Universidad Zaragoza
1542

Trabajo Fin de Grado

Comparación e implementación de técnicas de Beat-Tracking para la ayuda del aprendizaje musical

Comparison and implementation of Beat-Tracking techniques to aid musical learning

Autor

Daniel Saiz Azor

Director

Juan Pablo Martínez Cortés

Escuela de Ingeniería y Arquitectura. Universidad de Zaragoza

2020/21

Comparación e implementación de técnicas de Beat-Tracking para la ayuda del aprendizaje musical

RESUMEN

En este trabajo se comparan 3 algoritmos de medida del pulso musical, que implementan diferentes técnicas de procesado digital de señal, mediante una base de datos de tracks o pistas elaborado para ser una tarea complicada para dichos algoritmos. La comparativa se trata mediante herramientas estadísticas y se centra en 3 aspectos principales: qué algoritmo obtiene mejores resultados de obtención del pulso musical, qué algoritmo obtiene los mejores resultados en cada uno de los géneros utilizados y qué algoritmo alcanza la mejor actuación de las pistas con mayor o menor predominancia de frecuencias bajas (usualmente, instrumentos de percusión, bombo y caja). También se aborda una comparativa entre los algoritmos seleccionados y anotadores humanos. La conclusión finaliza con la elección del mejor de los 3.

El trabajo se completa con la implementación de uno de los algoritmos en un ESP32, un microcontrolador de altas prestaciones, muy compacto y barato. El prototipo basado en el ESP32, se comunicara con el ordenador y el algoritmo vía puerto USB y con un periférico LED, mediante SPI (método de comunicación serie). Este suplirá la función de representar el ritmo musical de la canción que se reproduzca en tiempo real, creando así un apoyo musical para estudiantes.

Índice

1. Introducción

1.1 Motivación

Desde hace milenios existe el concepto de música. La música es el **arte** de combinar sonidos y silencios respetando los principios fundamentales de la **melodía**, la **armonía** y el **ritmo** y que, además, sea agradable para el oído. Por otro lado, es también una manifestación artística y cultural de pueblos, zonas demográficas, corrientes políticas o tribus urbanas a la que van asociados múltiples factores externos tales como la estética, la forma de vivir o las referencias que pueden llegar a definir un determinado contexto sociocultural.

Miles de autores, músicos, productores, intérpretes y cantantes dedican su vida a estudiar y desarrollar esta disciplina, tanto en el ámbito académico como en el urbano, ampliando y modificando nuestro repertorio musical y nuestro conocimiento sobre teoría musical.

Es bien conocido que la música está compuesta por ciertas características que la definen, tales como el tempo o pulso, entre otras. Cuando alguien que está escuchando música y comienza inconscientemente a mover el pie arriba y abajo, está marcando el **ritmo**. Este ritmo viene definido por parámetros medibles como la posición temporal de un pulso musical o **beat**, o el intervalo de tiempo que existe entre cada uno de estos pulsos, el **tempo**. Estos dos parámetros son caldo de análisis del siguiente documento y a pesar de que existen otras múltiples características muy remarcables para estudiar, como el timbre o el tono, estas no se abordaran.

La realización de este trabajo está motivada por el tema que trata, y la finalidad es aumentar la información que existe hoy en día sobre el campo de la música, en el ámbito del pulso y el tempo.

1.2 Estado del Arte

El **Beat-Tracking** es una técnica que comenzó a desarrollarse en los años 80, con el auge de los microprocesadores y el aumento de la capacidad de cómputo. Siendo un proceso similar al que hacemos los humanos, consiste en la adquisición de información relevante de una señal melódica que, mediante técnicas de procesamiento, se pretende obtener la localización temporal de los pulsos. Esto permite, a posteriori, dar herramientas para los propios músicos o crear sincronismos con elementos eléctricos respectos de los pulsos de las canciones, que es el objetivo final de este documento.

