Escola Universitària d'Enginyeria Tècnica Industrial de Terrassa
Universitat Politècnica de Catalunya

Enginyeria Tècnica de Telecomunicacions (so i imatge)

Generació de veu sintètica cantada

Beatriz Blázquez Bárcena

Director: Ignasi Esquerra

Gener 2006

ÍNDEX

Capítol 1.

Introducció.

1.1. Tecnología TTS (Text To Speech).

Avui dia hi ha una progressiva proliferació d'aplicacions basades en el procès lingüístic automàtic del llenguatge parlat. Així doncs, són cada cop més comuns:

- les interfaces home-màquina controlades per veu
- els sistemes de resposta vocal interactiva
- l'automatització de sistemes telefònics

L'elevat nombre d'aplicacions possibles queda condicionat, a més de per els propis 'factors humans' que entren en joc al procès d'aceptació d'una nova tecnologia, per els propis de la tecnologia subjacent. Aquesta tecnologia rep el nom comú de Tecnologia de la Parla i s'estructura en quatre tecnologies bàsiques principals:

- El Reconeixement de Veu o Reconeixement de la Parla

És el procès de conversió d'un missatge parlat a text, que permet a l'usuari una comunicació amb la màquina. Es tracta de la tecnologia que major avanç ha experimentat als últims anys, passant de poder reconèixer tan sols a un parlant, dins un vocabulari limitat, a prototips que poden reconèixer a qualsevol parlant sobre vocabularis flexibles de milers de paraules.

- La Conversió Text-a-Veu

S'ocupa de la generació de missatges parlats mitjançant la simulació del procès de lectura d'un text escrit emmagatzemat en format electrònic. Aquesta tecnologia també compta actualment amb uns resultats excelents, plantejant-se com a repte major, augmentar la qualitat de la veu sintetitzada.

- El Reconeixement de Locutors

És el procès d'identificació o verificació de la identitat del parlant de forma automàtica a partir del senyal de veu. El grau de desenvolupament d'aquesta tecnologia és inferior al de les anteriors, potser com a conseqüència de la dificultat de les aplicacions on s'inserti.

- La Codificació de Veu

És una tecnologia que ha arribat a un grau de maduressa molt elevat, contant a l'actualitat amb un nombre important de procediments estandaritzats. El seu objectiu és la búsqueda de

representacions eficients en format digital del senyal de veu pel seu emmagatzemament i/o transmissió, buscant obtenir la major qualitat possible, pel menor nombre de bits per mostra.

Podríem, per tant, situar a la Tecnologia de la Parla com a receptora d'un ampli conjunt de coneixements i procediments d'actuació sobre l'informació representada al senyal de veu. Coneixements que s'articulen amb un alt grau de dificultat i especialització, ja que pertanyen a un marc científic-tècnic multidisciplinar, on es donen cita diferents branques del saber com: fisiologia, acústica, lingüística, processat del senyal, intel·ligència artificial, teoria de la comunicació i de la informació, i ciència de la computació.

1.2. Aplicacions de la Tecnologia de la Parla.

Fent ús de les tecnologies bàsiques es fa possible la comunicació de l'home amb les màquines utilitzant el llenguatge parlat, aconseguint un nombre de possibles aplicacions de la tecnologia de la parla enorme. Únicament les actuals limitacions tecnològiques i la imaginació per concebir aplicacions, on no s'ha d'oblidar el factor humà (representat per la necessitat d'aconseguir una amigabilitat que suposi el mínim risc d'aceptació per part dels usuaris, donada la novetat d'aquestes tecnologies) poden limitar el nombre d'aplicacions.

Als últims cins anys s'ha produït un notable avanç que fa possible disposar d'una tecnologia bàsica capaç de soportar aplicacions i serveis comercials. Avui dia, podem dir que la tecnologia de codificació de veu és una tecnologia madura i consolidada, i que la conversió text-veu ha arribat a un nivell suficient de maduressa, per la qual cosa el seu repte actual estriba en aconseguir una naturalitat equiparable a la veu humana, és a dir, que sigui indistingible d'un locutor. En reconeixement de veu, s'ha aconseguit reconeixedors que, tot i que són limitats en quant a tamany de vocabulari, posseixen una qualitat suficient per soportar un gran nombre d'aplicacions. Com productes comercials, estan disponibles reconeixedors de dígits aïllats i concatenats, i reconeixedors de paraula aïllada amb vocabularis de milers de paraules i, el més important, és possible definir el vocabulari del reconeixedor sense necessitat de realitzar un llarg i costós procès d'entreteniment (reconeixedor de vocabulari flexible). En fase precompetitiva (prototips de laboratori) existeixen reconeixedors de parla conectada que poden manegar vocabularis d'alguns milers de paraules. El reconeixement de locutor és actualment la tecnologia bàsica més endarrerida, tot i que ha començat a utilitzar-se amb éxit a algunes aplicacions de verificació de locutor com a reforç de tècniques clàssiques de verificació.
Als pròxims anys, s'espera un notable avanç de la tecnologia bàsica, especialment al reconeixement de veu, amb el que es superaràn moltes de les actuals restriccions. A aquest avanç ja de contribuir de forma decisiba l'evolució en microelectrònica que permetrà disposar d'una potència de càlcul i d'una quantitat de memòria molt superior a les utilitzades fins ara. D'això es beneficiarà tant l'investigació com el propi producte comercial que podrà incorporar els últims avenços aconseguits al laboratori, a uns costos raonables.

Com qualsevol tecnologia innovadora, és important que es destini a aplicacions útils, de manera que les persones que facin ús d'aquestes aplicacions obtinguin un benefici clar. Per tant, han de cuidar-se al màxim els detalls que facin còmode i agradable el diàleg amb els usuaris, ja que d'aquest diàleg depen l'aceptació (o el rebuig) d'una determinada aplicació i, per extensió, de tota la tecnologia que involucra. En aquest sentit el estudis sobre els Factors Humans als serveis de telecomunicació han de contribuïr de forma important a paliar les limitacions actuals de la tecnologia de la parla.

Donat l'ampli camp de l'aplicació de la Tecnologia de la Parla, s'ha considerat necessari classificar les seves aplicacions en tres grups diferents, cadascún dels quals impossa una sèrie de requisits comuns, a la vegada dòna lloc a conjunts d'aplicacions semblants. Aquests grups són:

- Aplicacions locals
- Resposta vocal interactiva
- Automatització de sistemes telefònics

1.3. Objectius del projecte.

L'objectiu d'aquest projecte és desenvolupar els mòduls necessaris per aconseguir generar veu sintética cantada. Utilitzant com a entrades un fitxer tipus MIDI (Interfaç digital per a instruments musicals) amb la música i un fitxer de text amb la lletra, per tal de generar un nou fitxer d'entrada adequat a un conversor de text a veu ja existent.

Les tasques a desenvolupar són diverses, en primer lloc realitzar un estudi de les tecnologies de la parla i revisar projectes o articles anteriors relacionats amb el tema.

Hauré d'escollir un conversor text a veu que será el que generarà la veu cantada, la elecció la faré entre els conversors que trobi de manera lliure, per tant, de baixa qualitat. Un dels factors més decissius per a la elecció serà que sigui compatible amb Windows, ja que serà l'entorn que faré servir per dur a terme el projecte.

Per realitzar les funcions necessàries per adaptar els diferents tipus de fitxers que faré servir he escollit la programació en C, ja que és la que hem estudiat a la carrera en un parell d'assignatures comuns.

Per últim es podria realitzar una interfície gràfica per tal d'agrupar totes les funcions realitzades, de manera que escollint una cançó a l'entrada és generessin tots els arxius intermitjos perquè el conversor la cantés.

Capítol 2.

Conversió Text a Veu.

2.1. Introducció.

La Conversió text a veu és el procès mitjançant el qual un sistema automàtic transforma una seqüència de caràcters, és a dir, un text genèric, en un señal de veu. Els processos que apareixen es poden classificar en els que realitzen processat del llenguatge i els que realitzen processat del senyal. A la primera classe está la normalització del text d'entrada en paraules convencionals, eliminant abreviatures, nombres, signes, etc., la determinació dels grups entonatius, la transcripció fonètica, i els models prosódics. A la segona classe están els procediments que generen veu, per exemple, mitjançant concatenació de segments. A aquests procediments es seleccionen segments pregrabats d'una base de dades, es modifiquen les característiques prosódiques perque s'ajustin a l'indicat pel modelat prosòdic i es concatenen mitjançant alguna tècnica que minimitzi la distorsió per concatenació. Al desenvolupament d'un sistema de conversió de text a veu intervenen disciplines molt diverses, que han d'enfrentar-se a problemes complexes, molts d'ells encara sense resoldre.

Un Conversor text a veu general es pot representar mitjançant una divisió en blocs, on es pot apreciar aquesta dualitat. Per un costat, apareix *un bloc de procés lingüístic* i per altre banda *un bloc de síntesi de veu*.

Finalment s'han d'integrar aquestes tasques en una estructura que faci possible el funcionament del conversor en diferents entorns, i la seva utilització en diferents aplicacions.

2.2. Principals esquemes de síntesi de veu.

Existeixen varies clasificacions possibles dels sistemes de síntesi de veu, però la més comú és la referida al tipus de codificació del senyal de veu i al nombre de regles que es necessiten per la seva reconstrucció posterior. Així doncs podem distingir:

- Sintetitzadors articulatoris.

Als sintetitzadors articulatoris l'objectiu és el de controlar un model de l'aparell fonador, de forma semblant a com ho fa el cervell, constituïnt els paràmtres de control (paràmetres circuitals) de dit model, la posició dels diferents órgans articulatoris i les lleis que regeixen el seu moviment.

Aquests sintetitzadors presenten la dificultat de l'obtenció i control de paràmetres pel seu maneig (dificultat a l'anàlisi de la posició i moviment dels òrgans articulatoris d'una persona que

parla normalment, així com de cara a la coordinació i derivació de la gran quantitat de paràmetres de control, existents a l'entrada del sintetitzador), el que ha motivat que siguin actualment els menys desenvolupats.

- Sintetitzadors per formants.

Estàn constituïts per una sèrie de filtres que modelen les resonàncies (formants) del tracte vocal, i que estan excitats per un sistema de fonts que modelen tant la vibració de les cordes vocals com el soroll que es produeix a la fricació. El principal avantatge que ofereixen aquests sistemes és que treballen amb paràmentres que mantenen una relació directa amb el mecanisme de producció de la parla, i són facilment manipulables de cara al control del sintetitzador. En funció del control que es faci dels paràmetres, es pot establir una subdivisió entre:

- Sistemes de síntesi per regla.
  Els paràmetres es calculen amb un conjunt de regles depenents del context (un fonema d'una classe precedit per un altre d'una altre classe presenta un espectre d'una forma determinada).
- Sistemes d'anàlisi-síntesi.
  Els paràmetres s'obtenen per anàlisi o parametrització de segments de veu natural.

Els sintetitzadors de formants gaudeixen de gran difusió per l'atractiu que presenten per estudis fonètics.

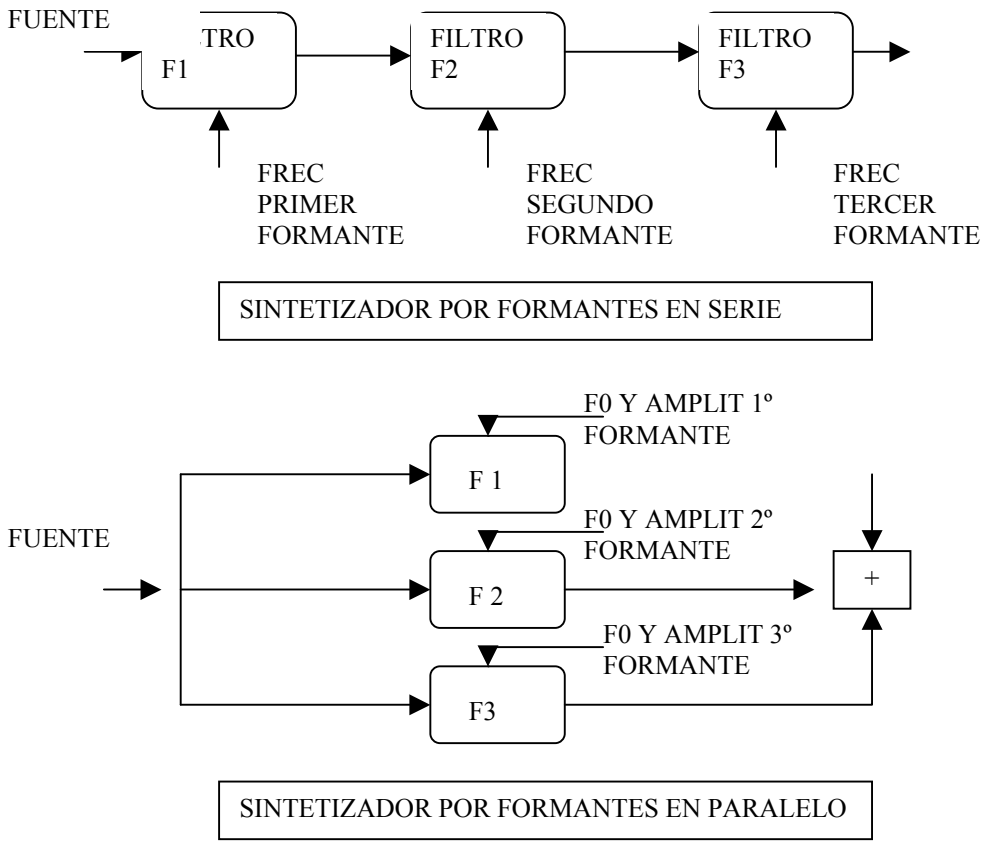


Figura 1. Esquema bàsic dels sintetitzadors per formants.

