be specified for the desired sound waveform initially. If not, default parameters will be set according to the specifications class. Finally, a synthesizer object is created, and all specifications are assigned. After that, the synthesizer is constructed.

```
#Synthesizer creation

inputs: specs
output: initialised synthesizer object
if specs is None:
    set specs to default specs

synthesizer_object = Syntheisizer()

for spec in specs:
    if spec is formant or BW:
        synthesizer_object.specs[spec] = spec[list]
    else:
        synthesizer_object.specs[spec] = spec

synthesizer_object.contsruct()
```

Figure 25 Pseudocode for synthesizer object creation.

(c) Pseudocode for sentence synthesis

Figure 27 contains a pseudocode for sentence synthesis. It checks every word in the sentence, then every phoneme in the word. Then, for every phoneme in a word, it checks the type of phoneme and creates a synthesizer object according to that phoneme's prosody.

```
#sentence synthesis
inputs: list of words
outputs: synthesized audio playback

imports simple audio

output = []

for word in words:
    output.append(makeWord(word))

for i in range(number of words):
    playback using simple audio buffer

def makeWord(word)
# example word: word = ['T','IY']
    for phoneme in word:
        if else to check phonemes
            sound = makeSound(prosody information)
    return appended sounds

def makeSound(prosody information):
    create_synthesizer(default specs)

    if else to check type of phoneme
        #set specs according to prosody information
        #perform required phoneme specification setting
    synthesizer.run()#call the synthesizer run function
    synthesizer.to_audio()#call the synthesizer audio conversion/scaling function
```

Figure 26 Pseudocode for sentence synthesis.

3.3.7. Component simulations

It would be preferred to simulate every possible outcome of the synthesizer to verify the correct implementation. However, it would be more practical to simulate the functionality of every non-redundant component instead. For this reasoning, simulations for the digital resonator, cascading resonators, impulse train generator, first difference module, and low-pass resonator will be simulated for verification. All components were implemented in the software according to theoretical designs. Simulations were conducted using Python, and the matplotlib library was used for displaying plots.

(a) Digital resonator

A digital resonator simulation is presented in ***Figure 28***. The input parameters for the simulation were a resonant frequency of 2000 Hz and a resonant bandwidth of 200 Hz. The resonant frequency observed in the image is approximately 2000 Hz, and the bandwidth observed is approximately 177.5 Hz (BW = 2087.5 Hz -1910 Hz). The result aligns with expectations.

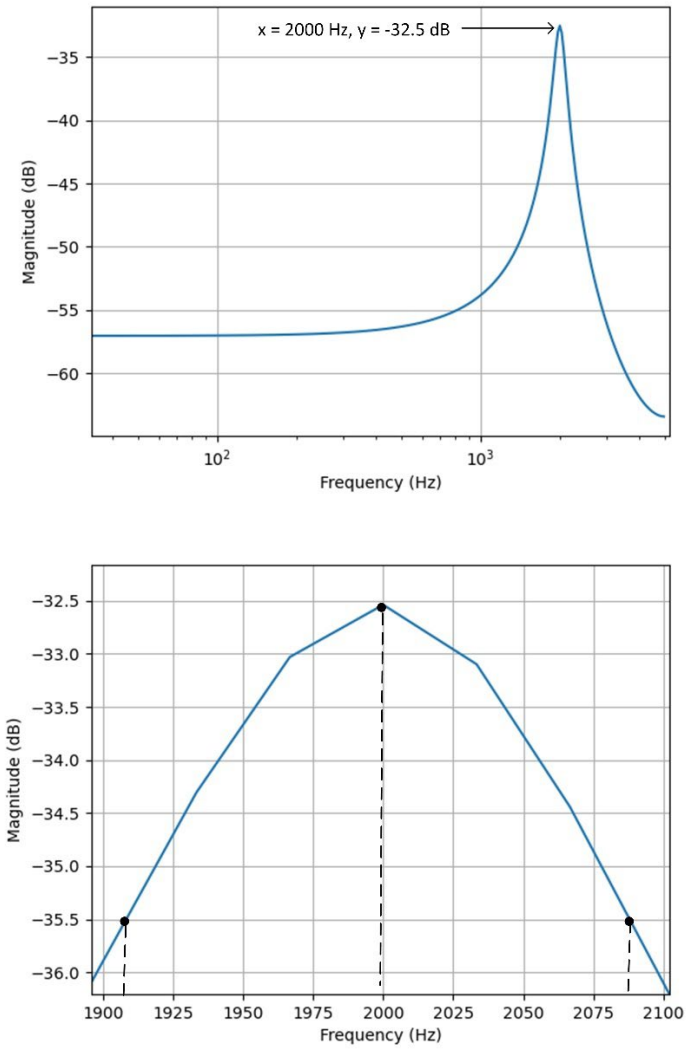


Figure 27 Simulation of digital resonator. The input parameters given was F = 2000Hz, and BW = 200 Hz. The figure below is a zoomed in plot.

(b) Cascading resonators

The functionality of cascading resonators was simulated in ***Figure 29***. The input parameters for the first resonator are; F = 1000 Hz and BW = 100 Hz, and the parameters for the second resonator are; F = 2000 Hz and BW = 200 Hz. ***Figure 30*** illustrates the area around the peak points for the cascade simulation. Cascading resonator parameters still lie within expectations.

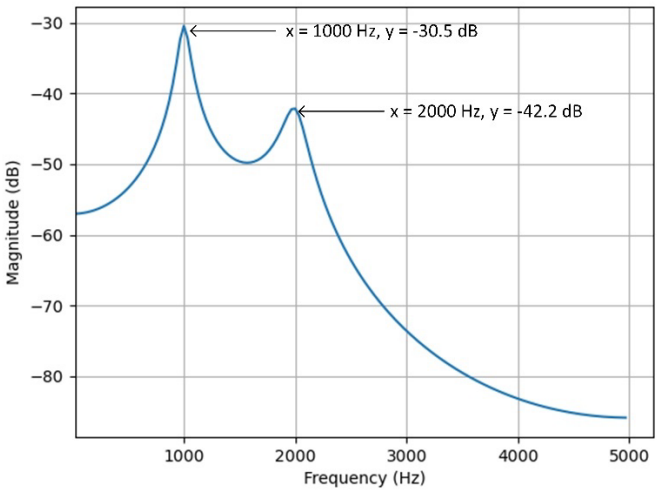


Figure 28 Cascading resonators simulation.

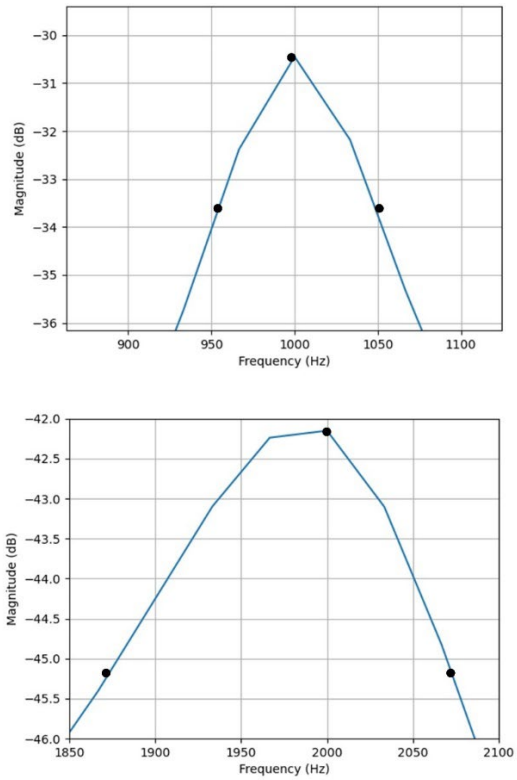


Figure 29 Zoomed in simulation of cascade resonators. Top corresponds to the first resonator and bottom corresponds to the second resonator.

(c) Impulse generator

Figure 31 illustrates the simulation of 500 samples of an impulse train generator. The input parameters specified were a sample rate of 10000 and a fundamental frequency of 100 Hz. An arbitrary value was used to specify the pulse magnitude. Pulses are expected to appear according to Eq. 12.

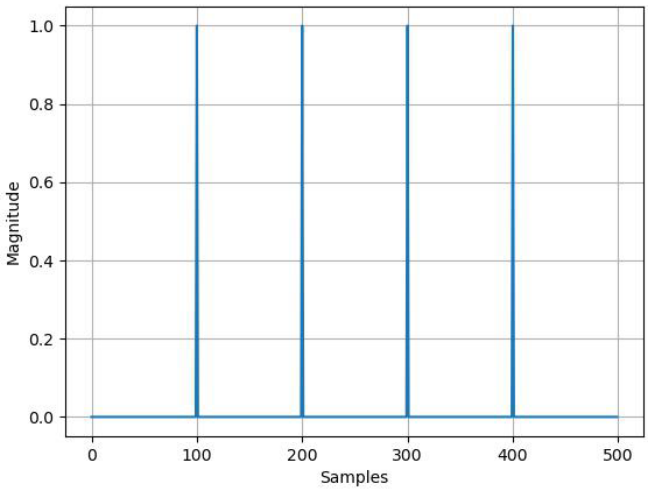


Figure 30 Simulation for 500 samples of an impulse train generator for sample rate of 10000, and fundamental frequency of 100 Hz.