Muchos fueron los que se embarcaron en el desarrollo de un algoritmo de Beat-Tracking/Tempo Extraction. Existen grandes nombres en el campo como son *Povel* y *Essens* (1985) [1], que comenzaron con patrones rítmicos puramente simbólicos, analizándolos buscando la estructura métrica que mejor encajase con el ritmo, o *Miller* (1992)[1], que partió de otra idea diferente, utilizando un conjunto de osciladores, a los que aplicando la señal y observando la respuesta, escogía la que devolviese una mayor amplitud (superposición de

ondas). Dos años después, *Large* y *Kolen* (1994) [1] seguirían desarrollando esta técnica. *Dixon* y *Cambouropoulos* (2000) [1], experimentarían con las ventajas de los sistemas MIDI.
Por otro lado, se había propuesto la idea de centrar la atención en los sonidos característicos de un pulso musical, el bombo y caja, pero esto conlleva al gran problema de que ciertas canciones no optan por añadir dichos instrumentos a la mezcla. No fue hasta 1998 que *Scheirer* [1] obtuvo el primer algoritmo real de Beat-Tracking, con una implementación de cálculos de envolventes multibanda en la señal de entrada que, a posteriori, serian introducidas a un banco de resonadores. *Dixon* mejoraría dicha técnica y añadiría una etapa inicial de detección de inicios.

De aquí en adelante se desarrollarían estas técnicas con distintos métodos, desde la visión del procesado digital de señal, mediante espectrogramas, algoritmos de máximos locales, flujo espectral entre otros, hasta la visión por inteligencia artificial, mediante redes convolucionales, que no será objetivo de este trabajo, pero que cabe remarcar que es un importante avance de la técnica hoy en día.

1.2 Objetivos principales

Los objetivos principales del trabajo son 3:

- Comparar tres algoritmos de Beat-Tracking, lo que permitirá conocer cuál de ellos desempeña mejor su función.
- Comparar los resultados de los algoritmos con los de anotadores humanos.
- Diseñar un prototipo funcional que utilice la funcionalidad de Beat-Tracking del algoritmo con mejores resultados.

1.3 Organización del trabajo

Este trabajo se divide en dos partes principales. La primera es una comparativa de los tres algoritmos más interesantes basados en procesado digital de señal. Esta comparativa se basa en el índice **P-score** [1], en medidas estadísticas basadas en verdaderos positivos, falsos negativos y falsos positivos, tales como el **recall** o la **precisión** o en histogramas, permitiendo valorar así, los resultados de los algoritmos respecto de la verdad de referencia o *ground truth*, apuntada por músicos cualificados.

La comparativa de algoritmos se divide en 3 fases. La primera se centra en ver diferencias directas entre los 3 algoritmos en base a la puntuación en conjunto que obtengan en cada uno de las pistas, de la que se puede extraer una idea general. Además, se realiza un análisis del error que comete cada algoritmo y se ejemplifica la posible causa con un histograma.

La segunda está dirigida a tener una visión más amplia. Primero se divide la base de datos por géneros musicales y posteriormente, se evalúan individualmente mediante el índice P-score, con lo que se podrá extraer mayor información sobre como los algoritmos tratan cada una de las características que definen cada género musical.

Y finalmente, la tercera hace una diferenciación entre las 30 pistas con mayor y menor predominancia de bombo y caja, consiguiendo información sobre lo útil o no que puede resultar para la determinación precisa del tempo y los instantes de los beats.

De forma consecutiva a la evaluación, se procede a determinar el algoritmo con mejores prestaciones y el que mejor convenga para la implementación física, que podría no ser el mismo. En la figura 1 se muestra un diagrama de bloques básico del prototipo implementado. El microcontrolador actúa como cerebro de la operación, procesando las operaciones lógicas, estableciendo comunicación entre el ordenador y el mismo y comunicándose mediante SPI con el periférico LED.