- Sintetitzadors derivats de les tècniques de pedicció linelal (LPC).

Es basen en la possibilitat de modelar el tracte vocal com una sèrie de cilindres buits de diàmetre variable. Les propietats de l'ona sonora, després d'haver passat per tots ells, poden ser predites tenint en compte que cadascun dels cilindres condiciona la forma de l'ona sonora a l'entrar al següent. Aquest càlcul complexe es simplifica mitjançant la utilització de la predicció lineal, i aprofitant la periodicitat de l'ona sonora.

Existeixen diferents classes segons el mètode concret de codificació utilitzat, tot i que els més extensos avui dia són els sintetitzadors multipols (MLPC). Com a característiques generals es podrien citar la baixa complexitat i la seva bona qualitat.

- Sintetitzadors per concatenació de forma d'ona.

En aquests sintetitzadors s'intenta augmentar la qualitat del senyal generat mitjançant una minimització del soroll de codificació, per aconseguir-ho es concatenen unitats digitalitzades (pregravades) i s'ajusta la seva prosòdia original a la de la nova frase. Podem distingir entre:

- Sintetitzadors basats en mètodes PSOLA (Pitch-Synchronus Overlap-Add).
- Sistemes basats en Codificació Armónica.
- Codificadors Multibanda.

A canvi de la seva elevada complexitat, aquests són els sistemes que ens ofereixen una major qualitat.

Com un cas especial d'aquest tipus de sintetitzadors, es troben els sintetitzadors per selecció. Aquests sistemes es basen en la selecció de les unitats a concatenar de cara a la síntesi en funció de les seves característiques prosódiques. En aquest tipus de sintetitzadors , a diferència dels casos anteriors no es realitza cap tipus de codificació ni de modificació prosódica, concatenant-se directament les unitats pregravades.

2.3. Divisió en blocs.

Als apartats següents descriurem les activitats que es realitzen per les diferents línies de treball en conversió text a veu presentades anteriorment.

2.3.1. Bloc de processat lingüístic.

Per llegir un text hem de saber els sons que s'han de produïr, i com s'han de produïr. Aquests són els objectius d'aquest bloc, és a dir, obtenir la cadena d'al·lòfons corresponent al text d'entrada i la informació de prosòdia referent a la producció dels mateixos: duració dels al·lòfons i evolució temporal de l'entonació o freqüència fonamental (Fo) al llarg del discurs.

La tasca de generar tota aquesta informació, per un text sense restriccions, és massa complexa per abastir-la globalment. Per tant és necessari dividir el problema en una sèrie de problemes més sencills.

- La primera fragmentació és la del text d'entrada, ja que no sería pràctic tractar tot el text d'entrada d'un cop (en principi il·limitat). S'ha escollit una unitat de treball per fraccionar-ho, reduïnt al mínim la relació (dins dels paràmetres del conversor) entre unitats de treball consecutives. Aquesta unitat de treball és la frase. Per al conversor dos frases consecutives són pràcticament independents.
- La segona fragmentació és la de la tasca que s'ha de realitzar. La tasca d'obtenir la seqüència de sons a generar, i la seva entonació, s'ha dividit en tasques més sencilles, que s'executen seqüencialment. Aquestes subtasques es realitzen en diferents mòduls.

A continuació, descriurem cada mòdul que forma el procès lingüístic del conversor text-veu, presentant les tasques que realitza.

2.3.1.1. Mòdul normalitzador.

La seva tasca principal és detectar i reunir un conjunt de caràcters al text d'entrada. Aquest conjunt de caràcters forma la unitat de treball que es prendrà com comú a tots els mòduls.

La unitat de treball que s'ha definit rep el nom de frase, tot i que no és el que s'entèn generalment per frase, serà una sèrie de paraules fins que es trobi amb algo que marqui el fo de frase (punt de fi de frase, signe d'interrogació, dos punts, etc).

Utilitzem el terme de paraula per referir-nos a les agrupacions de caràcters de la frase, separats per blancs, tabulats i noves línies, tot i que no sempre són paraules propiament dites (formades únicament per signes alfabétics), ja que podem trobar signes ortogràfics, abreviatures, etc.

Com ja hem dit abans, la tasca principal del mòdul normalitzador és aïllar una frase del text d'entrada per poder pasar-la a la resta dels mòduls com unitat de treball. A més, realitza una sèrie de tasques complementàries (Figura 2).

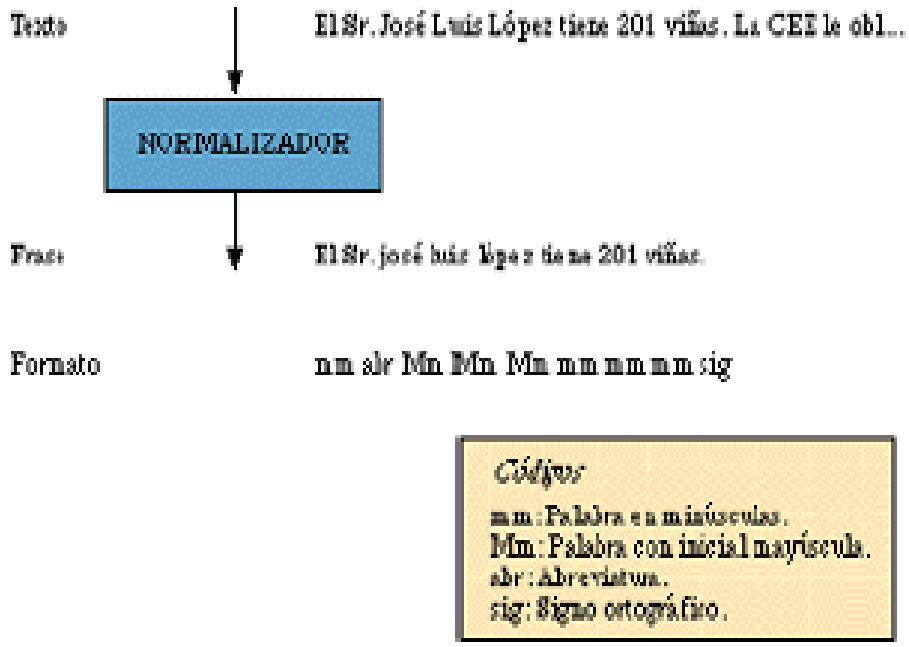


Figura 2. Exemple del tractament del mòdul normalitzador.

- Normalitzar l'escriptura de les paraules detectades, per simplificar comparacions i consultes a la resta del mòduls i als mòduls següents. La normalització soposa varies coses. Per una banda, s'emmagatzemen les diferents paraules que formen la frase separant unes de les altres per un únic espai en blanc. Per altre banda, es modifica l'escriptura de les paraules de la frase, pasant tots el caràcters a lletres minúscules. A més, també es modifica la representació de les seqüències de signes ortogràfics, per als casos de signes difícils de reconèixer tal i com apareixen normalment (punts suspensius) o que presenten alguna ambigüetat en la seva interpretació (les cometes i els guions de comentari, que poden obrir o tancar). Amb aquest canvi de representació solucionem l'ambigüetat.
- Obtenir dades sobre el tipus de paraula al que pertany cadascuna de les paraules de la frase. Això permetrà en altres mòduls donar un tractament diferent a les paraules, segons el tipus que pertany (per exemple, paraula, abreviatura, signe ortogràfic, etc).

El normalitzador comença per reunir els caràcters que formen la frase. La seva principal ocupació es decidir quan es produeix qualsevol de les circunstàncies que provoquen un fi de frase.

2.3.1.2. Mòdul de pre-processament.

Tot i que el mòdul normalitzador ja ha decidit i agrupat el que és una frase, separant els elements que la constitueixen i realitzant una primera normalització de la seva escriptura, la variabilitat del text d'entrada és encara massa gran. La resta de mòduls només podràn manegar paraules i signes ortogràfics, i s'han de reduïr a aquesta representació qualsevol altre cosa (abreviatures, representacions de dates, representacions horàries, etc).

Per tant la principal tasca d'aquest mòdul és la de reduïr la complexitat (variabilitat) del text (Figura 3). A més, es realitzen altres tasques:

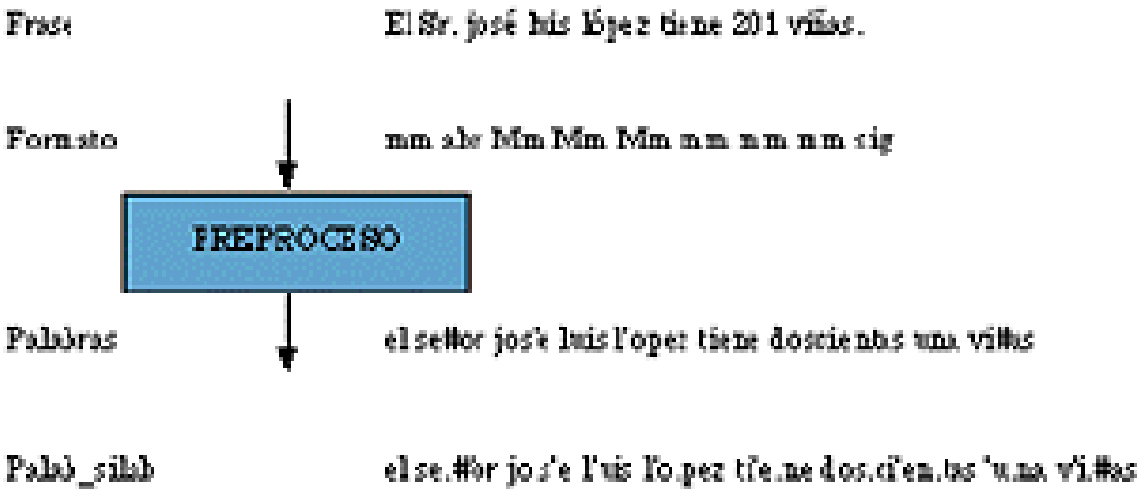


Figura 3. Exemple del tractament del mòdul de pre-processament.

- Silabicació
  La informació sobre la divisió de síl·labes de la paraula és necessaria per poder determinar l'accentuació fonètica. A més, influeix en la decisió dels al·lófons, controla les regles de entonació i pausa, e indica com s'han de tractar les paraules il·legibles (sigles o paraules no vàlides en castellà).La Academia de la Llengua proporciona unes regles sobre com s'ha de realitzar aquesta tasca, tot i que alguns casos no queden totalment determinats. El conversor emplea aquesta regla, tot i que s'han completat amb altres adaptades a aplicacions específiques del conversor.

- Accentuació fonètica
  Al castellà la major part de les paraules presenten un únic accent fonétic, i la seva posició queda determinada per la grafía de la paraula i la seva divisió en síl·labes.

  Normalment una paraula tindrà un accent, però no sempre és així.

Hi ha paraules que presenten dos accents. Són els advervis procedents d'adjectius acabats en ment. Un accent recau sobre l'arrel de la paraula que correspon a l'adjectiu, i l'altre a la primera "e" de "-ment".

També hi ha paraules inaccentuades, són paraules funció (preposicions, articles determinats, etc.), sense contingut semàntic per si mateixes. Però hi ha paraules amb la mateixa grafía que aquestes paraules funció però són verbs, noms, etc., i que per tant si han de portar accent. Aquest mòdul només les pot marcar com ambigües i hem d'esperar a que es realitzi l'anàlisi de categories per decidir l'accentuacció correcta.

Hi ha altres particularitats de l'accentuacció que es tindrán en compte en altres mòduls, com és la supresió de l'accent en algunes de les paraules components d'una quantitat numèrica o evitar que hi hagi paraules àtones abans d'una pausa.

- Tractament d'acrònims i seqüències impronunciables
  Quan s'ha de llegir una seqüència de lletres sense suficients vocals per pronunciar-la fàcilment, normalment es deletreixen. Per exemple: ftp es lleguiría efe te pe, HB hache be, frnos segurament sería efe erre nos.

  Per tractar aquests casos es fa una segona silabicació més ajustada i després es deletreixen les síl·labes que queden sense vocal. A més es valora la forma com estaba escrita la paraula i un conjunt d'excepcions que l'usuari pot adaptar a les seves necessitats.

Una de les majors dificultats que presenten les tasques d'aquest mòdul és la falta d'una normativa, i molts cops ni d'un consens per tractar tots els fenòmens de paraules que poden apareixer en un text real. La Real Academia proporciona normes i directrius per paraules en castellà, però és molt poc dinàmica per la incorporació de nous termes (sobre tot extranjerismes), o per donar normes realistes sobre el seu tractament.

Per al conversor ha estat necessari desenvolupar un conjunt de regles heurístiques que tractin aquestes situacions, així com proporcionar mecanismes per a que l'usuari pugui adaptar el funcionament del mòdul a les pecularietats del text a tractar a la seva aplicació.

2.3.1.3. Mòdul categoritzador.

La tasca principal d'aquest mòdul és la d'assignar categories a les paraules amb l'objectiu de que el conversor faci una lectura "amb sentit" dels texts (afegint, per exemple, pauses que no venien marcades ortogràficament). Les categories que s'assignen no són exactament categories gramaticals, sino un conjunt de codis que en molts casos es corresponen amb categories gramaticals reals, descobrir l'estructura sintàctica és encara una labor massa complexa si no es posen restriccions a la gramàtica.