(d) First difference component

Figure 32 is a simulation of the first difference component in the radiation characteristic. An arbitrary sine wave signal was used as input (blue), and the first difference operation is displayed as an output (orange). The first difference output is expected to be modelled according to Eq. 15.

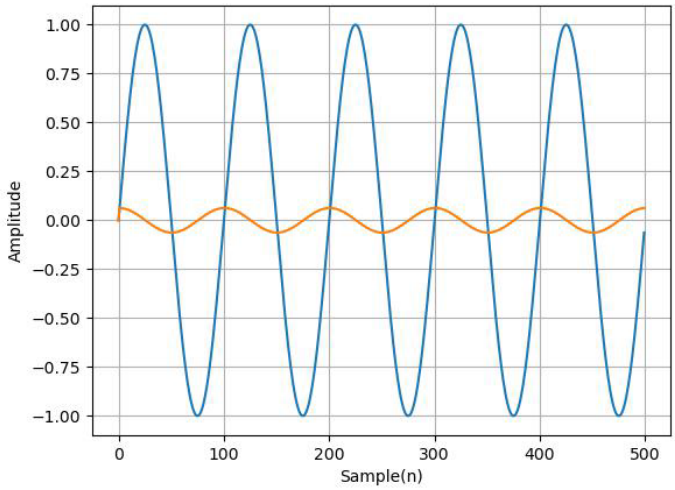


Figure 31 First difference component simulation. An arbitrary sine wave was used as an input signal (blue), and the output of a first difference component was simulated (orange).

(e) Low-pass resonator

Figure 33 is a simulation of a low-pass resonator discussed in section 3.3.1(e). By setting the resonant frequency to 0 Hz, a low-pass filter with a cut-off frequency of half the specified bandwidth is expected. The observed bandwidth is 300 Hz for a 1000 Hz bandwidth specification.

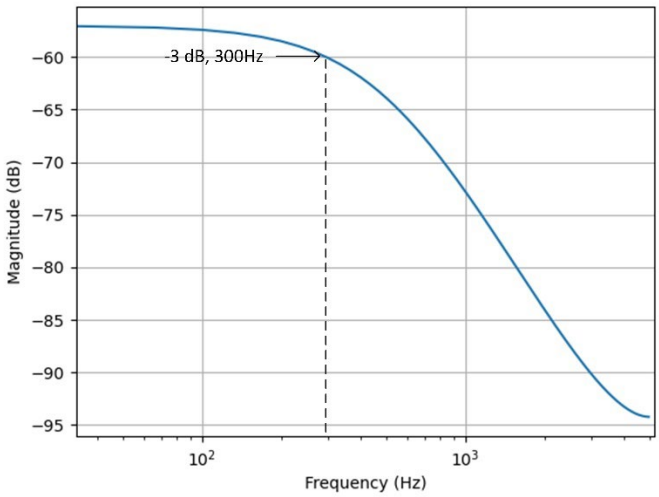


Figure 32 Simulation of the special case low-pass resonator. The input parameters for the resonator are F = 0 Hz, and BW = 1000 Hz.

3.4. Email client

For input to the system, an email client is required. The client will receive emails when a particular person's email address is recognised and displayed via a browser. The web browser library is used to display emails retrieved from the inbox. A GUI could instead be considered if system integration was accomplished. The output of an email is currently displayed in a Python console (see **Figure *8***) and in a browser.

3.5. Intercom system integration

The expected system design follows the previously discussed topics in sections 3.2 and 3.3. A high-level functional block diagram of the integrated subsystems is shown in ***Figure 34***. Although subsystem integration has yet to be achieved, the functional block diagram indicates the expectation from integration. Concerning hardware implementation, the system requires no self-developed hardware. Integration of the system onto an SBC (the Odroid) is yet to be done. Moreover, a cover for the Odroid has been bought off the shelf.

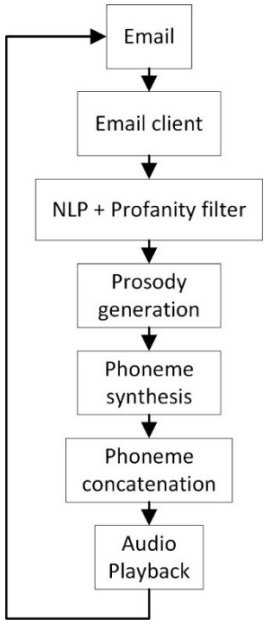


Figure 33 High-level functional block diagram of integrated subsystems.

4. Results

The results and qualification tests of the project are presented in the following sections.

4.1. Summary of results achieved

Table 12 and ***Table 13*** summarises the project intended outcomes and the actual outcomes achieved.

Intended outcome	Actual outcome	Location in report
Core mission requirements and specifications		
The announcement has to be broadcast in a synthetic voice with similar sounding voice qualities. The fundamental frequency, amplitude, duration, and formants in a speaker's voice should be modelled. Formant peaks should not differentiate more than 10% of the original voice.	A speech synthesizer was implemented that models the dynamics of fundamental frequency, amplitude, and formant characteristics. However, only individual words can be broadcast as integration has not yet been achieved. The first three formant peaks differentiate by 10.03%, 20.86%, and 17.39% respectively.	Section 4.2.1
The system should be able to convert arbitrary text input to speech information. NLP produces word error rates (WER) of 40% or below, after processing text. Processing for short non-complex sentences comprising of around 15-20 words should provide speech information to the synthesizer within 2 seconds.	An HMM optimized with the Viterbi algorithm can convert arbitrary text to speech information, yielding an accuracy of 97.55% to 98.48% with the corpus data used. This conforms to NLP having expected WERs of 2.45% to 1.52%. Processing for sentences consisting of 20 words provides speech information to the synthesizer after 0.58 seconds on average using a computer device.	Section 4.2.2

Table 10 Summary of results achieved Part 1

Intended outcome	Actual outcome	Location in report
Core mission requirements and specifications		
The message that is announced, should be filtered of all explicit inappropriate phrases, by using a comparative database. The accuracy of a large sample set, should be at 100%, to ensure correct filtering.	The accuracy of filtering a sample set of any size filters out 100% of the self-defined explicit profanity. Speech information for the message is now attached with silence when profanity occurs.	Section 4.2.3
A mail system should receive emails from a user. The text data that is sent by the user in the email, should be correctly reflected by the receiving end of the system, with no changes made to the input	The text data received by the system is displayed via a web browser and reflects no edits made on the receiving end of the system.	Section 4.2.4
Field condition requirements and specifications		
The intercom system has to be audible enough for a noisy indoor room environment. It should be operable from any location with internet access.	The synthesizer subsystem produces speech loud enough for noisy indoor environments with internet access.	Section 4.2.1

Table 11 Summary of results achieved Part 2

4.2. Qualifications Tests

The following sections describe the qualification test procedures implemented to obtain results.

4.2.1. Qualification test 1: Comparing formants and speech characteristics

- ***Objectives of the test or experiment***

Qualification test 1 is aimed at determining the formant and speech characteristics of synthesized speech and standard speech, so that differences between the two forms of speech can be compared. The test aims at verifying that the peak formant values present in both speech waveforms, do not differ by an error margin of at least 10%.

- ***Equipment used***

The equipment used comprises of the following tools and software:

1. The synthesis subsystem being tested.
2. A personal computer serving as a simulation platform.
3. A speaker for audio playback.
4. A personal smartphone with audio recording capabilities.
5. A USB cable for transferring audio from the smartphone to the computer.
6. Praat software for plotting audio waveforms.
7. An audio file converter to convert audio files to the .wav format.

In addition to the equipment specified above, recordings of the student's voice are required as the system has been modified to produce the student's voice.

- ***Test setup and experimental parameters***

The test environment requires that recordings be taken in a non-noisy environment with background room noise below 40 dB as most voices emit sound waves at a range of 60 dB. Other experimental parameters of the subsystem can only be varied through a subsystem console as there is not yet a GUI developed for it. Volume adjustments for the output of the speaker may need to be taken into consideration when testing. The test setup comprises of the following steps, assuming that the synthesis subsystem can be run from the computer:

1. Powering on the computer.
2. Plugging the speaker into the computer.
3. Launching the synthesis subsystem console.
4. Launch Praat software.
5. Plug the smartphone into the computer.
6. Enable file access permissions on the smartphone.
7. Launching the audio recording software on the smartphone.
8. Place the smartphone approximately 0.3 meters away from the speaker.