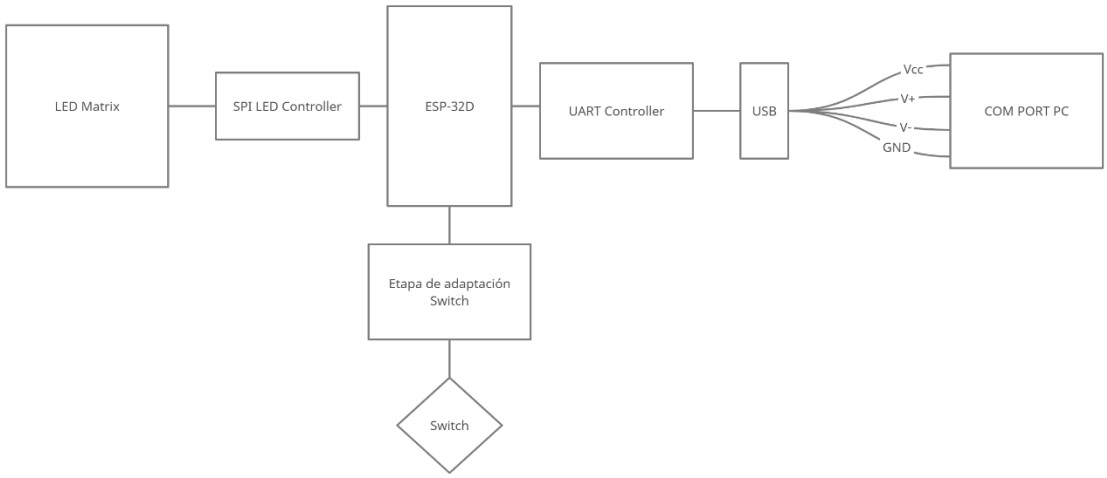

Figura 1. Diagrama de bloques explicativo sobre el conexionado básico y construcción a nivel esquemático del prototipo.

2. Algoritmos

La selección de algoritmos se ha basado en 3 **características** relevantes, ordenadas a continuación.

- **Procedimiento basado en el procesado digital de señal.**

  Se han descartado, por ello, los algoritmos basados en redes neuronales, como el realizado por Sebastian Böck [2].

- **Disponibilidad del algoritmo**

  Debido a que es necesario el uso del algoritmo, se han descartado todos aquellos que no estuviese disponible su código fuente.

- **Calidad de documentación**

  La documentación que existe en algunos casos es deficiente y no permite entender el método que siguen los algoritmos, por ello, se ha elegido aquellos que tengan una documentación relativamente buena.

Tras realizar una revisión bibliográfica amplias, se eligieron los siguientes 3 algoritmos, que cumplen con los 3 requisitos anteriores. En el Anexo 1 se hace un resumen del funcionamiento de cada uno de ellos.

- **BeatRoot.** (De ahora en adelante, BR)
- **IBT - INESC Porto Beat Tracker.** (De ahora en adelante, IBT)
- **Ellis.** (De ahora en adelante, ELL)

Una breve introducción de características de estos algoritmos se muestra en la tabla 1.

		BeatRoot	IBT	Ellis
Lenguaje		Java	C++ (Marsyas)	Matlab
Métodos principales		Flujo Espectral IOI Piscina de Agentes	Flujo Espectral mejorado Autocorrelacción Piscina de agentes	Envolvente de la fuerza de comienzos de eventos Bandas de Mel
Modo de aplicación		Offline	Tiempo real y offline	Offline
Características de la señal procesada	Tamaño de la ventana	2048 muestras	1024 muestras	256 muestras
	Superposición	50%	78.5%	87.5%
	Tipo ventana	Hamming	Hamming	Gausiana

Tabla 1 Resumen de características y parámetros más importante de los 3 algoritmos. Ampliado en el Anexo I.

3. Base de datos

Necesariamente, a la hora de comparar algoritmos se necesitan una serie de datos asociados a la tarea en cuestión. Como en muchas otras disciplinas, la adquisición de dichos datos con cierta validez y rigor es una tarea muy compleja y que es, incluso, un trabajo tan extenso como el estudio delos propios algoritmos.

En este documento no se aborda la creación de un set de datos, sino que se utiliza uno predefinido con una ligera modificación, ya que es una buena práctica tratar datos cualificados y respaldados por la comunidad.

La **base de datos** [3] obtenidos por 5 investigadores de instituciones como la universidad Pompeu Fabra, el INESC y la facultad de ingeniería de Porto está constituido por una serie de **217 extractos** musicales de múltiples géneros, características y tempos diferentes, que se construyó específicamente para desafiar a los algoritmos, basándose en una técnica de machine learning, el "**Selective Sampling**"[3], donde se comparan los resultados de diferentes algoritmos del estado del arte para verificar la dificultad de dichos extractos.

La parte más costosa e incierta de la construcción de una base de datos es la definición de una **verdad de referencia** con la que podamos trabajar a posteriori. En este caso, esta verdad de referencia se definió a partir de 5 anotadores expertos músicos.

En este trabajo se ha ampliado la base de datos con el objetivo de introducir un género más ("*progressive house*") a esta y completar los datos de otro("*experimental music*") , compuesto por 17 extractos musicales. Las anotaciones manuales las obtuvo el autor de este trabajo mediante un script escrito en Python que reproducía la canción y contabilizaba el tiempo en el que se pulsaba una tecla, marcando así, un pulso. Apara comprobar la fiabilidad de las anotaciones, se compararon los tempos obtenidos con la información disponible en la red sobre estas canciones [4].