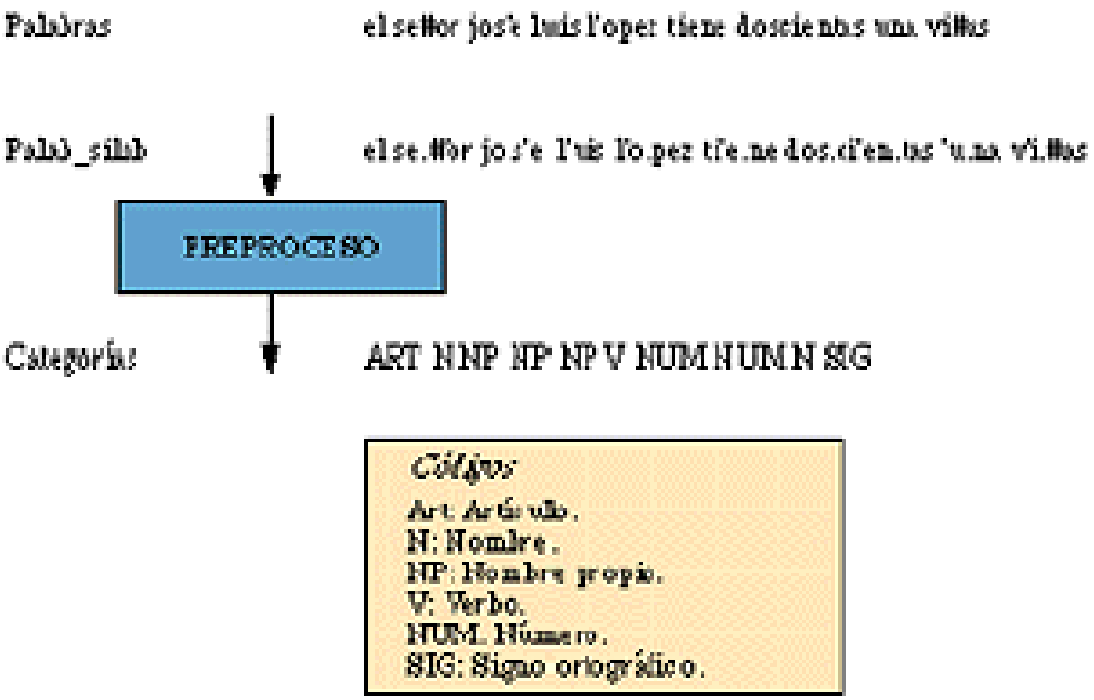


Figura 4. Exemple del tractament del mòdul catgoritzador.

Hi ha parts de la catgorització que ja s'han realitzat al mòdul de pre-processament, com és el cas de les paraules amb la inicial en majúscules sense ser principi de frase (noms propis), els resultats de les expansions (nombres, lletres, signes) i les paraules acabades en "-ment".

Per altre banda, amb el resultat de la caategorització es corretgeix en alguns casos l'accentuació, a les formes homógrafes on la presència d'accent depèn de la categoría a la que pertanyen (com el cas de 'sobre', que pot ser una preposició, un verb o un sustantiu. Només al primer cas és àtona).

La tasca de la categorització es pot dividir en tres tipus de subtasques: rutines que busquen en taules la paraula a categoritzar, rutines que comproben la estructura de la paraula per intentar descobrir la categoría, i rutines que comproben la relació de la paraula en qüestió amb altres paralules del seu entorn.

2.3.1.4. Mòdul estructurador.

La seva misió es la de realitzar un anàlisi sintàctic de les frases (Figura 5). Fins ara no s'han realitzat els estudis necessàris per poder aprofitar la informació generada, i és la informació de les categories gramaticals la que s'utilitza directament per decidir la inserció de pauses no marcades ortogràficament i determinar la prosòdia.

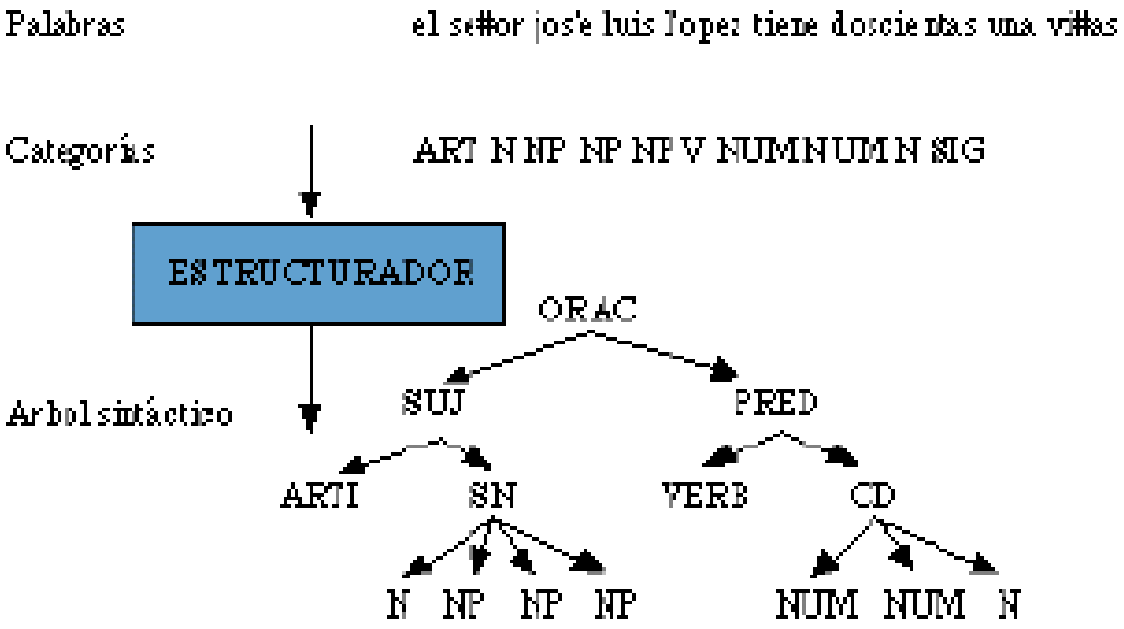


Figura 5. Exemple del tractament del mòdul estructurador.

Un primer problema és que tant la categorització com la inserció de pauses es realitza amb molt poc context (tres o quatre paraules com a molt a cada costat de la paraula a tractar), i per tant no es capta la estructura de la frase. No es poden esbrinar relacions entre elements de nivell superior.

Un altre problema és que per generar la prosòdia només s'utilitza la informació disponible al cas en que es decideixi fer una pausa. Si no, la informació sobre la estructura, que el mòdul pausador detectaría, desapareix.

Per una prosòdia natural sería necessari que el sistema tingués informació del significat de la frase. Actualment no es pot realitzar anàlisi semàntic i pragmàtic de text fora d'entorns molt restringits. Per això s'utilitza la estructura sintàctica com un reflexe de la estructura semàntica de la frase. Per fer això primer es necessita un mòdul que generi aquesta estructura sintàctica.

Tot i que l'anàlisi sintàctic del text és un problema abordable, el plantejament per aquest mòdul és més modest, per tres raons: és molt difícil definir una gramàtica que pugui tractar qualsevol tipus de text, el mòdul no podía ser molt complex, i no està massa clar el nivell d'exactitud i precisió necessàri per una tasca com la que es pretén.

La primera idea va ser la de fer agrupacions de paraules que formasin unitats sencilles, i després intentar fer que aquestes unitats s'agrupassin en un segon nivell, i així fins agrupar tot al nivell de frase.

El mòdul utilitza un sistema de regles, ja que no es disponía ni de coneixements per formalitzar les relacions gramaticals en un sistema expert o similar, ni de suficient text analitzat per desenvolupar i entrenar un sistema d'autoaprenetatge, o un sistema estocàstic.

Per realitzar l'agrupació de paraules es pot utilitzar una estrategia de dalt a baix (descompossant cada unitat en les de nivell inferior, partint de la frase), o de baix a dalt (integrant les unitats en altres d'un nivell superior, partint de la paraula). Tot i que la segona és més sencilla arriba un moment en que no s'esbrina el mode per continuar l'agrupació. Per altre banda, qualsevol de les dos no resol correctament determinades situacions, com decidir si una conjunció copulativa forma part d'una enumeració, o està coordinant dos proposicions, i a que nivell en aquest cas.

Finalment, es va utilitzar una estrategia mixta, les primeres tasques de baix a dalt i després s'intenta esbrinar la estructura principal de la frase de dalt a baix. Aquesta tasca és el nucli de l'anàlisi, ja que les primeres no aporten res, només simplifiquen les tasques posteriors.

En aquest procès apareixen fenòmens que no considerats, com la presència de text entre cometes que poden formar frases completes, proposicions subordinades, sintagmes, o frases sense una estructura gramatical completa.

L'anàlisi sintàctic está molt lligat a l'anàlisi per categories, aconseguint evitar els errors de categorització amb un estudi conjunt, mitjançant l'anàlisi de les estructures resultants per les diverses hipótesi. Però al haber de manegar un conjunt de categories per paraula i generar estructures per totes les combinacions possibles, es complica el seu disseny.

2.3.1.5. Mòdul pausador.

Al lleguir un text els lectors fan pauses, marcades per signes ortogràfics. Aquest fet és per la necessitat de recuperar l'alè. A més, la inserció de pauses i la utilització de la prosòdia fan possible trasmetre part del contingut del missatge i facilitar la comprensió a l'oient.

Evidentment, un conversor text-veu no necessita recuperar l'alè, però provocaria una sensació d'ofec si pronuncies un text sense cap signe ortogràfic.

La conversió grafema a a·lófon es fa mitjançant regles, algunes de les quals consulten la existència de pausa davant o darrere de la lletra estudiada. Així la conversió depén en alguns casos de les pauses, marcades ortograficament o no.

El pausador ajuda a la genració d'una prosòdia més correcta, i així pot semblar que el conversor entén el que està lleguint, aumentant la naturalitat de la veu sintètica que rep l'oient.

La tasca principal és insertar pauses automàticament, quan el text es troba entre dos pauses marcades ortografiicament generaria una seqüència massa llarga. A més corretgeix l'accentuació fonètica de les paraules seguint criteris prosódics. Per exemple, les paraules àtones davant una pausa no es pronuncien com a àtones, per tant sempre que hi ha una pausa (ortogràfica o no) es comproba si la paraula anterior és àtona, accentuant-la fonèticament i es marcaria com a tònica.

Després de realitzar les pauses marcades ortogràficament, si els fragments de les frases resultants són massa llargs, es busquen les millors posicions per realitzar noves pauses, d'acord a les categories assignades a les paraules. L'orde de prioritat és el següent:

- Conjuncions coordinants.
- Conjuncions subordinants.
- Verbs.
- Paraules funció.

Cada categoria contempla excepcions que no deixen introduïr pausa, només en el cas en que no es pugui introduïr pausa per una categoria es passa a la següent de prioritat més baixa.

2.3.1.6. Mòdul conversor de grafemes a al.lófons.

Aquest mòdul és el que s'encarrega de determinar quina és la seqüència d'al·lófons que correspon a la seqüència de lletres d'una frase. Utilitza la informació lingüística dels mòduls anteriors. S'ha de tenir en compte la correspondència entre lletres (grafemes) i sons (al·lófons) no sempre la mateixa lletra produeix el mateix al·lófon, sino que la conversa depén d'una sèrie de regles.

La informació utilitzada a les regles de conversió és:

- Caràcters (lletres) components de la frase.
- Límits entre paraules dins la frase.
- Límits entre sílabes dins la paraula.
- Localització de les pauses.

Si la silabicació ha estat correcta, amb aquestes normes s'obtenen transcripcions fonètiques académicament correctes.

2.3.1.7. Mòdul generador de paràmetres acústics.

La tasca d'aquest mòdul és la de generar els paràmetres que determinen la prosòdia (Figura 6). Aquesta tasca es pot dividir en dos, una per cada paràmetre: durada i entonació (contorn de la freqüència fonamental).

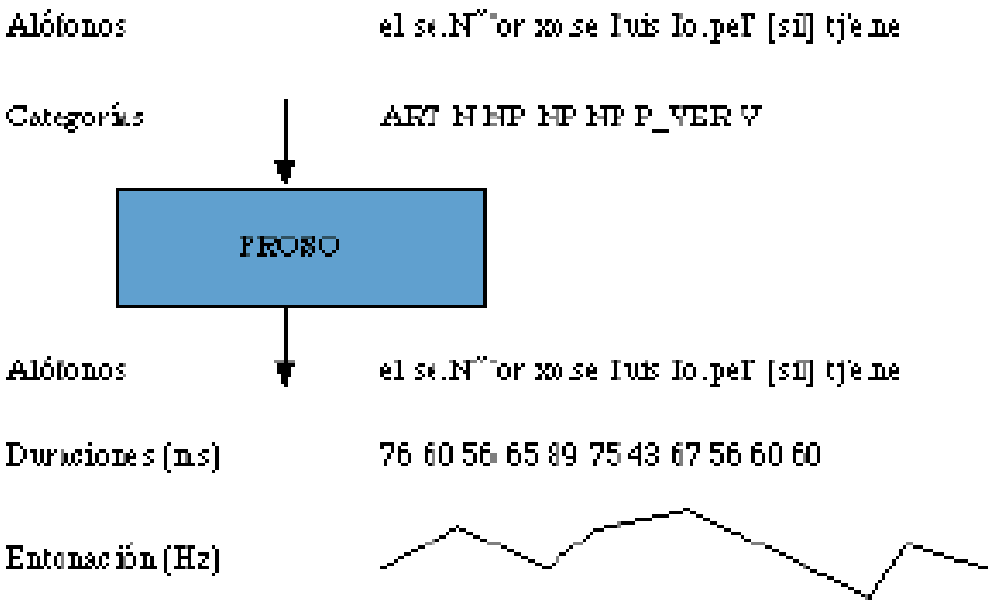


Figura 6. Exemple del tractament del mòdulo generador de paràmetres acústics.