- ***Steps followed in the test or experiment***

The following steps were followed to perform the experiment:

1. Connect the test setup as described.
2. Modify the synthesis subsystem console parameters for the desired phoneme recording.
3. Start recording on the smartphone device.
4. Run the synthesis subsystem twice ensuring that audio playback is obtained.
5. Stop recording on the smartphone device and save the recording with a memorable name, and as a .wav file if possible.
6. Repeat steps 2-5 until all desired phonemes have been captured.
7. Copy the recording(s) from the smartphone to the computer.
8. If the recording extension is not a .wav file, use an audio converter to convert the file to the correct extension.
9. Open both the student's voice file and the .wav file with Praat software.
10. Select the view and edit option in Praat for the desired audio files to compare.
11. Extract waveform information using Praat tools.

- ***Results or measurements***

Figure 35 is a plot captured in Praat, of the resulting waveform produced by the synthesizer subsystem when making the phoneme 'AA'. ***Figure 36*** is a plot, similarly captured, of a waveform resulting from human speech of the phoneme 'AA'. These results are just an example of one of the many waveforms that were evaluated.

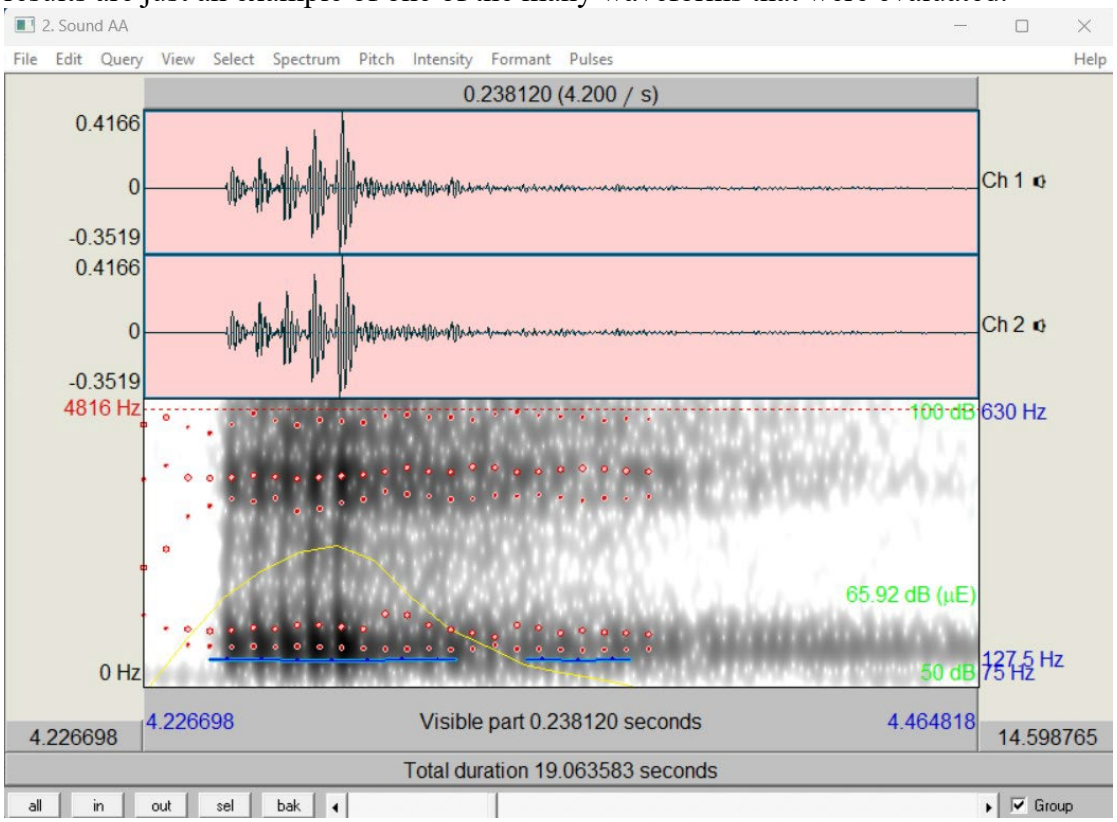


Figure 34 Result of the speech synthesizer subsystem making the 'AA' sound.

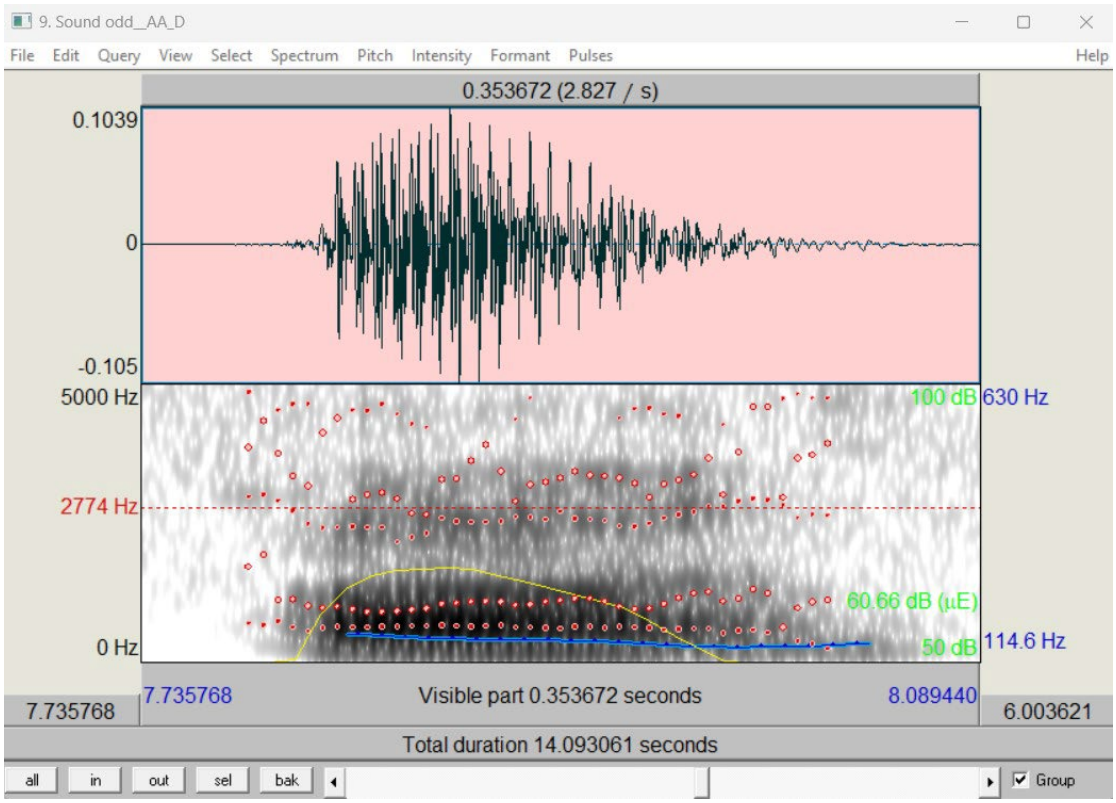


Figure 35 Result of human speech making the 'AA' sound.

Table 12 contains the measurement recordings of the synthesis subsystem output, for the experiment conducted to extract the first three formant peaks and bandwidths, as well as the fundamental frequencies and intensities. The average value for formants, fundamental frequency and amplitude of voicing is also indicated.

Phon.	F1 (Hz)	F2 (Hz)	F3 (Hz)	B1 (Hz)	B2 (Hz)	B3 (Hz)	F0 (Hz)	AV (dB)
'AA'	680	1059	3220	59	540	463	127	68.6
'AE'	754	2186	2843	1309	1006	392	117	70.4
'AH'	744	1564	3082	62	827	374	134	63
'AO'	564	857	3242	74	61	96	119	74.6
'AW'	775	1624	2929	65	747	696	127	61.6
'AY'	706	1683	3317	77	607	496	135	65.6
'EH'	485	2455	3304	15	213	134	122	66.5
'ER'	551	1670	2976	180	363	292	118	64.3
'EY'	606	2355	3323	71	327	150	145	59.9
'IH'	433	2217	3143	95	1506	340	124	62.8
'IY'	481	2305	3315	162	278	291	131	64.3
'OW'	753	1606	2966	449	888	458	119	61.2
'OY'	564	1234	3080	344	652	569	118	65.8
'UH'	576	1539	2603	205	1217	981	119	67.6
'UW'	769	2159	3409	294	70	111	116	65.4
AVG.	**629.4**	**1768**	**3117**	**230.7**	**620.1**	**389.5**	**124.7**	**63**

Table 12 Record of measurements taken from the synthesis subsystem, of the average formant peaks, bandwidths, fundamental frequencies, and intensities.

Table 13 contains the measurement recordings from human speech output, for the experiment conducted to extract the first three formant peaks and bandwidths, as well as the fundamental frequencies. The table also contains the average values of all parameters listed. Amplitude of voicing (AV) was not documented here as the value was used as an average of 60 dB to produce louder output in most cases.