Base de datos					
Extractos originales	**Extractos seleccionados**	**Extractos añadidos**	**Duración extractos**	**Géneros**	**Características**
217	200	17	40s	*Classical music*	Poco rítmico Tempo variable
				Guitar Solo	Tempo muy variable
				Experimental	Rítmico Bombo muy marcado
				Jazz	Rítmico Instrumentos variados
				Progresive House	Muy rítmico Bombo muy marcado
				Soul	Predomina la voz
				Chanson	Predomina la voz

Tabla 2 Características de la base de datos obtenida. Se hace una diferenciación por géneros y se describen brevemente las características más relevantes para este trabajo.

4. Extracción pulsos y tempos y comparación de algoritmos

4.1. Automatización de la extracción de pulsos y tempos

Una vez obtenidos los códigos fuente o ejecutables de los algoritmos seleccionados y la base de datos con la que se probaran y evaluaran, el siguiente paso a realizar es el de obtener los resultados de cada algoritmo para cada extracto.

Dados 217 extractos y 3 algoritmos, el procedimiento sería muy lento si se hubiera hecho manualmente, por lo que se prefirió automatizarlo. Se usaron una serie de **scripts** para dicha tarea.

4.2. Desarrollo de Aplicación Web para la obtención de anotaciones manuales

Uno de los objetivos perseguidos de este trabajo es comparar las capacidades humanas para hallar el ritmo musical. Para obtener dichos datos, se ha realizado una anotación de los pulsos que cada persona percibe de la canción que está sonando en tiempo real. Para ello, se elaboró una aplicación web en donde cada persona pueda acceder por internet y realizar los 15 extractos propuestos, extraídos aleatoriamente de la base de datos.

Se ha planteado como un "juego", donde cada persona que acceda obtendrá una puntuación basada en sus resultados, de manera que podrá competir contra sus amigos. La aplicación ha sido realizada con la librería **StreamLit** y está alojada en el repositorio de **GitHub** del autor y accesible a través de **Heroku**.

Aunque la aplicación web está totalmente operativa en la red, no es capaz de guardar los datos obtenidos por cada usuario por lo que se utilizó únicamente de forma local, es decir, los anotadores accedían a la aplicación web desde un ordenador donde se guardaban los resultados. La aplicación web está alojada en el siguiente enlace *https://tempo-tracking-app.herokuapp.com/*

4.3. Comparación Beat-Tracking

Con todos los datos obtenidos, una vez ordenados y filtrados los posibles resultados nulos, se procede con la comparativa. Es una parte importante donde se debe centrar la atención en qué técnica o técnicas se van a utilizar.

Tanto los resultados obtenidos como la verdad de referencia están compuestos por una serie de tiempos que representan el momento en el que ocurre el pulso. Se puede imaginar dichos resultados como un tren de pulsos con valor 1 en el instante marcado por el pulso y 0 en todos los demás.

Por supuesto, se debe entender la necesidad de que generar este tren de pulsos referido al tiempo necesita de una frecuencia de muestreo. En este caso, se utiliza una Fs de 100Hz, que permitirá tener una resolución del orden de la centésima del segundo. Es una resolución

suficiente teniendo en cuenta la incapacidad del oído humano para detectar como diferentes 2 tonos con menor separación que 70ms [5].

Una vez definidos los datos, se realiza la comparativa mediante la ecuación 1, el **P-Score**, que define una correlación entre los datos de la verdad de referencia, $a_s[n]$ y los obtenidos por los algoritmos $y[n]$ dentro de una ventana de tolerancia de longitud W, extrayendo así, el número de coincidencias entre los pulsos reales y los detectados.

La ventana de tolerancia, W se utiliza ya que se considera verdadero positivo que el algoritmo detecte un pulso ligeramente desplazado en el tiempo respecto de la verdad. Teniendo en cuenta las características explicadas del oído humano, esta tolerancia se ha fijado en 70ms.

$$P = \frac{1}{F} \sum_{m=-W}^{W} \sum_{n=1}^{N} y[n] * a_s[n-m] \tag{1}$$

El parámetro F viene definido por la ecuación 2, y sirve como normalizador,

$$F = \max\left(\sum_{n} y[n], \sum_{n} a_s[n] \right) \tag{2}$$

4.3.1. Comparativa entre algoritmos

Para comenzar esta evaluación se va a utilizar, como ya se ha explicado, el índice P-score, para los distintos subapartados.