S'assignen els valors de durada a les pauses (tant les ortogràfiques com les generades al mòdul pausador) i els al·lòfons.

La entonació és l'altre paràmetre, i és el de més influència en la qualitat i la naturalitat de la veu. Aquest mòdul genera valors de freqüència fonamental (F0), mentre que entonació és un terme relacionat amb la fonologia i la percepció. Però utilitzarem el terme entonació per fer referència a la freqüència fonamental, que és un paràmetre acústic.

El bloc de procés de senyal necessita rebre informació sobre el to amb un mostreig uniforme. Per generar aquests valors s'utilitza un model que assigna els valors de F0 a determinats punts

de les síl·labes (normalment als punts mitjos dels nuclis de les síl·labes tòniques i algunes àtones) i després interpola segons els valors de durada.

2.3.2. Bloc de síntesi de veu.

La misió del bloc de síntesi de veu és la de generar sons tan similars a la veu com sigui possible, presentant un alt grau de flexibilitat en quant a les seves capacitats per ser controlat, de tal manera que es pugui variar la realització dels sons.

La informació d'entrada a aquest bloc inclou la seqüència d'al·lófons que s'han de generar, i les dades de prosòdia (típicament, duració dels al·lófons, contorn de freqüència fonamental, i contorn d'energia o amplitud).

Hi ha dos enfocs que, d'alguna manera, determinen el tipus de sintetitzador que s'utilitza. El primer d'ells, que podríem denominar <<model de sistema>>, intenta modelar, amb major o menor detall, el mecanisme de producció de la veu. Aquest enfoc ha donat lloc a dos tipus de sintetitzadors: els *sintetitzadors articulatoris* i els *sintetitzadors de formants*. El segon enfoc, que podríem anomenar <<model de senyal>>, és el que intenta modelar no el mecanisme de producció , sino el senyal de veu; dins aquest enfoc es troben els *sintetitzadors per concatenació*. Aquests es basen en tenir un conjunt de petits segments de veu d'un parlant, que es concatenen per formar el discurs desitjat. La concatenació ha de ser controlada per evitar discontinuitats i sons anòmals, i per poder variar els elements prosòdics respecte a aquells que originalment tenien segments de veu escollits. Per decidir el tamany i el nombre d'unitats hi ha un compromís entre qualitat de veu i les limitacions de la memòria de dades.

El procediment de síntesi comprèn tres etapes diferenciades (Figura 7):

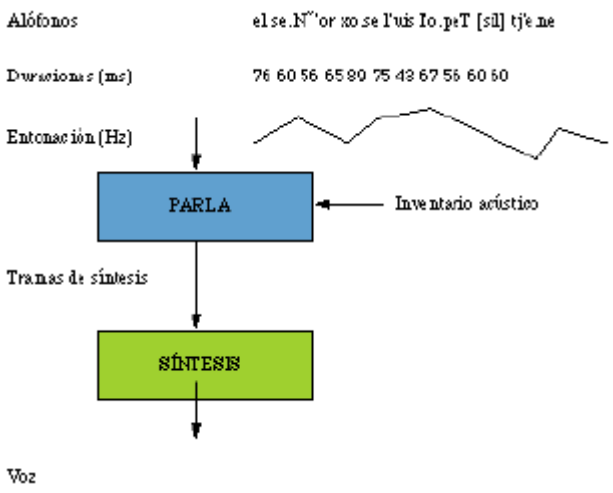


Figura 7. Exemple del tractament del bloc de síntesi.

- Escollir al seqüència d'unitats de la base de dades que correspon amb la seqüència d'al·lófons a sintetitzar, ja que segons la composició i els tipus d'unitats poden haber seqüències diferents per una determinada seqüència d'al·lófons.

- Ajustar el nombre de trames de cada al·lófon a la durada que s'hagi assignat, s'ha de tenir en compte les trames de l'al·lófon corresponents a dos unitats diferents, quan l'al·lófon no es trobi a la posició intermitja d'una unitat.

- Producció de les mostres de veu. Un cop ajustat el nombre se trames de síntesi per tots els al·lófons de la frase, es passen a l'algoritme que produeix les mostres de veu, però impossant els valors de F0 calculats al mòdul generador dels paràmetres prosódics.

Capítol 3.

El sistema MBROLA.

En aquest capítol es fa un breu resum del sistema de síntesi escollit per realitzar el projecte. Les raons de la seva elecció són varies, tenia altres opcions però hem vaig decantar per aquesta perquè hem va semblar que tenia una interfície senzilla d'entendre, un format de fitxers senzill per ser manipulats i perquè disposa de base de dades en castellà.

3.1. Introducció.

El projecte de MBROLA, iniciat pel laboratori de TCTS del Faculté Polytechnique de Mons (Bèlgica), té com a objectiu obtenir un sistema de sintetitzadors de parla per tants idiomes com sigui possible, i proporcionar-los lliurement per a usos no comercials. L'última meta és alçar la investigació acadèmica sobre síntesi de la parla, i particularment sobre la generació prosòdica, coneguda com un dels desafiaments més grans dels sintetitzadors TTS (Text a Veu).

El cor del projecte és MBROLA, un sintetitzador de parla basat en l'encadenament de difonemes. Pren una llista de fonemes com entrada, junt amb la informació prosòdica (durada dels fonemes i una descripció lineal per trossos), i produeix mostres de 16 bits (lineal), a la freqüència de mostreig de la base de dades utilitzada (per tant no és un sintetitzador de TTS, ja que no accepta el text com a entrada).

EULER és un projecte del grup d'investigació de la síntesi de la parla de la Facultat Politècnica de Mons (Bèlgica). El seu objectiu és proporcionar un dispositiu lliure, fàcil d'utilitzar i fàcil d'extendre, TTS genèric multilingüe que pugui integrar els resultats dels projectes existents.

Per què Euler?

Els laboratoris d'investigació privats i públics (des de universitats fins a operadors de telecomunicació) han invertit recursos en intentar dissenyar sintetitzadors multilingües. A la majoria dels casos, aquest esforç no coordinat de la investigació ha conduït a la incompatibilitat del sistema: a causa d'una manca d'eines i de bases de dades unificades, extensibles, i disponibles pel desenvolupament del sistema TTS, cada sintetitzador és una aplicació específica de laboratori de principis bàsics molt similars.

Això, a canvi, ha ocasionat la incompatibilitat del llenguatge: la majoria dels sistemes de TTS multilingües són merament recollides de monolingües independentment desenvolupats en laboratoris de recerca natius. Aquesta situació no només ha tingut impacte negatiu a l'extensibilitat de sistemes de TTS disponibles a llenguatges nous, dialectes, accents, veus, i estils de parla, sinó que també ha dificultat la seva integració a productes reals (per SMEs o empreses de telecomunicacions grans): en comptes d'incrementar un sintetitzador comú, d'ús

general i proporcionar-li aplicacions d'interfícies d'alta qualitat (per manegar documents de text complexes, per exemple), els laboratoris de investigació perden el temps. Finalment, però no per això menys important, la carència d'una espina dorsal comú per els sistemes de TTS ha fet molt difícil comparar la seva qualitat sobre una base de dades mòdul per mòdul, frenant les possibles millores.

Per una altre banda, les eines i les bases de dades avançades per el desenvolupament multilingüe del sistema de TTS ha estat fet recentment e independentment per diferents laboratoris de investigació de Europa, com per exemple:

- La Faculte Polytechnique de Mons (FPMs), SEA, ha pres recentment una mesura important cap al desenvolupament dels sintetitzadors d'alta qualitat, sintetitzadors multilingües de fonètica a parla, amb la forma del projecte MBrola. Aquest projecte vol fomentar col·laboracions internacionals per obtenir un sistema de sintetitzadors de discurs de MBrola per tants idiomes (dialectes inclosos) i veus com sigui possible.

- La Universitat de Edimburgo (UED), GB, ha fet recentment una contribució important pel desenvolupament lliure de la síntesi de la qualitat. El seu sistema lliure de síntesi de la parla, FESTIVAL, és un sistema genèric, modular, portable i extensible de TTS. El FESTIVAL s'ha desenvolupat amb una perspectiva multilingüe.

- La Universite de Provence (UP), FR, es coordinadora de la sèrie de projectes MULTEXT, l'objectiu dels quals és desenvolupar les eines, recopilacions i recursos lingüístics per una gran varietat d'idiomes. Totes aquestes eines i recursos són lliures i disponibles per propòsits no
  comercials.

Com a resultat de la disponibilitat d'aquestes eines, molts laboratoris han produït diversos mòduls/lèxics/bases de dades de TTS, però pocs els han fet públics. EULER està catalogat com a recurs disponible amb aquestes eines per aplicacions no comercials.

Distribució.

Per arribar als seus objectius, EULER vé com un paquet auto instalable compost d'una sèrie de mòduls compilables dinàmicament amb les biblioteques dinàmiques (DLL). Cada mòdul es pot adaptar a un llenguatge molt específic fàcilment, utilitzant bases de dades dependents de llenguatge ad-hoc, el format del qual ha estat escollit de la manera més compatible amb les normes existents. Els mòduls estan connectats a un contenidor multinivell central (MLC) i, el qual s'anomena seqüencialment amb una rutina. Els usos d'EULER basats en postres també es proporcionen.

El MLC és el cos del TTS. Ha estat dissenyat amb metodologies superiors de la tecnologia de dotació lògica, i es pot veure com a extensió de la biblioteca de plantilla estàndard de C++ (STL) per a estructures de dades multinivell. Es proporciona com codi font de GNU de C++,

amb la guia completa de programació d'EULER. Això fa possible que els usuaris creïn els seus propis mòduls de TTS per a EULER (o crear interfície entre els seus mòduls i el MLC) assegurant-li el control total del seu treball.
Els mòduls d'EULER també es proporcionen com codi font de GNU de C++ quan és possible. Si no, es proporcionen com DLLs.
Les bases de dades d'EULER es fan disponibles en un format compacte, per protegir els drets de proveïdors de base de dades. Tots els mòduls d'EULER, tanmateix, poden llegir les dos versions de bases de dades, compactades i no-compactades, de manera que els usuaris d'EULER puguin desenvolupar fàcilment les seves pròpies bases de dades i provar-les dins d'EULER.

3.2. Mòduls i bases de dades d'EULER.

Els mòduls disponibles desenvolupats per a cada tasca s'enumeren a sota. Les bases de dades existents depenents del llenguatge s'anomenen a cada cas. Quan és possible, els mòduls d'EULER utilitzen tecnologia basada en el corpus, que es pot adaptar fàcilment a uns altres llenguatges.

Tasca	Subtasca	Mòduls(llenguatges)
Text a fonètica	Preprocessament	RulesPreprocessorFr [fr, sea, ch]
	Accés lèxic i anàlisi morfològica	RulesLemmatizer [fr]
	Fonetització	ID3Phonetizer [fr, en, ar] PostPhonetizerFr [fr]
	Marcar amb etiquetes part de la frase	NgramTagger [fr]
	Generació de prosòdia	FMProsodyGenerator (corpus-based) [fr, es]
Fonètica a parla	Síntesi basada en difonemes	MBROLASynthesizer [us, ar, br, bz, cr, nl, ee, fr, de, gr, ro, sp, sw]

MBROLA posseix tres bases de dades per al castellà, es1, es2 i es4. Totes tres són de veu masculina, fent proves amb arxius parlats hem escollit la es1 perquè és la que millor sona.

La transcripció dels fonemes de la base de dades és la següent:

p	padre	"paDre
b	vino	"bino
t	tomo	"tomo
d	donde	"donde
k	casa	"kasa
g	gata	"gata
tS	mucho	"mutSo
jj	hielo	"jjelo
w	huele	"wele
f f	ácil	"faTil
T	cinco	"Tinko
s	sala	"sala
x	mujer	mu"xer
m	mismo	"mismo
n	nunca	"nunka
J	año	"aJo
l l	ejos	"lexos
L	caballo	ka"baLo
r	puro	"puro
rr	torre	"torre
i	pico	"piko
e	pero	"pero
a	valle	"baLe
o	toro	"toro
u	duro	"duro
_	silence	

Hi ha fonemes on té limitacions, en particular els diftongs, ja que no són tractats adequadament. Aquests són alguns exemples:

j	rei	rrej
	pie	pje
w	deuda	"dewDa
	muy	mwi
B	cabra	"kaBra (= /b/)
D	nada	"naDa (= /d/)
G	luego	"lweGo (= /g/)

Capítol 4.

Descripció dels programes.

4.1. Fitxers d'entrada i esquema general.

Fitxers PHO.

Els fitxers d'entrada de MBROLA són fitxers .pho, cada linia conté un fonema, una durada (en ms) i una sèrie de possibles dades; el tant per cent de la durada total del fonema i el valor de la freqüència fonamental (en Hz).