Phon.	F1	F2	F3	B1	B2	B3	F0
'AA'	618	1010	2583	130	248	451	120
'AE'	705	1825	2602	138	144	78	113
'AH'	681	1271	2562	146	265	248	124
'AO'	475	709	2710	149	30	661	116
'AW'	682	1107	2430	219	204	256	119
'AY'	599	1383	2461	121	435	435	123
'EH'	469	2323	2830	48	428	271	119
'ER'	451	1511	2654	66	67	84	117
'EY'	475	2135	2835	121	556	599	138
'IH'	370	2165	2750	26	377	346	126
'IY'	300	2045	2960	45	200	400	115
'OW'	555	1275	2663	120	66	93	113
'OY'	473	1400	2513	105	105	260	113
'UH'	413	1025	2200	65	110	140	113
'UW'	365	1475	3185	52	133	547	114
AVG.	**508.7**	**1510.6**	**2662.5**	**103.4**	**224.5**	**324.6**	**118.9**

Table 13 Record of measurements taken from human speech output, of the average formant peaks and fundamental frequencies.

- ***Observations***

From observing the collected data, it is evident that there are a few outliers from the synthesizer outputs, such as the phoneme 'IH' and it's second bandwidth parameter. Outliers like that skew the average value of the formants, so every formant value will have to be individually considered for statistical analysis. From ***Figure 35*** and ***Figure 36*** it can be seen that the formant and bandwidth energies within the spectrogram approximately align with the expected outcome. However, the shape of the waveform does not match up as well. The amplitude of frication and aspiration have to be increased to generate a larger noise amplitude for the system. The fundamental frequency seems to be consistent and within expectations, indicating that if a different person's voice was used, there would be no problem in modelling it.

- ***Statistical Analysis***

A statistical analysis is performed on the measured data from the synthesizer and from the student's voice. The phonemes along with their respective formant errors were calculated using the Eq. 16,

$$\%error = \left(\frac{estimation - actual}{actual}\right) * 100 \qquad (16)$$

where *estimation* is the value obtained from the system, and *actual* is the value obtained from the person's voice. ***Table 14*** documents this error of the first three formants.

Phon.	F1 error (%)	F2 error (%)	F3 error (%)
'AA'	10,03236	4.851485	24.66125
'AE'	6.950355	19.78082	9.262106
'AH'	9.251101	23.05271	20.29664
'AO'	18.73684	20.87447	19.631
'AW'	13.63636	46.7028	20.53498
'AY'	17.86311	21.69197	34.78261
'EH'	3.411514	5.682307	16.74912
'ER'	22.17295	10.52283	12.13263
'EY'	27.57895	10.30445	17.2134
'IH'	17.02703	2.401848	14.29091
'IY'	60.33333	12.71394	11.99324
'OW'	35.67568	25.96078	11.37814
'OY'	19.2389	11.8571	22.56267
'UH'	39.4673	50.1463	18.31818
'UW'	110.6849	46.37288	7.032967
AVG.	**10.03236**	**20.85826**	**17.385**

Table 14 Statistical analysis to obtain average estimated error for formants.

From the table it can be seen that the three formant peaks have an average error of 10.03%, 20.86%, and 17.39% respectively. As expected, outlier phoneme measurements have a negative impact on the system performance.

4.2.2. Qualification test 2: Measuring word error rate and processing speed

- ***Objectives of the test or experiment***

Qualification test 2 is aimed at determining the accuracy of the NLP unit by testing its linguistic analysis algorithms, which is the HMM optimized with the Viterbi algorithm. Another aim for this qualification test is to determine the execution time that the NLP unit requires for sending speech information to the synthesizer. Although WER is used for measuring the accuracy of speech recognition systems (not TTS systems). the accuracy of the linguistic analysis algorithm could be used to translate *expected* WER. Linguistics analysis accuracy should therefore, translate to an estimated WER of 40% or below, and the execution time of the process is expected to be within 2 seconds.

- ***Equipment used***

The equipment used comprises of the following tools and software:

1. The Natural Language Processing subsystem being tested.
2. A personal computer serving as a simulation platform.

In addition to this, the system requires access to generic Python libraries such as the time library, and it should have the universal tag set corpus provided by nltk already downloaded. If it is not available on the computer, then the computer requires internet access.

- ***Test setup and experimental parameters***

The testing setup is performed by starting up the computer and launching the NLP subprocesses, i.e., the tokenizer and HMM POS tagger (including profanity filtering). The tester will be able to input any arbitrary sentence to the system. The tester will also be able to modify the training set size within the validation set of the POS tagger.

- ***Steps followed in the test or experiment***

The following steps were taken to perform the test:

1. Perform the test setup as required.
2. Modify the input sentence of the NLP unit to a non-complex sentence consisting of 20 words or less.
3. Modify the training test set size ratio in the HMM file.
4. Run each of the subprocesses and document the execution times from each process, as well as the accuracy of the system for that training set.

- ***Results or measurements***

Execution time is estimated from the beginning of the system receiving a text input of 20 words and is recorded in **Table *15***. The execution time in the second column of the table considers training time, and the last column considers pre-trained components.

Sub-process	**Execution time (sec)**	**Trained execution time (sec)**
Tokenization	0.00000323	0.00000323
HMM matrix training	5.0407	0
POS tagging	0.3104545714	0.3104545714
Profanity filtering	0.000011142	0.000011142
Lexicon file load	0.2623819	0.2623819
Phonetic transcription	0.003974	0.003974
Total	5.617524843	0.5768248434

Table 15 Estimated execution time of NLP unit

The accuracy of the POS tagger is obtained by dividing the corpus into trained and test data. The test data is then tagged using the algorithm and compared against its actual tags to determine how accurate the model is. The ratio of splitting the data is modified, and the accuracy is compared for every different ratio. Testing time is also recorded. The values are shown in **Table *16***.

Trained corpus data (%)	**Accuracy (%)**	**Testing time (s)**
70	97.549	27.256
80	98.56	27.513
90	98.48	33.896

Table 16 Accuracy of POS tagger

- ***Observations***

The execution time for sub-processes was measured in **Table *15***, and the total execution time could be estimated to measure the performance of the NLP unit. It is important to note that these values are from a modern computer system. Integration onto the single board computer (SBC) will most likely yield slower results as the processor is not as powerful. Accuracy of the POS tagger was proven to increase with a larger corpus provided. However, testing using a larger trained corpus data becomes slower as a result. This should not influence the efficiency of the system if pre-trained data is used. Also, by observing the change in accuracy, the improvement is miniscule and perhaps a smaller corpus could be used.

- ***Statistical Analysis***

As for statistics such as word error rate (WER), it cannot be determined from text analysis alone. WER is determined by having the synthesizer produce speech for a speech recognition system or a person. The WER is then measured by the discrepancies between the perceived output and the given input. The extent to which WER can be considered is only by *expectations*.

Estimated WER can be assumed to be the error in words perceived for the given input of an *ideal* system. Therefore, the *expected* WER could be calculated as the percentage of inaccuracies of the POS tagger, i.e., 100% - accuracy. Based on this, *expected* WER is calculated and presented in ***Table 17***. In general, WER is only a good performance measure for speech recognition systems and not TTS systems.

Trained corpus data (%)	Accuracy (%)	Expected WER (%)
70	97.549	2.451
80	98.56	1.44
90	98.48	1.52

Table 17 Expected WER

4.2.3. Qualification test 3: Measuring accuracy of profanity filter

- ***Objectives of the test or experiment***

The objective of the test is to measure the number of words that is filtered for the profanity filtering subsystem. It should be noted that the requirement is for an *explicit* profanity filter, therefore, no derivations of profanity can be filtered by the system. The performance measure of the *explicit* profanity filtering system is expected to yield an accuracy rate of 100%.

- ***Equipment used***

Equipment used for performing this test is as follows:

1. The profanity filtering subsystem.
2. A computer which serves as a simulation platform.

- ***Test setup and experimental parameters***

The testing setup is performed by starting up the computer and launching the HMM POS tagger which includes consists of the profanity filtering subsystem. Profanity filtering occurs during POS tagging so that profanity can be tagged with certain characters to indicate that it is profanity. The tester will therefore have to make changes to the POS tagging test sentences.

- ***Steps followed in the test or experiment***

The steps followed to perform the test is as follows:

1. Perform the test setup as required.
2. Edit the sentence to a sentence that contains profanity as specified in the dictionary.
3. Observe the output.

- ***Results or measurements***

Figure *36* contains an example output of the profanity filter. In the sentence printed out above the output, the words 'lookee' and 'garbo' have been defined in the profanity dictionary.

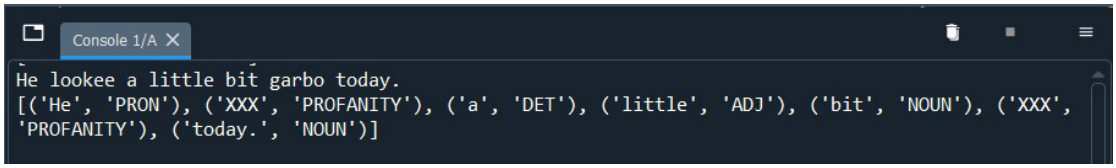


Figure 36 Profanity filtering example output

- ***Observations***

As seen in the output of **Figure *36***, the filter can mark words as profanity correctly. The implementation of profanity filtering is a trivial one. The model will always yield an accuracy of 100% since explicit profanity filtering compares tokens with a pre-defined lookup table. This however, does not mean that it is a perfect filtering tool. This profanity filter does not handle derivations of profanity, but its requirement is for *explicit* profanity filtering, and so it is be treated as such.