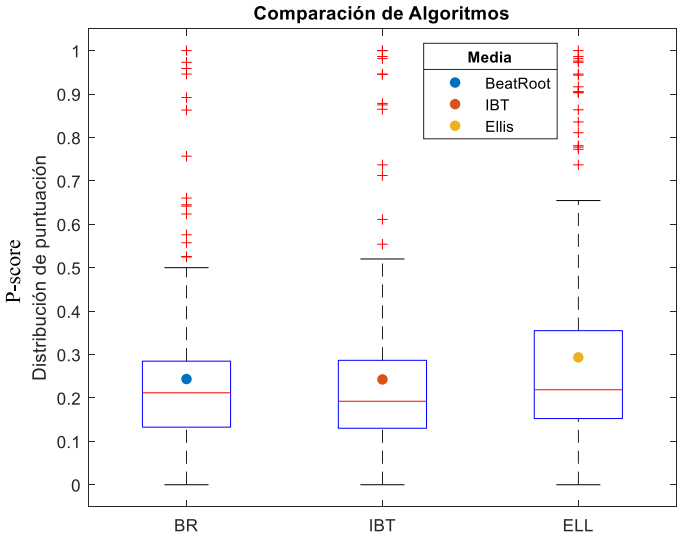

Figura 2 Distribución de las puntuaciones P-Score de cada uno de las 217 pistas en cada uno de los algoritmos, visualizado mediante un diagrama de cajas.

Una visión general sobre el funcionamiento de cada algoritmo por separado respecto de todo el set de datos puede ser un resultado interesante para tener una idea inicial. Es por ello que, en la figura 2, se refleja la evaluación con el P-Score de los 3 algoritmos elegidos.
Como se aprecia en el diagrama de cajas, donde coexisten la media (punto central), mediana (línea centrada dentro de la caja), cuartiles 1 y 3 (límites de la caja) y valores atípicos (puntos externos), los valores obtenidos son más bien bajos.

Tanto BeatRoot como IBT, conformados bajo la misma técnica, tienen una puntuación muy similar, mientras que Ellis, aun con una puntuación ligeramente mayor, tampoco presenta valores altos.

Hay un elemento que sorprende y son la cantidad de valores atípicos que se escapan de la distribución. Esto nos indica que hay unas pistas con ciertas características a las cuales los algoritmos propuestos tienen más facilidad a la hora de estimar los pulsos.

Aun así, no se puede fijar un mejor método a partir de esta comparación general. Dada la ecuación del P-Score puede suceder un problema que no se trata con esta técnica, en la que los algoritmos obtengan buenas puntuaciones porque estén estimando un número mayor de predicciones respecto a la verdad. Una visualización rápida de los valores de precisión y recall (figura 3 y 4) permite ver que puede estar ocurriendo esta situación por el alto recall y la baja precisión que se obtiene. El algoritmo de Daniel Ellis es el que mejores resultados obtiene, con el recall y la precisión más alta, es decir, es el que más pulsos verdaderos consigue estimar. Aun así, es muy probable que estime más pulsos de los que de verdad hay.

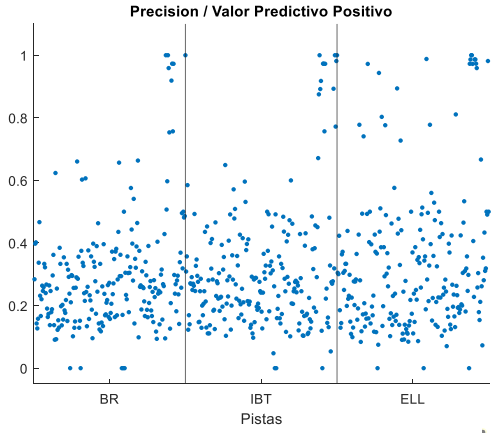

Figura 3 Valores de precisión de cada pista en cada algoritmo. $\frac{VP}{FP+VP}$