Exemple d'un fitxer .pho:

```
%Tenia el labio hacia detras
_       50      0 120
t       85
e       90
n       80      0 100
i       96      30 130
a       90      90 100
e       90
l       80      0 100
a       108     30 130
b       60
i       60
o       90      90 100
a       90
T       100
i       60
a       90
d       60
e       90
t       85      0 100
r       50
a       108     30 130   80 90
s       110     99 80
_       50      99 80
```

La primera columna és la durada. La segona el tant per cent, que pot ser zero com en algún cas d'aquest exemple. Aquest valor indica el tant per cent de la durada que aquest fonema estará sonant a la freqüéncia que el segueix, és per això que hi ha fonemes que tenen més de tres columnes. El fonema marcat, per exemple, dura 108ms, però un 30% sonarà a una freqüència de 130Hz i la resta del temps a una de 90Hz.

Fitxers MIDI i KAR.

Com fitxers d'entrada d'àudio he utilitzat fitxers MIDI, "Musical Instruments Digital Interface" (Interfaç digital per a instruments musicals), i per al text de la cançó fitxers tipus text.

Vaig escollir els fitxers MIDI perquè tenen informació de la durada i de les notes MIDI, amb la qual cosa podia obtenir informació sobre la freqüència. Per tant, amb aquests fitxers aconsegueixo totes les dades de la melodia.

Buscant informació vaig trobar uns fitxers tipus KAR (Karaoke), que no són més que arxius MIDI però amb el text, la qual cosa hem va fer replantejar el projecte, ja que podia utilitzar un sol fitxer d'entrada amb tota la informació desitjada. Aquests fitxers tenen la mateixa informació acústica que els fitxers MIDI.

Un arxiu Kar, a l'igual que un midi, conté bàsicament dues informacions importants, informació d'encapçalament (o capçalera) i informació de pista. Conté una línia d'encapçalament on es descriu el format del fitxer, l'autor,etc., i diferents trossos de pista. Es pot pensar en una pista de la mateixa manera com una pista en un cassette multi-track. Podem assignar una pista a cada veu, cada instrument o qualsevol cosa que vulgui, de forma que buscant el canal de la melodia s'extreurà la informació necessària per generar veu cantada.

Fitxers ml.

Aquests tipus de fitxers són els que obtenim a la sortida del transcriptor fonètic escollit, el propi de la UPC. La sortida d'aquest transcriptor és massa detallada per l'estudi posterior, de forma que només s'utilitza la part de les síl·labes i la dels fonemes, que és el que es necessita posteriorment, ja que la resta d'informació la aconseguim del fitxer MIDI/kar.

El fitxer de sortida del transcriptor és un fitxer .ml, amb aquesta forma:

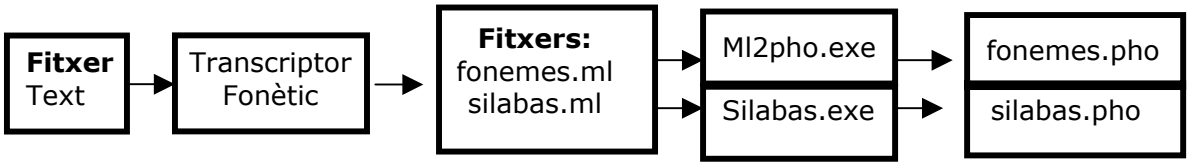
```
#-------------------------------------------------------------------------------
@LAYER PHONE
"p"     D=72.5183       E=0     F0=170;
"o"     D=50.5238       E=67.5  F0=170.8;
"r"     C=1     D=37.2318       E=59.14 F0=171.6;
"k"     D=79.611        E=0     F0=172.4;
"e"     D=66.0272       E=68.69 F0=173.2;
"e"     D=81.5746       E=69.19 F0=174;
"s"     C=1     D=94.3516       E=63.06 F0=169.467;
"t"     D=65.2023       E=0     F0=164.933;
"e"     D=47.0074       E=68.69 F0=160.4;
"m"     D=67.3524       E=64.62 F0=159.2;
"u"     D=73.2414       E=67.17 F0=168;
"n"     C=1     D=54.8177       E=64.27 F0=163.467;
"d"     D=54.917        E=54.48 F0=158.933;
"o"     D=46.4678       E=67.5  F0=154.4;
"n"     D=45.1622       E=64.27 F0=153.2;
"o"     D=68.5462       E=68    F0=162;
```

```
"l"     D=54.917        E=65.46 F0=150.8;
"o"     D=63.5161       E=67.5  F0=151.267;
"e"     D=47.0074       E=68.69 F0=151.733;
"n"     C=1     D=54.8177       E=64.27 F0=152.2;
"t"     D=65.2023       E=0     F0=152.667;
"j"     D=48.4014       E=66.92 F0=151.133;
"e"     D=74.7358       E=69.19 F0=149.6;
"n"     C=1     D=54.8177       E=64.27 F0=143.067;
"d"     D=54.917        E=54.48 F0=136.533;
"o"     D=77.1308       E=67.5  F0=130;
EVENT="BREAK"           EVENT_IDX=16;
```

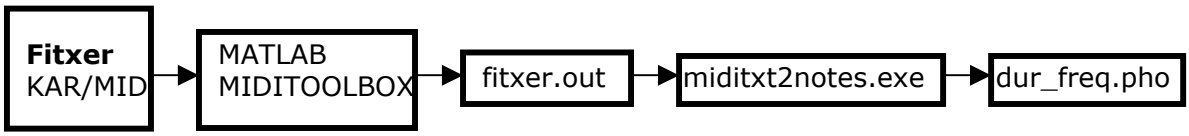
Esquema general.

Aquest és l'esquema gràfic dels mòduls a generar per aconseguir la generació de veu cantada.

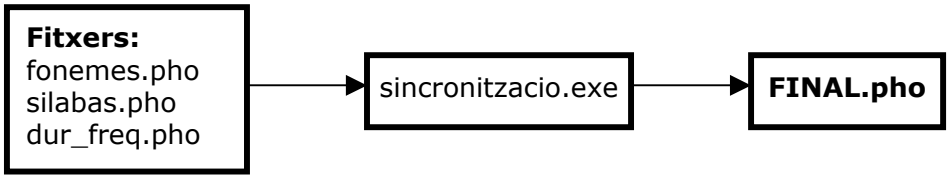
Branca del text.



Branca de l'àudio.



Sincronització text + àudio.



4.2. Conversió dels fitxers MIDI binaris.

Durant la recerca de informació sobre els fitxers MIDI vaig trobar un toolbox de MATLAB per a fitxers MIDI, amb el que es poden aconseguir uns fitxers de MIDI en format text, més fàcils de tractar posteriorment.

La funció del MIDItoolbox per aconseguir aquest fitxer és la funció readmidi, que converteix el codi binari d'un fitxer MIDI en una matriu de notes. A la funció es genera un fitxer temporal (.out), el fitxer de text que interessa, per tant s'ha de modificar la funció per tal que a la sortida no el borri i es pugui utilitzar.

Funció readmidi:

```
function nmat = readmidi(fn);
        nmat=[];
        if strcmp(computer,'MAC2')
                currdir = cd;
                % remove high bytes that occasionally occur in the path
                currdir = char(bitand(double(currdir),255));
                toolboxpath = which('readmidi');
                toolboxpath=toolboxpath(1:end-11); % remove ':readmidi.m'
                cd(toolboxpath);
                fid = fopen('mf2t.param','wt');
                ifile = [currdir ':' fn];
                fprintf(fid, '%s\n', ifile);
                ofile = [toolboxpath ':mf2t.out'];
                fprintf(fid, '%s\n', ofile);
                fclose(fid);
                ! mf2tmac
                delete mf2t.param
                nmat = mftxt2nmat(ofile);
                cd(currdir);
        elseif strcmp(computer,'PCWIN')
                midi2text(fn, 'MF2T.OUT'); % uusin mex konvertteri [2. tammikuuta 2003]
                nmat = mftxt2nmat('MF2T.OUT');
                delete mf2t.out
                clear mex
        elseif isunix
                midi2textunix(fn,'mf2t.out');
                nmat = mftxt2nmat('MF2T.OUT');
                delete mf2t.out
                clear mex
        else
                disp('This function works only on Macintosh and Windows!');
                return;
        end
```

Linia que s'ha d'eliminar per no destruïr el fitxer de text.

Com hem comentat anteriorment, els fitxers KAR són com els MIDI, per tant aquest toolbox també ens serveix per aquests fitxers, obtenint així un arxiu més entenedor. Per tant, un cop modifiquem la funció ja podem generar el fitxer en format text per tal de visualitzar-ho.

El fitxer que obtenim és un fitxer de tipus out, i té aquesta forma:

Pista de text, es troba el text de la cançó

480 Meta Text "\\AMARAL "
960 Meta Text "/Toda la noche en la calle "
1920 Meta Text "\\Sincronizado por: "
2880 Meta Text "\\JESUS RUIZ "
3600 Meta Text "/jideas@eresmas.com "
4560 Meta Text ""*
5280 Meta Text ""*
6000 Meta Text ""*
6960 Meta Text ""*
7800 Meta Text "\\POR"
8160 Meta Text "QUE ES"
8280 Meta Text "TE "
8520 Meta Text "MUN"
8760 Meta Text "DO "

Pista de la melodia, es troba la informació acústica.

105843 On ch=6 n=50 v=0
105878 On ch=6 n=57 v=0
105892 On ch=6 n=62 v=0
105899 On ch=6 n=66 v=0
106076 On ch=6 n=50 v=91
106078 On ch=6 n=57 v=91
106079 On ch=6 n=66 v=114
106081 On ch=6 n=62 v=96
106140 On ch=6 n=62 v=0
106140 On ch=6 n=66 v=0
106157 On ch=6 n=57 v=0
106160 On ch=6 n=50 v=0
106560 On ch=6 n=57 v=96
106560 On ch=6 n=66 v=123
106561 On ch=6 n=62 v=106

Al veure el format de sortida de la informació de text vaig pensar com extreure el que interessa i com fer-ho entenedor per al següent módul, el transcriptor fonètic. El problema que va aparèixer va ser el de final de frase i el de final de paràgraf, a més de no saber com agrupar les paraules, ja que venen separades en síl·labes. Al transcriptor s'ha d'entrar el text correcte ja que sino es generen les pauses i les durades de forma errònea. Després de romiar com solucionar aquest problema vaig pensar que el millor era tornar a la idea del principi, text i àudio en fitxers separats.

La informació que s'obté de la melodia és la comentada anteriorment. Per una banda la durada de cada nota, útil per calcular la durada dels fonemes. També s'aconsegueix el número de la nota Midi que s'activa, per tant, es pot calcular la freqüència. I per últim apareix una informació sobre la velocitat, que indica quan s'activa i quan es desactiva cada nota.

4.3. Transcripció fonètica i conversió a PHO.

L'objectiu del transcriptor fonètic és obtenir a partir de les paraules els sons que s'han de pronunciar. Transcriptors fonètics en castellà MBROLA només té el d'Alistair Conkie. El problema és que està desenvolupat en llenguatge PERL i, per tant, pensat per a funcionar en entorns UNIX/LINUX, és a dir, no compatible amb WINDOWS que és el que s'utilitza en aquest projecte. Per tant vaig utilitzar el transcriptor fonètic de la UPC. La sortida d'aquest transcriptor és massa detallada per l'estudi posterior, de forma que només s'utilitza la part de les síl·labes i la dels fonemes, que és el que es necessita posteriorment, ja que la resta d'informació la aconseguim del fitxer MIDI/kar.

Els fitxers de sortida del transcriptor, comentats anteriorment, s'han d'adaptar de forma que MBROLA els pugui entendre, és a dir, s'han de convertir a format .pho.

Per fer-ho he utilitzat programació en C, ja que és l'entorn que hem estudiat al llarg de la carrera, aconseguint diferents executables que generen els nous arxius amb els formats adequats.

Com he comentat anteriorment els arxius .pho tenen quatre columnes mínimes, els fonemes, les durades, el tant per cent i la freqüència, al igual que els fitxers .ml, però amb sintaxi diferent. Per exemple, els fonemes als arxius .ml están entre cometes, en canvi als pho no porten, les durades i les freqüències van precedides de lletres, als pho són només les dades, etc. Per tant s'ha de programar una funció per fer aquesta transformació, és la funció anomenada ml2pho.