4.2.4. Qualification test 4: Validation of email output

- ***Objectives of the test or experiment***

The objective of this test is to verify that the mail system receives emails from a user, and that email text is correctly reflected by the receiving end of the system, with no changes made to the input. The objective is thus to visually verify an arbitrary email and verify that text is handled correctly.

- ***Equipment used***

Equipment used for performing this test is as follows:

1. The email subsystem to be tested.
2. A computer used as a simulation platform.
3. A smartphone or any other device capable of sending emails.

In addition to the required equipment used, the computer used as a simulation platform should have access to the internet so that it can retrieve emails from the email address. The system email address required is ir.intercom@outlook.com.

- ***Test setup and experimental parameters***

The test setup is as follows:

1. Turn the computer on.
2. Ensure that the sending and receiving ends have internet access.
3. Open the email subsystem.
4. Open a mail client on the sending device.

- ***Steps followed in the test or experiment***

The steps followed to perform the test is as follows:

1. Perform the test setup as required.
2. On the sending device, send an email containing some text to the system address previously specified.
3. When the email is sent on the sender's end, on the computer, run the email subsystem.
4. A browser should open displaying the contents of the email for verification.
5. The text message can also be verified on the console.

- ***Results or measurements***

Figure 37 and ***Figure 38*** display the results after sending an email to the email subsystem as per the test procedure. The former image indicates output on the browser, and the latter indicates output on the console.

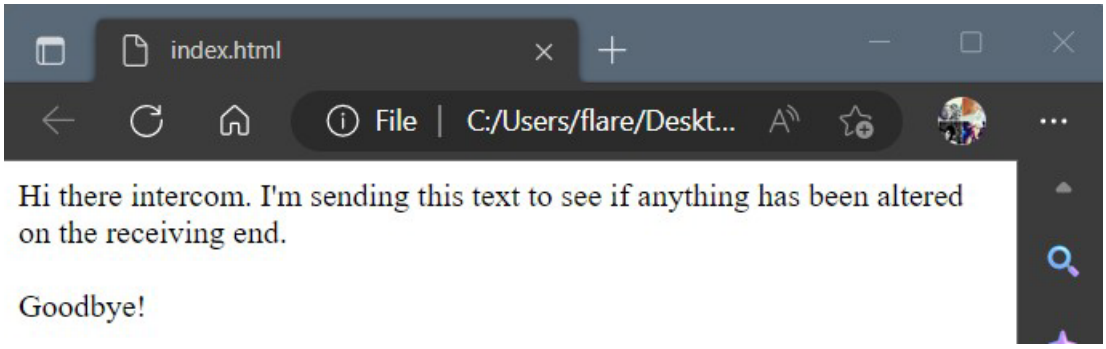


Figure 37 Email output on the browser.

```
Reloaded modules: mail_client_ver3
BEGINNING PROGRAM

Receiving email

Subject: Testing for results section.
From: Ishaque Ramedies <flarez0912@gmail.com>
Hi there intercom. I'm sending this text to see if
anything has been
altered on the receiving end.

Goodbye!

================================================
================================================
```

Figure 38 Email output on the console.

- ***Observations***

The email text received by the subsystem is the exact same text that was sent to the subsystem. No text has been removed or edited and visual verification shows that the tester can be identified from their email address.

5. Discussion

The following sections discuss how the results are interpreted, the critical evaluation of the design, design ergonomics, the health, safety and environmental impacts, and finally the social and legal impacts of the design.

5.1. Interpretation of results

The results obtained in section 4 along with some considerations will be discussed in the following sections.

5.1.1. Overview of results

With regards to the results required and obtained for the system (**Table *10*** and **Table *11***), The system natural language processing aspects have been achieved. As for the requirements of the text-to-speech synthesizer, measuring formant peaks is a reasonable metric to consider when implementing a formant synthesizer. However, when other synthesis techniques are considered, then analysing the formants of speech may not be the best metric to use. As the project progressed, it was evident that intelligibility tests are preferred for systems that cannot be accurately classified, such as in speech systems.

To summarise the results, the specification for testing speech, **for a particular person's** voice was not achieved, but it still sounds adequate or intelligent enough to understand. The natural language processing unit yields very accurate and fast results. However, this subsystem has yet to be integrated onto hardware, so the performance of the subsystem could be lower than measured, using a computer. Explicit profanity filtering was seen as trivial implementation as a LUT could be used to solve the problem. As for emails, they can be received by the system, and the system is able to interpret the text from them correctly, which the display feature verifies.

An overview of the results just mentioned is provided in ***Table 18*** and each subsystem target requirement will be discussed.

Overview of requirement	Target specification
Speech should be produced, and should sound like a particular person.	Formant error rate of 10%, and modelling considers pitch, intensity, duration, and loudness.
Fast execution of NLP, and high accuracy of the subsystem.	WER of 40% or below, and execution time of NLP under 2 seconds.
Filtering out all explicit profanity.	A profanity filtering accuracy of 100%.
The email should be verifiable by the user.	Development of an email display feature.

Table 18 Overview of target specifications

5.1.2. Comparing formants and speech characteristics

With regards to the results obtained from Qualification Test 1 (section 4.2.1), the cause of the resulting waveforms is due to not adjusting the amplitude of aspiration. The amplitude of aspiration adds breathiness to the output speech so that it is more natural sounding. As seen in the waveforms, the synthesized waveform is more repetitive, as it is generated from a impulse train defined by a sample rate. Failure to consider all sources of sound, as discussed in section 3.3.1(a), may result in speech output that sounds robotic.

Another cause of the results obtained is most likely the input system used to make the recordings initially. Microphone quality affects overall noise levels and how well a waveform can be interpreted in software. Some parameters however require the use of a parallel configuration (such as frication sounds), which was not used for the results obtained. Furthermore, tuning of every parameter is a long process and so it may be that tuning is an issue.

One more design flaw that was used, is that of modelling the formants and all other important parameters by considering their start and stop points. In general, it would seem that formants are linear over time, but some formants are logarithmic in how they change over the period of producing a vowel sound. To achieve the best possible sound output, mathematical functions for every formant could be implemented that best represents how that formant changes over time. The naïve approach was used in this case, which still produced intelligible speech, however if future work was to be considered then extracting formants should be handled in a more elegant way.

Some of the design considerations that were explored consisted of methods requiring large amounts of data. These are the data-driven methods. Although these methods are known to yield better speech results, the process is long and may take lots of time to collect all the data and train the system. That is why, today, there are only a few of these modern synthesizers which have been reproduced for distribution. Something like that cannot be achieved from first principles within a short amount of time.

Formant synthesis is a well-developed technique whereby there is a lot of flexibility in producing speech. There are many parameters to consider, however, it is the most logical form of rule-based synthesis that requires little data to yield an adequate speech output. Implementing the system design in class format is the best way to visualise sections and components in the system. Things that would be considered for changing would be the prosody generation function., since it does not consider all the possible parameters when tuning.

5.1.3. Measuring word error rate and processing speed

With regards to the results obtained from Qualification Test 2 (section 4.2.2), the natural language processing module obtained very adequate results. This subsystem alone, however, cannot determine WER. WER was incorrectly chosen as a design specification, and to determine whether the NLP unit was good enough, should have been determined by considering its accuracy

The accuracy of the system is very accurate for words that may already be present in a corpus. That is why the accuracy of the system is so high. Training data and test data were taken from the same corpus, so there is a natural pattern to how sentences are formed in that corpus. If differing corpora was used as test data, for instance, corpus data from a sports newspaper, and corpus data from a business newspaper, then the accuracy may not be so high. The availability of corpora for schools should be considered for development if the accuracy of the system is to remain high.

As for the design decisions made for linguistics analysis implementation, the hidden Markov model is a very good statistical model for parts of speech tagging. The problem with the hidden Markov model was that it took too long to train data without a Viterbi algorithm, which essentially reduced the overall space and time complexity of the system. The system therefore operates at a very fast rate. This subsystem was however, only tested using a computer since integration was not accomplished. So, the results are taken with a grain of salt, and should it should be assumed that the results on a hardware platform will not be as good. But it should still be able to meet the project specifications.

5.1.4. Measuring accuracy of profanity filter

With regards to the results obtained from Qualification Test 3 (section 4.2.3), the result obtained from the profanity filter was as expected. The implementation is a trivial one, however it manages to accomplish the design specification. A look up table can be used in most cases where a direct relationship is assumed, and since the implementation of an explicit profanity filter was required, this implementation was used. This does however, cause problems for the system should the input of the system be derivations of profanity. The system will not be able to filter that out.