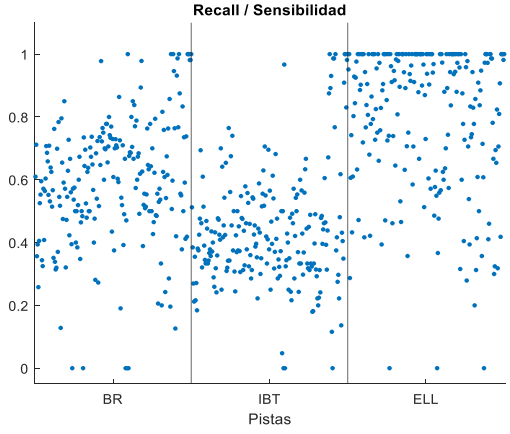

Figura 4 Valores de recall de cada pista en cada algoritmo. $\frac{VP}{FN+VP}$

	BeatRoot	IBT	Ellis
Media Precisión	0.2939	0.3095	0.3327
Media Recall	0.5933	0.4420	0.7938

Tabla 3 Resumen de los datos obtenidos de las figura 3 y 4. Valores de puntuación entre 0 y 1.

Si se sigue tirando del hilo se debería ver con qué ratio dan sus resultados, es decir, ya se ha comprobado que al menos Ellis estima de forma continuada más pulsos que los otros 2 algoritmos pero, ¿Cuánto más? o ¿Cuánto más que la verdad?

Es una realidad (que se comprueba en el apartado 4.5) que las personas tienden a obtener valores del pulso musical a un múltiplo del tempo correcto, con valores como x2, x3, x$\frac{1}{2}$ y x$\frac{1}{3}$. Algo parecido ocurre con los algoritmos y que IBT, por ejemplo, tiende a controlar con un valor límite, modificable por programa, de tempos máximo y mínimo.

En la figura 5 se muestra información más precisa sobre esta cuestión, donde se observa la distribución de tempos que cada algoritmo ha estimado de cada pista y la distribución de la verdad de referencia, en pulsos por minuto (de sus siglas en inglés, BPM).

Un apunte relevante que confirma los hechos anteriormente expuestos es la distribución del algoritmo Ellis, el cual estima valores muy por encima de la realidad. Por ejemplo, Ellis estima algún extracto musical por encima de los 800BPM, mientras que el extracto con mayor tempo real es de 201BPM (ratio de 4). Por otro lado, BeatRoot también cuenta con esta problemática, tendiendo a estimar a x2 una gran cantidad de pistas. En este aspecto, el algoritmo que mejor funciona es IBT, cuya distribución es la más parecida a la verdad de referencia. Aun así, sigue la tendencia de resultados no muy buenos que se llevan observando en todo el trabajo.

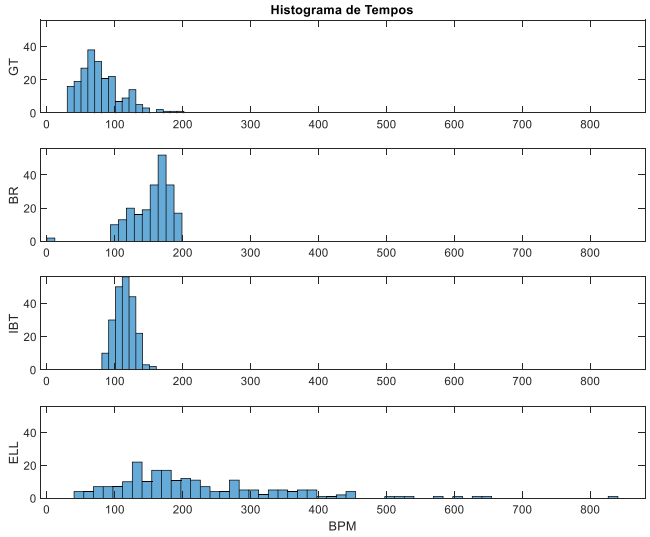

Figura 5 Histograma de las medias de los tempos obtenidos en cada pista mediante cada algoritmo. El tempo se ha obtenido como la media de todos los tiempos. GT: Ground Truth (Verdad de Referencia)

Otro aspecto relevante de la caracterización de los algoritmos viene dado por el momento en el que estiman el pulso, es decir, si lo hacen antes, después o en el momento exacto y se debe medir a lo largo de todos los pulsos del extracto. Un algoritmo muy bueno podría ser aquel que, aun no estimando los pulsos exactamente en su valor real, los estimase sistemáticamente un cierto tiempo antes o después, siendo este tiempo un valor pequeño respecto al valor del tiempo entre pulsos. Este concepto cobra validez debido a que el anotador que define la verdad de referencia no sea lo suficientemente preciso para definir el pulso en el momento justo, ya sea por tiempo de reacción como por tiempo mecánico de pulsación de su dedo