Codi de la funció ml2pho.cpp:

```
// ml2pho.cpp : Defines the entry point for the console application.
//
```

S'inclouen llibreries i es defineixen constants.

```
# include <stdlib.h>
# include <string.h>        //incluim llibreria string.h perquè treballem amb strings de caràcters
# include <stdio.h>
# define MAX_LINIA 100
```

El fitxer d'entrada és el de la sortida del transcriptor fonètic amb els fonemes (fitxer .ml), i el de sortida serà un fitxer tipus .pho, amb els fonemes, les durades i les freqüències del text parlat.

```
int Empleo (char * prog) {

/*FichIn= fitxer de sortida del transcriptor (.ml)
  FichOut= fitxer d'entrada a MBROLA (.pho parlat)*/


        (void) fprintf(stderr, "\n");
        (void) fprintf(stderr, "Empleo:\n");
```

```
        (void) fprintf(stderr, "\t%s (FichIn.txt|-) (FichOut.txt|-)\n", prog);
        (void) fprintf(stderr, "donde\n");
        (void) fprintf(stderr, "\t(FichIn.txt|-) :\tFichero con texto a procesar   o entrada estandar\n");
        (void) fprintf(stderr, "\t(FichOut.txt|-) :\tFichero con texto procesado o salida estandar\n");
        (void) fprintf(stderr, "\n\n");
        return -1;
}
```

Creem variables i obrim fitxers d'entrada i sortida.

```
main(int argc, char **argv)
{
 FILE *fIn, *fOut;                  //fitxers d'entrada i sortida
 char  linia[MAX_LINIA];            //creem una linia de caràcters per llegir després les
                                    //linies dels fitxers
    if (argc != 3) return Empleo(*argv);        //han d'haber tres argumets quan
                                                //cridem la funció

  /* obté nom fitxer entrada */
  if (!strcmp(argv[1], "-"))
          fIn = stdin;
  else {
            fIn = fopen(argv[1], "rt");         //obrim el primer fitxer i el llegim
           if (fIn == NULL) {
         fprintf(stderr, "Error al abrir fichero de entrada '%s'\n", argv[1]);
         return Empleo(*argv);
           }
  }
/* obte nom fitxer sortida */
  if (!strcmp(argv[2], "-"))
          fOut = stdout;
  else {
            fOut = fopen(argv[2], "wt");        //obrim el segon fitxer i veiem que
                                                //es d'escriptura, serà el fitxer de
                                                //sortida
           if (fOut == NULL) {
         fprintf(stderr, "Error al abrir fichero de salida '%s'\n", argv[2]);
          return Empleo(*argv);
           }
  }
```

Creem variables i les inicialitzem.

```
char *event, *link;
char fonema[3], *pc;
```

Llegim fitxer d'entrada linia a linia, les dos primeres linies no interessen, són la linia de punts que separa els nivells i la linia d'inici de nivell.

```
 /* lectura entrada */
fgets(linia,MAX_LINIA,fIn);   /* primera linia no s'aprofita */
fgets(linia,MAX_LINIA,fIn);   /* segona linia no s'aprofita */
```

Inicialitzo link com a Null per detectar l'inici de la secció, quan trobi '#------' a la linia la variable link serà diferent de Null i per tant haurà acabat la secció.

```
link = NULL;            /* serveix per detectar inici secció @LINK */
while ( (feof(fIn)==0) && (link == NULL)) {
        fgets(linia,MAX_LINIA,fIn);         //agafem la següent linia del fitxer fIn, és a
                                            //dir, del d'entrada
```

Busco si a la linia que ha llegit es troba la paraula 'EVENT', si la troba vol dir que és una pausa, per tant llegirà el valor, el multiplica per 20, perquè sino aquest valor de durada per a MBROLA és massa petit, i escriu el fitxer de sortida.

```
event = strstr(linia,"EVENT");      //busquem si a la linia trobem la paraula EVENT
                                    //event!=NULL vol dir que és un event, per tant,
                                    //tindrem una pausa
link = strstr(linia,"#-------");    /* link!=NULL significa que hem arribat a la
                                    línia anterior a l'inici de la secció @lINK */
if (event != NULL) {                // es un Event !
float eve;
pc=event;                           //coloquem el punter pc on estaba event
sscanf(pc+24,"%f",&eve);            //avancem 24 posicions i guardem el valor a eve
eve= eve*20;                        //multipliquem per 20 perquè es un valor massa petit
                                    //per a MBROLA
fprintf(fOut,"_\t%f\n",eve);        //escrivim al fitxer de sortida
}
```

Si no és un event ni un link, vol dir que és un fonema, per tant el llegim i el guardem.

```
else if (link == NULL) {   // no es el final link, ergo fonema
        if (strlen(linia)>1) {      //si la longitud de la linia es mes gran que 1
```

Els fonemes s'han d'adaptar perquè siguin compatibles amb la base de dades escollida de MBROLA. Podria ser que hi hagin fonemes amb dos caràcters, per tant, per no trobar errors de compatibilitat, poso a cero el segon caràcter del fonema.

```
// podria ser que hi hagin fonemes de 2 caràcters (p.e. /jj/, /tS/)
//sscanf(linia+1,"%s",fonema);posem a cero el segon fonema
//fonema[strlen(fonema)-1]='\0';

//Adaptem fonemes xq siguin compatibles amb la base de dades de MBrola
        pc=strchr(fonema,'j');
        if (pc != NULL){
                fonema[0]= 'i';             //canvi de j per i
       }
        pc=strchr(fonema,'w');
        if (pc != NULL){
                fonema[0]= 'u';             //canvi w per u
        }
        pc=strchr(fonema,'B');
        if (pc != NULL){
                fonema[0]= 'b';             //canvi B per b
        }
        pc=strchr(fonema,'D');
        if (pc != NULL){
                fonema[0]= 'd';                     //cavi D per d
        }
       pc=strchr(fonema,'G');
        if (pc != NULL){
                fonema[0]= 'g';                     //canvi G per g
        }
        pc=strchr(fonema,'N');
        if (pc != NULL){
                fonema[0]= 'n';                     //canvi N per n
        }
```

```
pc=strchr(fonema,'z');
 if (pc!=NULL){
        fonema[0] ='s';                    //canvi la z per la s
 }
 pc=strchr(fonema,'Z');
 if (pc!=NULL){
        fonema[0] ='s';                    //canvi la z per la s
 }
// podria ser que hi hagin fonemes de 2 caràcters (p.e. /jj/, /tS/)
 if (strlen(fonema)!= 1){
        fonema[1]='\0';
 }
```

Es localitzen les durades i les freqüències dels fonemes a la linia del fitxer d'entada, es copien i s'escriuen al fitxer de sortida, de forma que sigui compatible amb MBROLA, per tant hem de crear una columna corresponent al tant per cent, que serà de 50.

```
float dur,txc, F0;
  txc=50;                    //afegim columna de 50%
  pc = strstr(linia,"D="); //localitzem la durada a l'string
  sscanf(pc+2,"%f",&dur);   //copiem durada a dur
  pc = strstr(linia,"F0=");          //localitzem la freqüència
  sscanf(pc+3,"%f",&F0);   //copiem freqüència a FO
fprintf(fOut,"%s\t%f\t%f\t%f\n",fonema,dur,txc,F0);          //copiem fonema,
                                                            //durada, tantxcent i
                                                            //freq al fitxer pho
```

Per últim es tanquen els fitxers d'entrada i sortida.

```
/* tanquem fitxers */
 if (fIn != stdin)
    fclose(fIn);
if (fOut != stdout)
   fclose(fOut);
 return 0;
}
```

Amb aquesta funció aconsegueixo els fonemes i les pauses correctes i les durades i les freqüències que corresponen al text parlat. Per tant hauré d'aconseguir a partir del fitxer MIDI la resta de la informació.

Format del fitxer de sortida de la funció ml2pho, fonemes.pho,:

```
_       340.000000
p       72.518303       50.000000       170.000000
o       50.523800       50.000000       170.800003
r       37.231800       50.000000       171.600006
k       79.611000       50.000000       172.399994
e       66.027199       50.000000       173.199997
e       81.574600       50.000000       174.000000
s       94.351601       50.000000       169.466995
t       65.202301       50.000000       164.932999
e       47.007401       50.000000       160.399994
m       67.352402       50.000000       159.199997
u       73.241402       50.000000       168.000000
n       54.817699       50.000000       163.466995
d       54.917000       50.000000       158.932999
o       46.467800       50.000000       154.399994
n       45.162201       50.000000       153.199997
o       68.546204       50.000000       162.000000
l       54.917000       50.000000       150.800003
o       63.516102       50.000000       151.266998
```

e	*47.007401*	*50.000000*	*151.733002*
n	*54.817699*	*50.000000*	*152.199997*
t	*65.202301*	*50.000000*	*152.667007*
i	*48.401402*	*50.000000*	*151.132996*
e	*74.735802*	*50.000000*	*149.600006*
n	*54.817699*	*50.000000*	*143.067001*
d	*54.917000*	*50.000000*	*136.533005*
o	*77.130798*	*50.000000*	*130.000000*
_	*360.000000*		

Així doncs només he d'aconseguir les durades i les freqüències corresponents a la melodia de la cançó per fer que la veu generada sigui cantada.

4.4. Extracció de la melodía.

Com he esmentat en capítols anteriors, amb la durada dels fonemes i la freqüència es pot generar la melodia de la cançó, per tant a partir del fitxer d'àudio, el midi, hauré d'aconseguir aquestes dades per tal de poder generar la veu cantada.

Els arxius midi, com ja he comentat, tenen diferents pistes o canals amb diferents dades, per tant, hauré de localitzar el canal on apareixen les dades de la melodia. Per fer-ho he treballat amb VOYETRA, un editor de fitxers midi que permet visualitzar totes les pistes d'un arxiu en forma de partitura i per tant veure quin és el canal a processar.

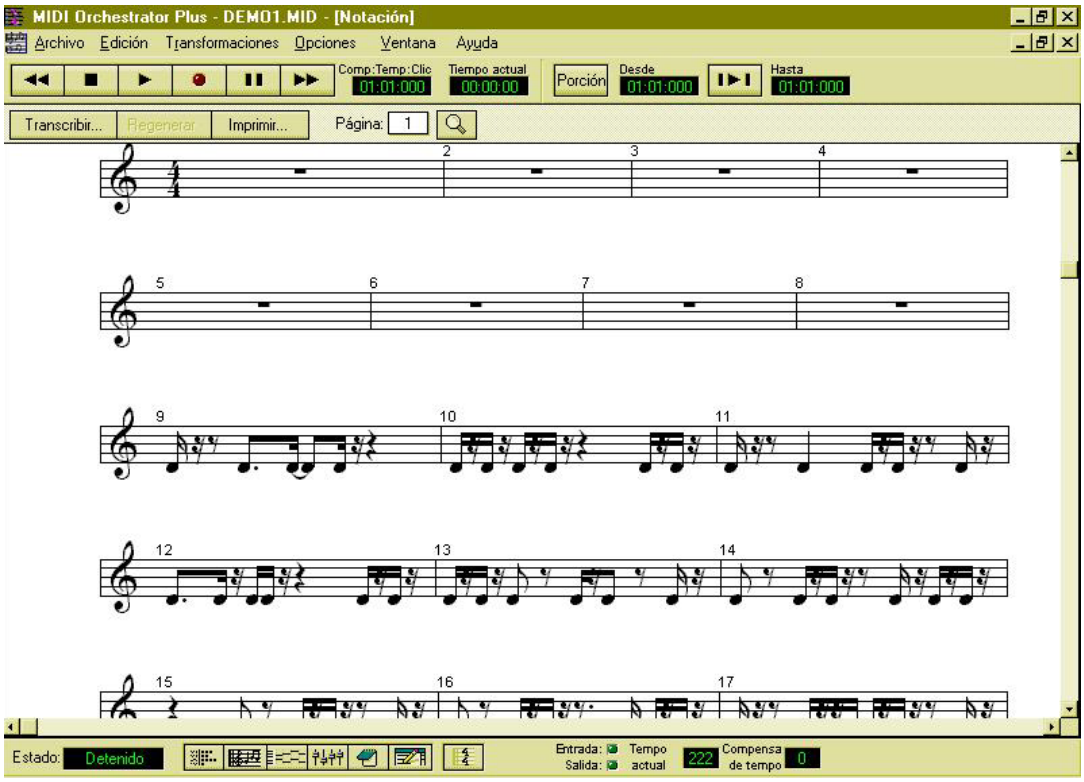


Figura 8. Vista del programa VOYETRA.

Així doncs un cop localitzat el canal de la melodia només he de buscar-ho a l'arxiu midi text i extreure la informació necessària, en aquest cas la durada i la freqüència.
Als fitxers midi trobem informació de canal, de nota i de velocitat, que ens indica quan s'activa i quan es desactiva una nota (serà 0 quan es desactiva). També apareix informació de temps, els números de principi de linia, útils per esbrinar la durada. Per tant, el que he de fer és, seleccionar el canal corresponent a la melodia, agafar la nota midi i convertir-la a freqüència i anar llegint les velocitats de forma que quan es desactivi una nota poder fer la resta del temps per saber quina és la durada d'aquella nota. Així doncs vaig programar la funció miditxt2notes, on obtenim un fitxer .pho amb les durades i les freqüències corresponents a la melodia.