Standard classifiers may be used to accomplish profanity filtering, as well as manual rule-based implementations. These classifiers usually involve methods similar to the hidden Markov model, where it requires information about every character, and their position in the word.

5.1.5. Validation of email output

With regards to the results obtained from Qualification Test 4 (section 4.2.4), the result obtained is a display of the email that was sent to the system. This is verified visually by the user of the system. The display of the mail system could alternatively be implemented using a graphical user interface. This implementation makes the design more complex, however it also compiles all elements of the project into a single application, thus making readability much better. In hindsight, the project requirement that was set out to be met was accomplished, and the solution is more efficient.

The implementation of the email client requires that an API be used to grant access to the system email address. Domain specific addresses like Google, have reduced accessibility to their systems through Python libraries, so the implementation is restricted in that the system is required to use a domain that is not Google.

5.2. Critical evaluation of the design

5.2.1. Aspects to be improved in the present design

Integration of the project has to be accomplished so that all aspects of the project can be tested more thoroughly. A profanity filter that handles explicit profanity might not be the best solution, but considering that its applicable environment is a school, there are assumptions that can be made that teachers will not send profane text to the system. The synthesis implementation does not scale phonemes correctly at times when concatenating them. This part of the project has to be looked into. A GUI implementation would serve as a better ergonomic choice for the user. The lexicon used in obtaining prosodic information does not contain any part of speech description, so the use of a different lexicon will have to be considered. The corpus used is also limited in that it does not explore all possible words, however it is large enough for this implementation.

5.2.2. Strong points of the current design

Synthesis of phonemes is successful, although it requires tuning, the temporary system yields intelligible words that a person can distinguish. A part of speech tagger using a HMM and Viterbi algorithm provides a very accurate solution. The formant synthesizer as a whole is a good design implementation choice that only requires a lot of parameters tuning and a good prosody function to make results sound better.

5.2.3. Under which circumstances is the system expected to fail?

The system is expected to fail when it receives junk values as input text.

5.3. Design ergonomics

The design implements a web browser whereby the user can view the email. The system is not integrated, and if it were, the design would provide an interface for users to modify synthesizer parameters and open an aaplication.

5.4. Health, safety and environmental impact

The volume of the output speaker should be adjusted to levels that are suitable for human hearing and not hazardous to the environment.

5.5. Social and legal impact of the design

The design of subsystems uses algorithms and models that are not my own solutions, such as part of the model of the synthesizer, and the algorithms for natural language processing. These should not pose any legal issues however, since the ideas of the solutions have been distributed to everyone. The system however cannot be used to imitate the voices of other people without their permissions.

The synthesizer can be used to assist people that are illiterate, blind or similarly are vision impaired. There are thus a lot of use cases for these cases and similar cases.

6. Conclusion

6.1. Summary of the work completed

This report describes the work carried out on the design of a text to speech synthesis system, with the objective of delivering speech in the voice of a user, while filtering out profanity.

A literature survey was completed on natural language processing elements and speech synthesis techniques. The software implementation of a natural language processing module and a speech synthesis subsystem, was designed from first principles. The hardware elements of the system that would lead to system integration has not been realised. The core of the system is a Klatt modelled formant synthesizer, which produces speech from formant analysis. All software procedures and details were developed surrounding the Klatt synthesizer, and all simulations and implementations conducted was done using the Python language. The subsystems of the system were implemented and tested throughout the design stage of the project. Several recordings of the synthesizer output were taken and the result for one of them was shown in the report.

6.2. Summary of observations and findings

Formant synthesis has been achieved although error rate, due to outliers that were not properly tuned, is higher than expected. Natural language processing provides accurate speech information to the synthesizer with a fast execution time. Filtering of explicit profanity by a self-defined database was achieved although the implementation is trivial. The email sent to the system can be verified through a web browser.

6.3. Contribution

New software that was learned is Praat which handles sound waveforms, and spectrograms which was not handled in any previous modules. Knowledge about natural language processing, specifically for speech systems was learnt and implemented into a large part of the project. Knowledge surrounding speech synthesis techniques was also covered throughout the course of the project.

6.4. Future work

The prosody generation function was not correctly implemented and does not consider any pause breaks during concatenation. Some testing choices were not properly researched, a good test for the system is an intelligibility test. Also, no consonants at the middle and end of words were tested, although consonants at the beginning of words could still be used.

7. References

[1] Y. Zhigang, "An overview of speech synthesis technology," in *Eighth International Conference on Instrumentation and Measurement, Computer, Communication and Control*, Beijing, China, 2018.

[2] U. D. Reichel and H. R. Pfitzinger, "Text Preprocessing for Speech Synthesis," in *TC-Star Speech to Speech Translation Workshop*, Munich, Germany, 2006.

[3] Y. L, "Improvement for the automatic Part-of-speech Tagging Based on Hidden Markov," in *2010 2nd International Conference on Signal Processing Systems (ICSPS)*, China, 2010.

[4] L. R. Rabiner and R. W. Schafer, Theory and Applications of Digital Speech Processing - First Edition, Santa Barbara: Pearson Higher Education, 2011.

[5] D. H. Klatt, "Software for a cascade/parallel formant synthesizer," Massachusetts Institute of Technology, Cambridge, Massachusetts, 1979.

[6] G. Fant, Acoustic Theory of Speech Production, 1960.

[7] S. Y, "ATR-TALK speech synthesis system," in *Proceedings of International Conference on Spoken Language Processing*, 1992.

[8] K. H, M.-K. I. M and d. A, "Restructuring speech representations using pitch-adaptive time-frequency smoothing and an instantaneous-frequency-based F0 extraction possible rol of a repetitive structure in sounds," in *Speech Communication*, 1999, pp. 187-207.

[9] H. Zen, "Statistical parametric speech synthesis," in *Speech Communication*, 2009, pp. 1039-1064.

[10] C. Zhen-huai and Y. Kai, "An investigation of implementation and performance analysis of DNN based speech synthesis system," in *Proceedings of International Conference on Sociology and Psychology*, 2014.

[11] H. K. Dunn and S. D. White, "Statistical Measurements on Conversational Speech," *The Journal of the Acoustical Society of America,* vol. 11, no. 3, 1940.

[12] N. R. French and J. C. Steinberg, "Factors Governing the Intelligibility of Speech Sounds," *The Journal of the Acoustical Society of America,* vol. 19, no. 1, 1947.

[13] D. H. Klatt, "Acoustic Theory of Terminal Analog Speech Synthesis," in *International Conference on Speech Communication and Processing*, 1972.

Part 4. Appendix: technical documentation

A.1 HARDWARE part of the project

Record 1. System block diagram

Figure ***Figure 39*** is the block diagram for the system.

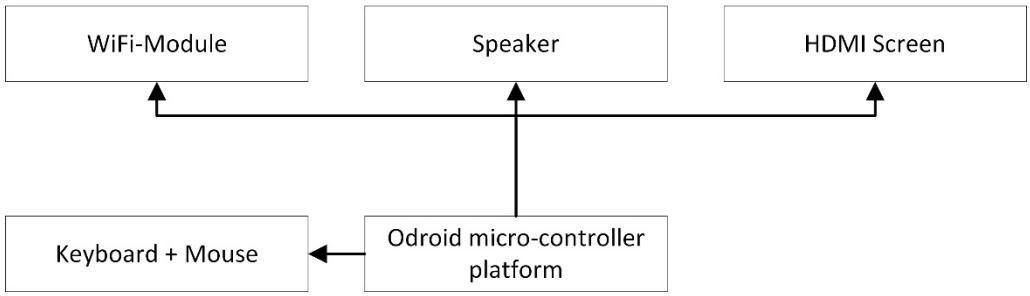


Figure 39 System block diagram

Record 2. Systems level description of the design

The following record provides a systems level description of ***Figure 39***.

The code is run on an SBC, which was selected as the Odroid micro-controller platform. The output emails and user interface are displayed on an HDMI screen. The audio output of the synthesizer is played over a loud speaker. A Wi-Fi module enables setting up of an internet connection with the system. To navigate the user interface, a keyboard and mouse are required to launch the application. The SBC is powered through a power outlet.

Record 3. Complete circuit diagrams and description

No physical circuits were considered for this project.

Record 4. Hardware acceptance test procedure

The user should ensure that all peripherals are plugged into the system before powering. Once powered, a red LED will flash on the SBC, and after some moments the screen should turn on. The user interface should be navigable from this point.

Record 5. User guide

After powering the device, the user should log in to the system. An application containing the operable software should be launched, and it should operate for as long as it is powered.

A.2 SOFTWARE part of the project

Record 6. Software process flow diagrams

Figure 40, ***Figure 41***, ***Figure 42***, ***Figure 43***, and ***Figure 44*** represent software process flow diagrams of the implementation and UML class diagrams of the synthesizer design.

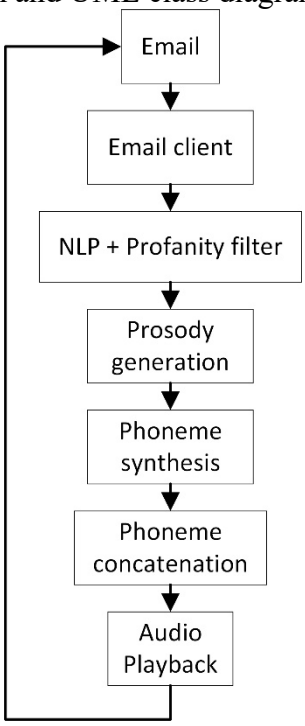


Figure 40 Software process flow of integrated software.

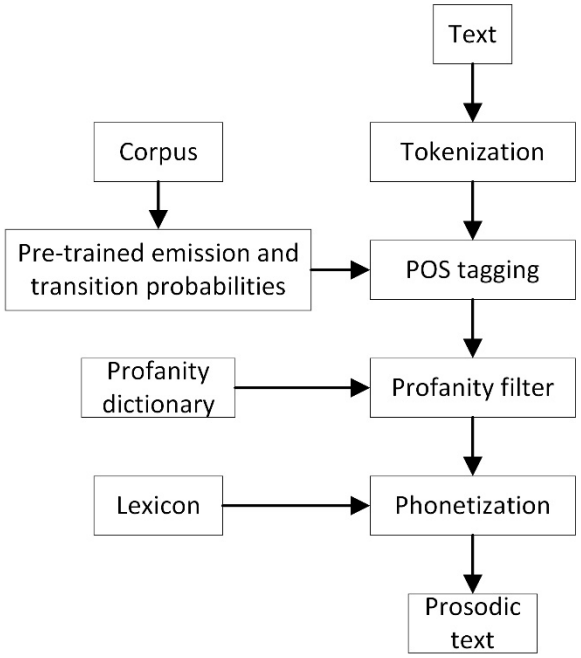


Figure 41 Software process flow of NLP unit

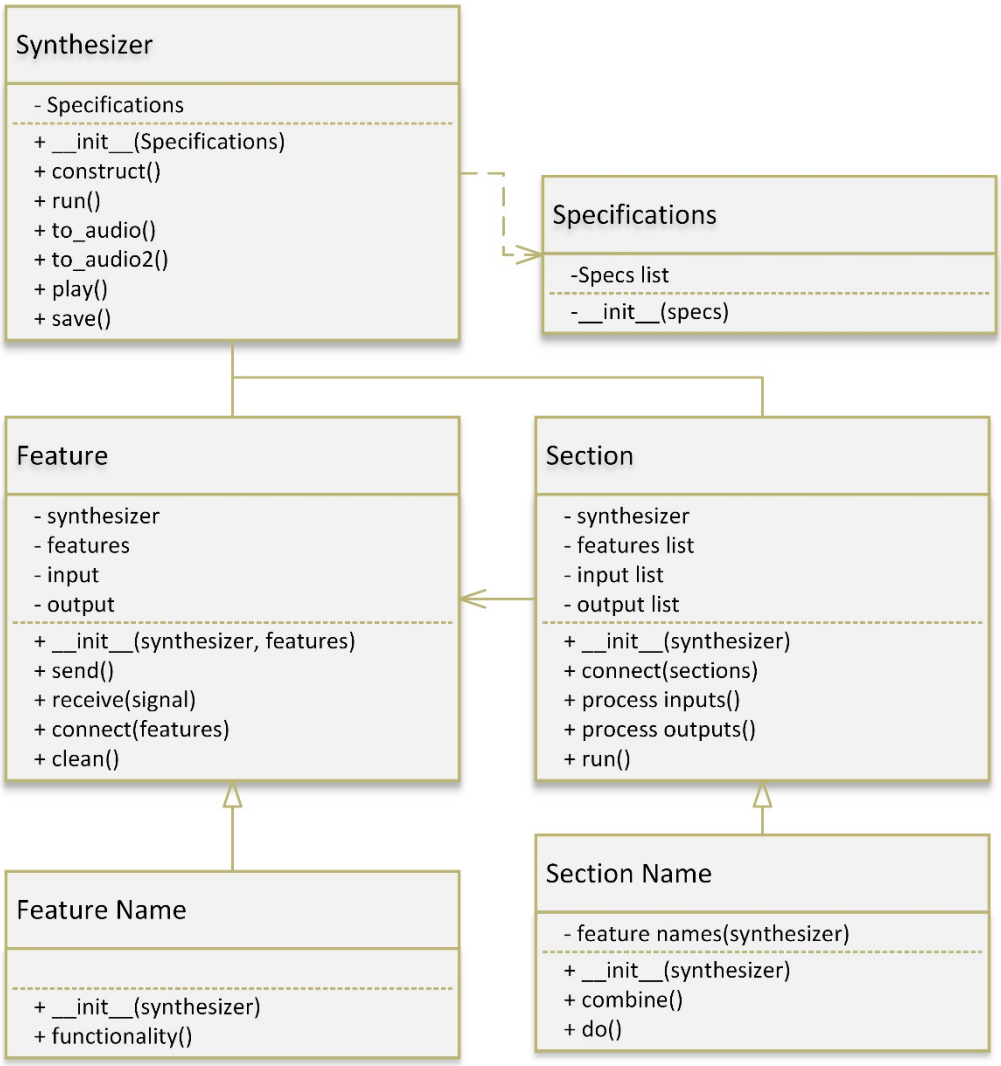


Figure 42 UML Class diagram of the synthesizer module

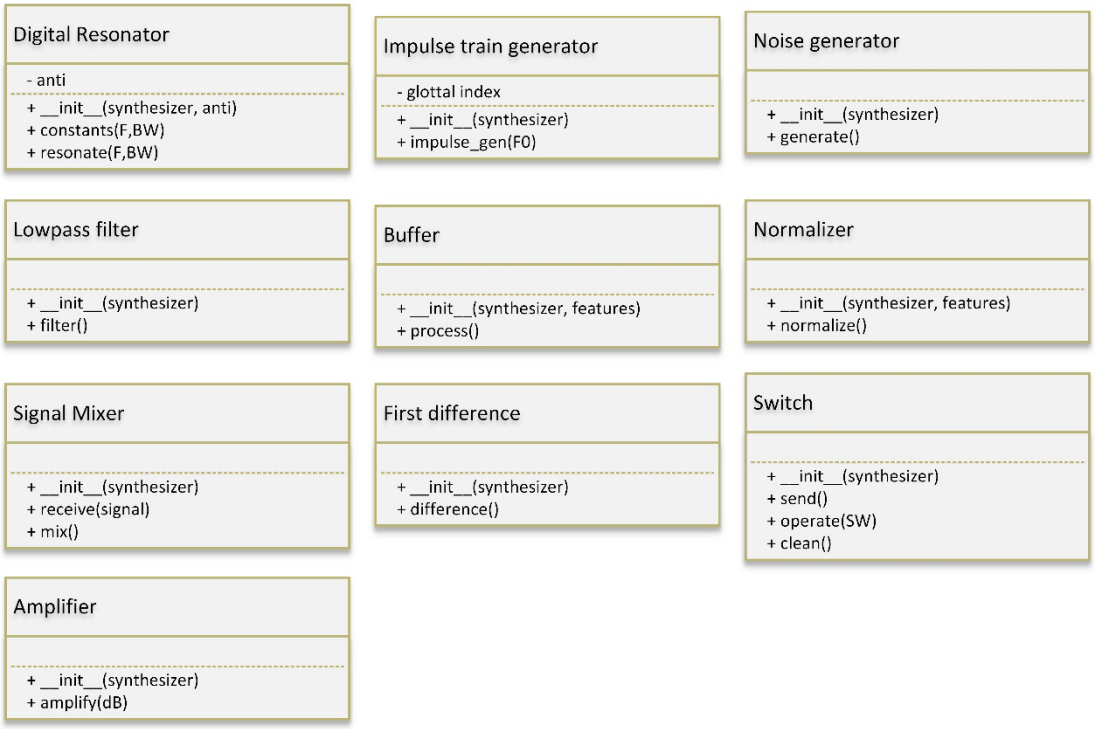


Figure 43 Continued UML Class diagram of the synthesizer.