Codi de la funció miditxt2notes.cpp:

```
// miditxt2notes.cpp : Defines the entry point for the console application.
//
```

S'inclouen llibreries

```
# include <stdlib.h>
# include <string.h>
# include <stdio.h>
# include <math.h>          //llibreria per funcions matemàtiques
# include <iostream.h>
```

Definin constants, caràcters màxims de linia i nombre màxim de notes.

```
# define MAX_LINIA 1000
# define max_note  150
```

Quan s'executa aquesta funció s'ha d'escollir el canal del qual volem obtenir les dades. El fitxer d'entrada és el de midi de text.

```
Int Empleo3 (char * prog) {

        (void) fprintf(stderr, "\n");
        (void) fprintf(stderr, "Empleo3:\n");
        (void) fprintf(stderr, "\t%s  [-c channel] (FichIn.txt|-) (FichOut.txt|-)\n", prog);
        (void) fprintf(stderr, "donde\n");
        (void) fprintf(stderr, "\t[-c channel]:\tnumero de canal de la melodia del fichero kar\n");
        (void) fprintf(stderr, "\t(FichIn.txt|-) :\tFichero con texto a procesar o entrada estandar\n");
        (void) fprintf(stderr, "\t(FichOut.txt|-) :\tFichero con texto procesado o salida estandar\n");
        (void) fprintf(stderr, "\n\n");
        return -1;
}
```

Es creen les variables, obrim fitxers i llegim el número de canal d'on hem d'obtenir les dades necessàries.

```
        main(int argc, char **argv)
        {

          FILE *fIn, *fOut;
```

```
char linia[MAX_LINIA];
char *nch;

if (argc !=5) return Empleo3(*argv);

if (argv[2]=="-c"){                    //si el primer que escribim es -c
        nch=argv[3];            //ha de guardar el segon argument, ja que es el
                                //numero de canal de la melodia
}
argv = argv+2;                  //sumem 2 a l'argument

/* obte nom fitxer entrada */
  if (!strcmp(argv[1], "-"))
        fIn = stdin;
  else {
        fIn = fopen(argv[1], "rt");
        if (fIn == NULL) {
                fprintf(stderr, "Error al abrir fichero de entrada '%s'\n", argv[1]);
        return Empleo3(*argv);
        }
  }

/* obte nom fitxer sortida */
  if (!strcmp(argv[2], "-"))
        fOut = stdout;
  else {
        fOut = fopen(argv[2], "wt");
        if (fOut == NULL) {
                fprintf(stderr, "Error al abrir fichero de salida '%s'\n", argv[2]);
        return Empleo3(*argv);
        }
  }
```

Creem variables i les inicialitzem.

```
  /*creem les variables*/
  char *on, *ch,*vel,*note;
  float freq,txc,n,exp,m;
  float time,vl,nota,timeon,timeoff,not,dur,duron,duroff;
  int tabla[max_note][3];
  int fila;
  char chanel;
```

Mentre no s'arribi al final del fitxer d'entrada llegim linia a linia del canal que hem escollit i mirem que estigi On, és a dir, actiu. Busquem el número de la nota per poder calcular la freqüència i guardem la durada de cada linia per saber després la durada total.

```
while (feof(fIn)==0) {

        fgets(linia,MAX_LINIA,fIn);             //llegim linia de fitxer d'entrada

        on = strstr(linia,"On");        //busquem al string que la nota estigui ON
        ch = strstr(linia, "ch=7");     /busquem que sigui el canal de la melodia,
                                        //en aquest cas el 7
        vel = strstr(linia,"v=");
        note = strstr(linia,"n=");
        //sscanf(ch+3,"%s",&chanel);
        //chanel haura de ser nch, el numero de canal que escollim a l'argument
```

```
if (ch!=NULL){                  //chanel no ha de ser NULL,
                                //així serà el canal seleccionat

    if (on!=NULL){              //nota ha d'estar On

    sscanf(linia,"%f", &time);//guardem a time el temps
    sscanf(vel+2,"%f",&vl);     //guardem a vl la velocitat, per saber quan //s'encen
                                //i quan s'apaga
    sscanf(note+2,"%f",&nota);  //guardem a nota el número de nota
```

Abans s'ha creat una taula on guardar el temps i el número de les notes. La taula té tres columnes, a la primera posaem la nota, p.ex. si la nota és la 45 doncs la col·loquem a la primera columna a la fila 45. A la segona el temps quan la velocitat sigui diferent de cero, és a di, quan la nota s'activa. I a la tercera el temps quan la velocitat sigui cero, quan la nota es desactiva.

```
fila=(short)nota;           //guardem la nota a la fila corresponent al seu
                            //número p.ex.nota 32 a la fila 32
tabla[fila][0]=(short)nota;         //la guardem a la taula


    if (vl != 0){           //activem nota
    timeon=time;            //temps quan activem
    tabla[fila][1]=(short) timeon;  //el guardem a la segona columna
    }
    else {                  //desactivem nota
    timeoff=time;           //temps quan es desactiva
    tabla[fila][2]=(short)timeoff;  //el guardem a la tercera columna
    not=(short)tabla[fila][0];      //agafem la nota a la correspon
```

A continuació es relacionen les notes MIDI amb la freqüència en Hz. S'agafa com a freqüència de referència els 440Hz, que corresponen amb la nota midi número 69.

```
/*relacionem nota MIDI amb freqüència en Hz
1 octava = 12 semitons
12*log2 (Freq/Freq de ref) = n
Freq = (Freq de referencia) * (2 elevat a n/12);
on n es el numero de semitons*/

n = not - 69;       //restem 69 a la nota perque la nota 69
                    //son els 440Hz(La)i a partir d'aquesta
                    //diferencia podem calcular la freqüència
exp = n/12;         //1 octava = 12 semitonos, n = numero de semitonos
m = pow( 2,exp); //funcio per realitzar el 2 elevat a exp

freq = 440 * m;     //per saber la freqüència multipliquem la
                    //freq de referencia (440Hz, la de la nota 69)
                    //per el 2 elevado a exp
```

Es calcula la durada de cada síl·laba restant al temps quan la nota es desactiva el temps quan la nota s'activa, és a dir, el temps de la tercera columna menys el de la segona.

```
/*trobem durada*/
duron=(short)tabla [fila][1];       //temps activa
duroff=(short)tabla [fila][2];      //temps desactiva
dur=duroff-duron;                   //diferencia es el temps que ha
                                    //sonat la nota
```

Escrivim el fitxer de sortida, com en aquesta funció no hi ha cap informació dels fonemes creem una columna amb el fonema 'a'.

```
/*escrivim fitxer de sortida*/
txc=50;                                    //afegim una columna de 50%
fprintf(fOut, "a\t D= %f\t%f\t F= %f\n",dur,txc,freq);          //escrivim fitxer
                                                                //de sortida
                                                                //fonemes tots a
}
```

Per últim tanquem fitxers.

```
/*tanquem fitxers*/
if (fIn != stdin)
        fclose(fIn);
if (fOut != stdout)
        fclose(fOut);
return 0;
}
```

La intenció al programar la funció era que a la linia de comandes, quan s'executa, es pugués escollir el número de canal del que es vol extreure la informació. Però a la hora de programar no m'he ensortit, de manera que per cada arxiu s'ha de canviar al codi de la funció el número de canal on anar a buscar les dades i tornar a compilar. Tot i que a la linia de comandes s'ha de escriure el canal, ja que no ha estat modificat.

Un cop s'executa aquesta funció obtenim un fitxer .pho amb les durades i les freqüències correctes, però no amb els fonemes. Un exemple del fitxer de sortida (dur_freq.pho) és aquest:

```
a       D= 138.000000   50.000000       F= 146.832382
a       D= 167.000000   50.000000       F= 369.994415
a       D= 168.000000   50.000000       F= 220.000000
a       D= 174.000000   50.000000       F= 293.664764
a       D= 208.000000   50.000000       F= 146.832382
a       D= 214.000000   50.000000       F= 369.994415
a       D= 234.000000   50.000000       F= 220.000000
a       D= 236.000000   50.000000       F= 293.664764
a       D= 165.000000   50.000000       F= 146.832382
a       D= 197.000000   50.000000       F= 369.994415
a       D= 135.000000   50.000000       F= 146.832382
a       D= 389.000000   50.000000       F= 220.000000
a       D= 177.000000   50.000000       F= 391.995422
a       D= 424.000000   50.000000       F= 293.664764
a       D= 370.000000   50.000000       F= 195.997711
a       D= 389.000000   50.000000       F= 391.995422
a       D= 391.000000   50.000000       F= 246.941650
a       D= 408.000000   50.000000       F= 293.664764
a       D= 336.000000   50.000000       F= 195.997711
a       D= 358.000000   50.000000       F= 391.995422
a       D= 366.000000   50.000000       F= 246.941650
a       D= 378.000000   50.000000       F= 293.664764
a       D= 314.000000   50.000000       F= 195.997711
a       D= 343.000000   50.000000       F= 391.995422
a       D= 347.000000   50.000000       F= 246.941650
a       D= 353.000000   50.000000       F= 293.664764
```

Arribats a aquest punt hem vaig adonar que les notes de midi no corresponen a fonemes aïllats, sinó que corresponen a síl·labes, ja que quan es canta no es donen freqüències diferents a cada fonema, el que es fa és donar durades diferents als fonemes, però la freqüència és la mateixa. A més dins una síl·laba sempre la vocal és la que té la durada major. Per tant s'ha d'aconseguir també les síl·labes del text. Això no és cap problema ja que el transcriptor té un nivell en el que ens fa la partició per sílabes.

El fitxer .ml de les síl·labes té aquesta forma:

```
#----------------------------------------------------------------------------
@LAYER SYLLABLE
"por";
"ke";
'"es";
"te";
"m'un";
"do";
"n'o";
"lo";
"en";
"tj'en";
"do";
EVENT="BREAK"          EVENT_IDX=16;
"por";
"ke";
'"aj";
"Be";
"r'a";
"no";
"jj";
'"aj";
```

Així doncs, s'ha de generar una nova funció per adaptar el text. La funció silabas agafa només els fonemes que formen les síl·labes.

Codi de la funció silabas.cpp:

```
// silabas.cpp : Defines the entry point for the console application.
//
```

S'inclouen llibreries i es defineixen constants.

```
# include <stdlib.h>
# include <string.h>
# include <stdio.h>

# define MAX_LINIA 100
```

El fitxer d'entrada és l'arxiu de la sortida del transcriptor amb les síl·labes del text.

```
int Empleo6 (char * prog) {

    /*FichIn = fitxer de síl·labes del transcriptor
    FichOut = fitxer de síl·labes adaptat*/

    (void) fprintf(stderr, "\n");
    (void) fprintf(stderr, "Empleo6:\n");
    (void) fprintf(stderr, "\t%s (FichIn.txt|-) (FichOut.txt|-)\n", prog);
```

```
        (void) fprintf(stderr, "donde\n");
        (void) fprintf(stderr, "\t(FichIn.txt|-) :\tFichero con texto a procesar o entrada estandar\n");
        (void) fprintf(stderr, "\t(FichOut.txt|-) :\tFichero con texto procesado o salida estandar\n");
        (void) fprintf(stderr, "\n\n");
        return -1;
}
```

Inicialitzem variables i obrim els fitxers d'entrada i sortida.

```
        main(int argc, char **argv)
        {

          FILE *fIn, *fOut;
         char  linia[MAX_LINIA];

        if (argc != 3) return Empleo6(*argv);

        /* obte nom fitxer entrada */
        if (!strcmp(argv[1], "-"))
                fIn = stdin;
        else {
                fIn = fopen(argv[1], "rt");                     //obrim i lleguim fitxer d'entrada
                if (fIn == NULL) {
                        fprintf(stderr, "Error al abrir fichero de entrada '%s'\n", argv[1]);
                return Empleo6(*argv);
                  }
          }

         /* obte nom fitxer sortida */
        if (!strcmp(argv[2], "-"))
                fOut = stdout;
        else {
                fOut = fopen(argv[2], "wt");                    //obrim un fitxer d'escriptura
                if (fOut == NULL) {
                        fprintf(stderr, "Error al abrir fichero de salida '%s'\n", argv[2]);
                return Empleo6(*argv);
                  }
          }
```

Inicialitzem variables.

```
        char *event,*pc;
        char silaba[2],fon;
```

Llegim les primeres linies del fitxer d'entrada perquè no les necessitem.

```
        /* llegim entrada */
        fgets(linia,MAX_LINIA,fIn);         /* primera linia no s'aprofita */
        fgets(linia,MAX_LINIA,fIn);         /* segona linia no s'aprofita */
```

Mentre no s'arribi al final del fitxer d'entrada, anem llegint linia a linia i copiem al fitxer de sortida les síl·labes sense les cometes ni res, només els fonemes.

```
while (feof(fIn)==0) {

        fgets(linia,MAX_LINIA,fIn);                     //llegim següent linia
        event = strstr(linia,"EVENT");              //busquem EVENT dins la linia

        if (event == NULL) {                        //com event és NULL, per tant, no ho possa a la
                                                    //linia, es una silaba
```

```
                sscanf(linia+1,"%s",&silaba);     //copiem la silaba sense les cometes
                silaba[strlen(silaba)-2]='\0';    //borrem les cometes del final

                fprintf(fOut,"%s\n",silaba);              //esccrivim la sil·laba al fitxer de sortida
        }
}
```

Tanquem fitxers d'entrada i sortida.

```
        /* tanquem fitxers */
        if (fIn != stdin)
                fclose(fIn);
        if (fOut != stdout)
                fclose(fOut);
        return 0;
        }
```

Un cop executes aquesta funció s'obté un fitxer (silabes.txt) d'aquest tipus:

```
por
ke
es
te
mun
do
no
lo
en
tje
do
por
ke
aj
Be
ra
no
jj
```

Per tant, ara ens caldrà realitzar una sincronització de tot el que s'ha aconseguit fins al moment.

4.5. Sincronització.

Per realitzar la sincronització he anat llegint tots tres fitxers alhora. Per fer-ho he realitzat la funció sincronització.