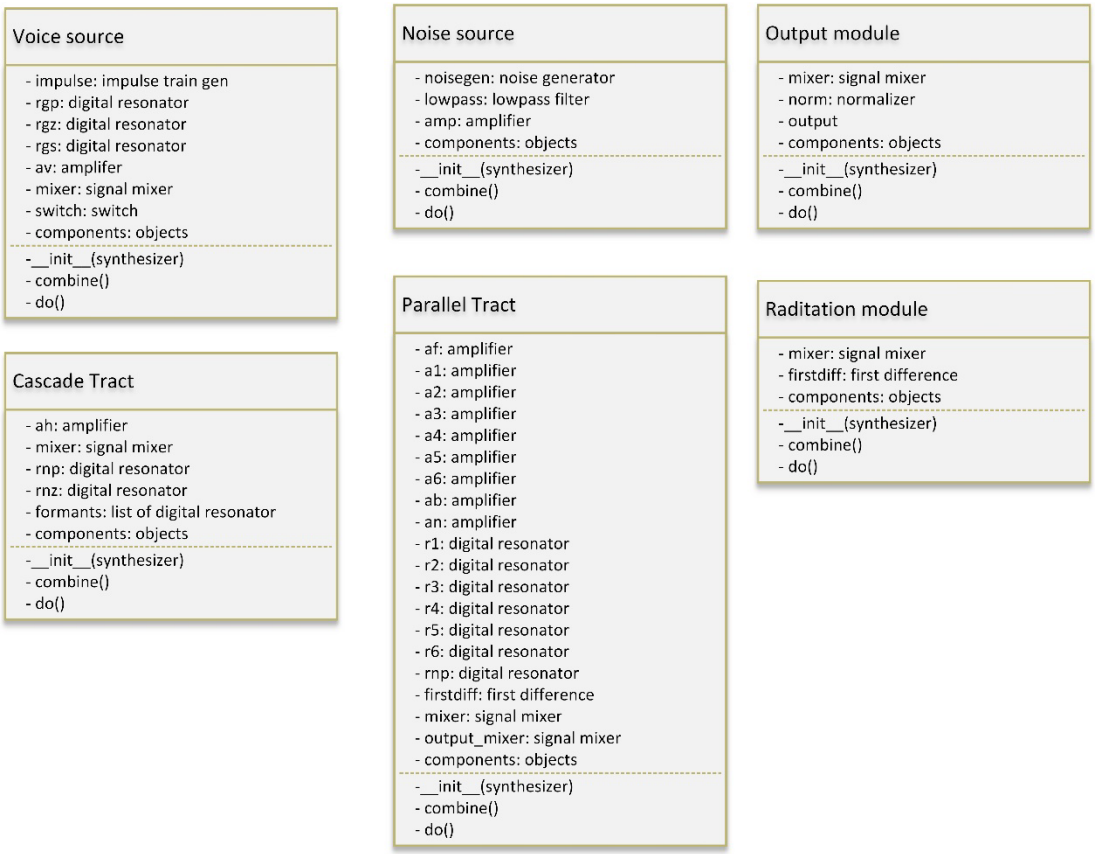


Figure 44 Continued UML Class diagram of the synthesizer module

Record 7. Explanation of software modules

Figure 40 describes the connection of all software modules. The first block is the email client. The email client waits for a user to send an email to the system, which it then receives and displays. The next section is NLP and profanity filtering, where the text in the email that was received will be processed by tokenizing the text and attaching a part of speech to each text. It will also interact with a lexicon where it will assign speech information to the text. If profanity is detected in a self-defined database, then the speech information that it will be marked with will be silence.

After speech information or phonemes are attached to each word. A formant synthesizer will synthesize each phoneme and concatenate the results. After all of the synthesis is completed, the system will playback the audio through a speaker. Finally the email will await the next email announcement to be displayed.

Record 8. Complete source code

Complete code has been submitted separately on the AMS.

Record 9. Software acceptance test procedure

A fully integrated system has yet to be developed. However, individual subsystems can be run and tested.

Record 10. Software user guide

A fully integrated system has yet to be developed. There is, therefore, no guide on how to set up the software or run the integrated system.

A.3 EXPERIMENTAL DATA

Record 11. Experimental data

Table *18*, **Table *19***, **Table *20***, **Table *21***, and **Table *22*** are the consonant values that are still under investigation and tuning. ***Table 23*** and ***Table 24*** are the latest results from fine tuning the system. ***Table 25*** is the current best specifications for the results in ***Table 23*** and ***Table 24***.

Sonor	F1	F2	F3	B1	B2	B3
w	290	610	2150	50	80	60
y	260	2070	3020	40	250	500
r	310	1060	1380	70	100	120
l	310	1050	2880	50	100	280

Table 19 Parameter values for sonor consonants.

Fric.	F1	F2	F3	B1	B2	B3	A2	A3	A4	A5	A6	AB
f	340	1100	2080	200	120	150	0	0	0	0	0	57
v	220	1100	2080	60	90	120	0	0	0	0	0	57
θ	320	1290	2540	200	90	200	0	0	0	0	28	48
δ	270	1290	2540	60	80	170	0	0	0	0	28	48
s	320	1390	2530	200	80	200	0	0	0	0	52	0
z	240	1390	2530	70	60	180	0	0	0	0	52	0
š	300	1840	2750	200	100	300	0	57	48	48	46	0

Table 20 Parameter values for fricative consonants.

Aff.	F1	F2	F3	B1	B2	B3	A2	A3	A4	A5	A6	AB
č	350	1800	2820	200	90	300	0	44	60	53	53	0
ĵ	260	1800	2820	60	80	270	0	44	60	53	53	0

Table 21 Parameter values for affricate consonants.

Plo.	F1	F2	F3	B1	B2	B3	A2	A3	A4	A5	A6	AB
p	400	1100	2150	300	150	220	0	0	0	0	0	63
b	200	1100	2150	60	110	130	0	0	0	0	0	63
t	400	1600	2150	300	120	250	0	30	45	57	63	0
d	200	1600	2600	60	100	170	0	47	60	62	60	0
k	300	1990	2850	250	160	330	0	53	43	45	45	0
g	200	1990	2850	60	150	280	0	53	43	45	45	0

Table 22 Parameter values for plosive consonants.

Nasal	FNP	FNZ	F1	F2	F3	B1	B2	B3
m	270	450	480	1270	2130	40	200	200
n	270	450	480	1340	2470	40	300	300

Table 23 Parameter values for nasal consonants.

Phon.	F1	F2	F3	B1	B2	B3	F0
'AA'	618	1010	2583	130	248	451	120
'AE'	705	1825	2602	138	144	78	113
'AH'	681	1271	2562	146	265	248	124
'AO'	475	709	2710	149	30	661	116
'AW'	682	1107	2430	219	204	256	119
'AY'	599	1383	2461	121	435	435	123
'EH'	469	2323	2830	48	428	271	119
'ER'	451	1511	2654	66	67	84	117
'EY'	475	2135	2835	121	556	599	138
'IH'	370	2165	2750	26	377	346	126
'IY'	300	2045	2960	45	200	400	115
'OW'	555	1275	2663	120	66	93	113
'OY'	473	1400	2513	105	105	260	113
'UH'	413	1025	2200	65	110	140	113
'UW'	365	1475	3185	52	133	547	114
AVG.	**508.7**	**1510.6**	**2662.5**	**103.4**	**224.5**	**324.6**	**118.9**

Table 24 Record of measurements taken from human speech output, of the average formant peaks and fundamental frequencies.

Phon.	F1 (Hz)	F2 (Hz)	F3 (Hz)	B1 (Hz)	B2 (Hz)	B3 (Hz)	F0 (Hz)	AV (dB)
'AA'	680	1059	3220	59	540	463	127	68.6
'AE'	754	2186	2843	1309	1006	392	117	70.4
'AH'	744	1564	3082	62	827	374	134	63
'AO'	564	857	3242	74	61	96	119	74.6
'AW'	775	1624	2929	65	747	696	127	61.6
'AY'	706	1683	3317	77	607	496	135	65.6
'EH'	485	2455	3304	15	213	134	122	66.5
'ER'	551	1670	2976	180	363	292	118	64.3
'EY'	606	2355	3323	71	327	150	145	59.9
'IH'	433	2217	3143	95	1506	340	124	62.8
'IY'	481	2305	3315	162	278	291	131	64.3
'OW'	753	1606	2966	449	888	458	119	61.2
'OY'	564	1234	3080	344	652	569	118	65.8
'UH'	576	1539	2603	205	1217	981	119	67.6
'UW'	769	2159	3409	294	70	111	116	65.4
AVG.	**629.4**	**1768**	**3117**	**230.7**	**620.1**	**389.5**	**124.7**	**63**

Table 25 Record of measurements taken from the synthesis subsystem, of the average formant peaks, bandwidths, fundamental frequencies, and intensities.

Phon.	F1 error (%)	F2 error (%)	F3 error (%)
'AA'	10,03236	4.851485	24.66125
'AE'	6.950355	19.78082	9.262106
'AH'	9.251101	23.05271	20.29664
'AO'	18.73684	20.87447	19.631
'AW'	13.63636	46.7028	20.53498
'AY'	17.86311	21.69197	34.78261
'EH'	3.411514	5.682307	16.74912
'ER'	22.17295	10.52283	12.13263
'EY'	27.57895	10.30445	17.2134
'IH'	17.02703	2.401848	14.29091
'IY'	60.33333	12.71394	11.99324
'OW'	35.67568	25.96078	11.37814
'OY'	19.2389	11.8571	22.56267
'UH'	39.4673	50.1463	18.31818
'UW'	110.6849	46.37288	7.032967
AVG.	**10.03236**	**20.85826**	**17.385**

Table 26 Statistical analysis to obtain average estimated error for formants.