Codi de la funció sincronització.cpp:

```
// sincronitzacio.cpp : Defines the entry point for the console application.
//
```

S'inclouen llibreries i es defineixen constants.

```
# include <stdlib.h>
# include <string.h>
# include <stdio.h>
# include <math.h>
# include <iostream.h>

# define MAX_LINIA 100
# define max_note 100
```

Tenim tres fitxers d'entrada:

1.Fitxer fonemes.pho

2.Fitxer silabes.txt

3.Fitxer dur_freq.pho

I un de sortida amb els fonemes, les durades i les freqüències correctes.

```
int Empleo5 (char * prog) {

(void) fprintf(stderr, "\n");
(void) fprintf(stderr, "Empleo5:\n");
(void) fprintf(stderr, "\t%s (FichIn.txt|-) (FichIn2.txt|-) (FichIn3.txt|-) (FichOut.txt|-)\n", prog);
(void) fprintf(stderr, "donde\n");
(void) fprintf(stderr, "\t(FichIn.txt|-) :\tFichero con texto a procesar o entrada estandar\n");
(void) fprintf(stderr, "\t(FichIn2.txt|-) :\tFichero con texto a procesar o entrada estandar\n");
(void) fprintf(stderr, "\t(FichIn3.txt|-) :\tFichero con texto a procesar o entrada estandar\n");
(void) fprintf(stderr, "\t(FichOut.txt|-) :\tFichero con texto procesado o salida estandar\n");
(void) fprintf(stderr, "\n\n");
return -1;
}
```

Llegim els tres fitxers d'entrada i creem el fitxer de sortida.

```
main(int argc, char **argv)
{
 FILE *fIn,*fIn2,*fIn3, *fOut;

char  linia1[MAX_LINIA];          //linia del primer fitxer
char  linia2[MAX_LINIA];          //linia del segon fitxer
char  linia3[MAX_LINIA];          //linia del tercer fitxer

  if (argc != 5) return Empleo5(*argv);

/* obte nom primer fitxer entrada */
if (!strcmp(argv[1], "-"))
        fIn = stdin;
  else {
        fIn = fopen(argv[1], "rt");
         if (fIn == NULL) {
```

```
                fprintf(stderr, "Error al abrir fichero de entrada '%s'\n", argv[1]);
        return Empleo5(*argv);
                }
  }

/* obte nom segon fitxer d'entrada */
if (!strcmp(argv[2], "-"))
        fIn2 = stdout;
  else {
        fIn2 = fopen(argv[2], "rt");
        if (fIn2 == NULL) {
                fprintf(stderr, "Error al abrir fichero de entrada2 '%s'\n", argv[2]);
        return Empleo5(*argv);
                }
  }
/* obte nom tercer fitxer d'entrada */
 if (!strcmp(argv[3], "-"))
        fIn3 = stdout;
  else {
        fIn3 = fopen(argv[3], "rt");
        if (fIn3 == NULL) {
                fprintf(stderr, "Error al abrir fichero de entrada3 '%s'\n", argv[3]);
        return Empleo5(*argv);
                }
  }

/* obte nom fitxer sortida */
if (!strcmp(argv[4], "-"))
        fOut = stdout;
  else {
        fOut = fopen(argv[4], "wt");
        if (fOut == NULL) {
                fprintf(stderr, "Error al abrir fichero de salida '%s'\n", argv[4]);
        return Empleo5(*argv);
                }
  }
```

Inicialitzem les variables.

```
char *freq,*dur;
float frecuencia,durada,txc,durfon,dura;
char silaba[10],fonema;
int numfone;
txc=50;
```

Llegim el primer fonema del fitxer fonemes.pho.

```
gets(linia1,MAX_LINIA,fIn);             //llegueixo linia del primer fitxer(fonemes)
sscanf(linia1,"%c",&fonema);            //guardo el fonema
```

Mentre no s'arribi al final de cap dels tres fitxers, es va llegint la durada i la freqüència del fitxer dur_freq.pho i les síl·labes del fitxer silabes.txt. La durada que tenim és la corresponent a la síl·laba sencera, per tant s'ha de dividir entre el número de fonemes de la síl·laba, però com la durada de la vocal ha de ser més llarga que la de la resta de fonemes doncs la dividim entre el nombre de fonemes més un.

```
while ((feof(fIn)==0) && (feof(fIn2)==0) && (feof(fIn3)==0)){

        fgets(linia3,MAX_LINIA,fIn3);       //llegueixo linia del tercer fitxer(dur_freq)
        freq=strstr(linia3,"F=");           //busco freqüència
        dur=strstr(linia3,"D=");            //busco durada
        sscanf(freq+2,"%f",&frecuencia);  //guardo freqüència d'una silaba
        sscanf(dur+2,"%f",&durada);         //guardo durada d'una silaba
        //frecuencia = (frecuencia)/(2);  //baixem una octava per tal de millorar el resultat

        fgets(linia2,MAX_LINIA,fIn2);       //llegueixo linia del segon fitxer(silabes)
        sscanf(linia2,"%s",&silaba);        //guardo silaba
        numfone=strlen(silaba);             //torna el numero de fonemes de la silaba
        durfon = durada / (numfone+1);    //aconsegueixo durada de cada fonema
                                            //divideixo durada de silba entre el nombre
                                            //de fonemes mes 1, per fer que la vocal
                                            //de la silaba tingui una durada major
```

Si el fonema que hem llegit és una pausa copiem la linia sencera del fitxer fonemes.pho al fitxer de sortida.

```
if (fonema == '_'){                         //si es una pausa copiem la linia igual

        fprintf(fOut,"%-10.10s\n",linia1);  //si es una pausa copiem la linia sencera
        fgets(linia1,MAX_LINIA,fIn);        //llegueixo fonema del primer fitxer
        sscanf(linia1,"%c",&fonema);        //guardo el fonema

}
```

Fins que no s'acabin els fonemes d'una síl·laba no llegeix una nova durada i una nova freqüència. L'ùnic que s'ha de llegir cada cop és el fonema, i si és una vocal donar-li una durada més gran.

```
while (numfone!=0){                         //fins que no acabi la silaba

/*si el fonema es una vocal hem de fer que la durada sigui major*/

if ((fonema == 'a') || (fonema == 'e') || (fonema == 'i') ||
   (fonema == 'o') || (fonema == 'u')){

        dura=durfon*2;                                  //durada serà el doble

        /*escrivim fitxer de sortida*/
        fprintf(fOut,"%c\t%f\t%f\t%f\n",fonema,dura,txc,frecuencia);

        numfone=numfone-1;              //disminuim el nombre de fonemes de la silaba

        fgets(linia1,MAX_LINIA,fIn);        //llegueixo fonema del primer fitxer
        sscanf(linia1,"%c",&fonema);        //guardo fonema
 }

else{                                       //si el fonema no es una vocal

        /*escrivim el fitxer de sortida amb la durada corresponent a un fonema*/
        fprintf(fOut,"%c\t%f\t%f\t%f\n",fonema,durfon,txc,frecuencia);

        numfone=numfone-1;              //disminuim el nombre de fonemes de la silaba

        fgets(linia1,MAX_LINIA,fIn);        //llegueixo fonema del primer fitxer
        sscanf(linia1,"%c",&fonema);        //guardo fonema
```

Per últim tanquem tots els fitxers.

```
/* tanquem fitxers */
  if (fIn != stdin)
        fclose(fIn);
  if (fOut != stdout)
        fclose(fOut);

  return 0;
}
```

Com la base de dades que utilitzem és de veu masculina, quan generem un fitxer que originalment està cantat per una veu femenina s'ha de baixar una octava (dividir la freqüència a la meitat), perquè sino sona massa agut, sona com una veu de 'pito'. És per això que apareix una linia on la freqüència es divideix a la meitat, perquè segons l'arxiu original aquesta linia s'activa o no.

```
//frecuencia = (frecuencia)/(2);  //baixem una octava per tal de millorar el resultat
```

Capítol 5.

Proves.

A aquest capítol es llisten els diferents fitxers de prova que he anat realitzant durant la realització d'aquest projecte.

5.1. Fitxers .pho.

Els fitxers d'aquest apartat són fitxers pho parlats aconseguits amb la base de dades de MBROLA. Escoltant-los es pot apreciar la qualitat del conversor. La veu sona una mica 'robot'. Faig servir la base de dades es1 perquè és la que proporciona millors resultats, les altres bases de dades o sonen pitjor o tenen accent americà.

Exemples de fitxers .pho parlats:

- example1.pho
- example11.pho
- example12.pho
- example13.pho
- example14.pho
- example15.pho
- example16.pho
- example17.pho
- example18.pho
- example19.pho
- example2.pho
- example20.pho
- example21.pho
- example3.pho
- example4.pho
- example5.pho
- example9.pho

5.2. Fitxers .pho parlats extrets dels fitxers midi de text.

Aquests són els fitxers .pho generats amb la funció ml2pho, utilitzant com a entrada el fitxer de sortida del transcriptor fonètic (fitxer .ml). El resultat és força bó, tot i que per als fitxers cantats originalment per veu femenina el resultat és una mica pitjor.

Fitxers pho parlats extrets de la funció ml2pho:

- Amaral.pho
- Belen.pho
- Calamaro.pho
- Rosana.pho

5.3. Fitxers .pho cantats fets manualment.

Aquest és un fitxers .pho que vaig generar manualment per probar el funcionament de MBROLA generant veu cantada, el vaig fer mirant l'arxiu midi de text. Vaig anar llegint les notes midi i vaig calcular la freqüència corresponent. Amb aquest fitxer vaig pensar que els resultats serien bons, ja que sona força bé, però no vaig pensar en el problema de la veu femenina.

Fitxer generat manualment:

- Calamanual.pho

5.4. Fitxers .pho definitius.

Dins aquesta llista trobem els fitxers .pho cantats generats amb la funció sincronització a partir dels fonemes, les síl·labes i les durades i les freqüències. Degut al problema amb la veu de la base de dades comentat a l'apartat anterior he generat dos llistes diferents. Els fitxers que porten el 2 al final són els generats baixant una ocatava (dividint per dos), per tant, els fitxers Amaralfin2, Belenfin2 i Rosanafin2, sonen bastant millor que els fitxers amb les freqüències originals. Mentre que amb el fitxer Calamaro passa el contrari, ja que al baixar la freqüència sona molt pitjor, ja que la freqüència és massa baixa. Com a conclusió podem dir que el resultat per a veu masculina és bó, mentre que per la femenina s'hauria d'acurar.

Fitxers pho cantats:

- Amaralfin.pho (freqüència original).
- Belenfin.pho (freqüència original).
- Calamarofin.pho (freqüència original).
- Rosanafin.pho (freqüència original).

- Amaralfin2.pho (freqüència a la meitat).
- Belenfin2.pho (freqüència a la meitat).
- Calamarofin2.pho (freqüència a la meitat).
- Rosanafin2.pho (freqüència a la meitat).

Capítol 6.

Conclusions.

En aquesta memòria presento els possibles blocs per la generació de veu sintética cantada. El major problema trobat al llarg de tot el projecte, a més de la manca de informació en quant als formats dels arxius escollits i els sistemes TTS, ha estat la programació en C, degut al baix nivell de coneixements, però era la forma més sencilla de totes les opcions possibles, per tant vaig haber de realitzar un repàs i buscar manuals d'ajuda. També ha estat un incovenient la falta de 'teoria musical', és a dir, coneixements sobre música, sobre tot a la part de la extracció de la melodia i amb el càlcul de la freqüència a partir de les notes midi.

Els resultats obtinguts, per veu masculina són acceptables, mentre que els resultats per a veu femenina són força dolents, tot i haber modificat les freqüències, ja que la base de dades que he fet servir és de veu masculina. No es pot saber com sonaria amb una base de dades femenina, ja que MBROLA no ofereix cap.

Degut a la manca de temps hi ha ajustos força sencills, com per exemple la durada, que seria un possible punt a millorar en futurs projectes, ja que a la sincronització la assignació és molt arbitrària, ja que es podria fer d'una forma més estudiada.

També es podria millorar la funció miditxt2notes, de forma que el canal es pugui seleccinar a la hora de executar la funció i no haber de compilar cada cop que la executem.

El problema amb la base de dades també seria una possible millora, ja que es podria fer algun ajust per tal d'ajustar-la sempre a una octava i així aconseguir un resultat millor.

Una altre posible millora podria ser crear una interfície gràfica, on escollint una cançó a la entrada, es generèssin tots els arxius necessàris i directament s'escoltés la veu cantada.

Capítol 7.

Referències.

Bibliografia.

1. Bonafonte et al. Actividades del TALP en el _area de conversi_on de texto a habla. I Jornadas en Tecnolog__a del Habla, Sevilla, 2000.

2. Bonafonte, I.Esquerra, A. Febrer, J.A. Rodríguez Fonollosa, F. Vallverdú, "The UPCText-to-Speech System for Spanish and Catalan", Proc. of the 5th International Conferenceon Spolen Language Processing, ICSLP'98, Sydney 1998.

3. X. Huang, A. Acero, H.W. Hon. Spoken Language Processing. A Guide to Theory, Algorithm and System Development, Prentice Hall, 2001.

4. B. Gold, N. Morgan. Speech and audio signal processing: processing and perception of speech and music, Wiley, 2000.

5. L. Hernández Gómez, F. J. Caminero Gil, C. de la Torre Munilla, L. Villarrubia Grande. Estado del arte en Tecnología del Habla. Universidad Politécnica de Madrid, Telefónica Investigación y Desarrollo.

Llocs web.

6. http://tcts.fpms.ac.be/synthesis/introtts.html

7. www.atlas-cti.com

8. www.naturalvox.com

9. www.loquendo.com

10. www.readspeaker.com

11. www.aculab.com

12. www.cplusplus.com

13. http://speech.bme.ogi.edu/tts/flinger/

14. http://cslu.cse.ogi.edu/tts/research/sing/sing.html
15. http://cslu.cse.ogi.edu/tts/demos/index.html#sing
16. http://www.iua.upf.es/mtg/cat/
17. http://tcts.fpms.ac.be/synthesis/mbrola.html
18. http://tcts.fpms.ac.be/synthesis/mbrolign/
19. http://www.cstr.ed.ac.uk/research/projects/festival/
20. http://www.jyu.fi/musica/miditoolbox/
21. http://www.expressive-speech.net/
22. http://www-2.cs.cmu.edu/People/awb/talks/index.html