Por último y para finalizar este apartado, se ha considerado el error a la hora de obtener el pulso, como un elemento importante. Por supuesto, cuando un algoritmo infiere un resultado, puede haber 4 posibilidades, que obtenga el valor correcto, que obtenga un valor aproximado al real pero con una cierta distancia de error con la que pueda seguir considerándose correcto, que obtenga uno no correcto y, la situación que no se puede tratar, que obtenga el valor correcto de la verdad de referencia pero que esta verdad de referencia sea errónea.

En la figura 6 se muestra una cuantificación de los errores cometido por cada uno de los algoritmos en cada pista musical, expresado en segundos. Un algoritmo perfecto sería aquel que, aun obteniendo un error medio no nulo (por supuesto, no mayor del rango de los 70ms),

su desviación estándar fuese mínima, lo que querría decir que estima la posición de los pulsos con un error sistemático (el mismo para todos los pulsos).

Cabe resaltar que la estadística del error se ha caracterizado con la media aritmética, para dar información sobre si el algoritmo tiende a adelantarse o retrasarse del pulso, pero esto conlleva a mediar errores de diferente signo que pueden llegar a anularse. Aun así, la desviación estándar aporta esa información adicional que permite conocer sobre si ha habido mucha o poca dispersión alrededor del error medio.

Resultado común a los 3 algoritmos es el alto valor de sus deviaciones estándar, denotando una baja estabilidad a la hora de estimar el pulso, es decir, presenta errores positivos y negativos con una varianza importante.

Un resultado a recalcar, en la figura 6, es el obtenido por IBT en la pista 191, correspondiente a una canción de jazz, que obtiene un valor medio de 60ms de error con una desviación estándar de únicamente $\pm$18ms.

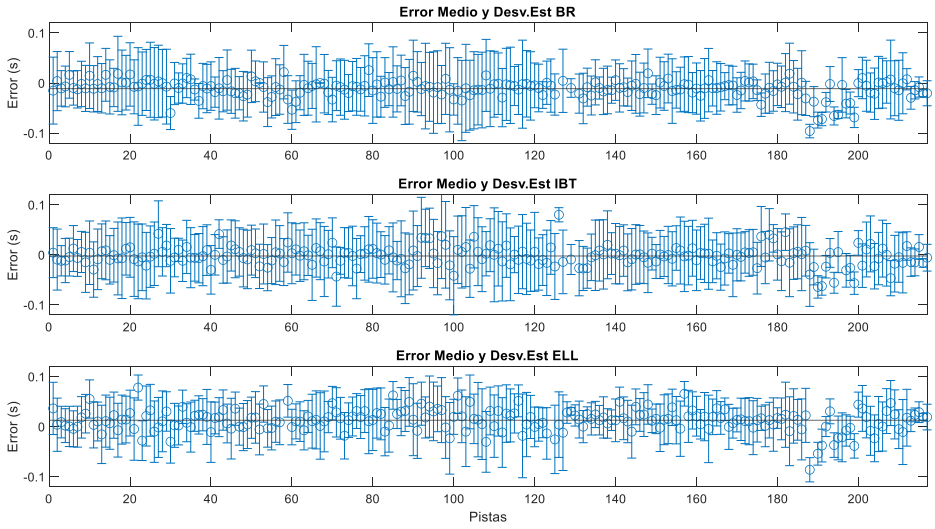

Figura 6 Error medio y desviación estándar (en segundos) de las predicciones en cada uno de las pistas de la base de datos. Resultados obtenidos con la técnica de la ecuación 9.

	BeatRoot	IBT	Ellis
Media global y desviación estándar de los errores (s)	-0.0113$\pm$ 0.0482	-0.0023 $\pm$ 0.0533	0.0128 $\pm$ 0.0435
Nº de pistas en los que tiende a adelantarse	44	102	162
Nº de pistas en los que tiende a retrasarse	168	111	52

Tabla 4 Resumen de los datos obtenidos de la figura 6. Valores positivos definen adelantos y valores negativos, retrasos. Los errores se miden en segundos. La suma de las pistas que adelantan o retrasan no dará la totalidad de pistas debido a que los algoritmos no han logrado obtener datos sobre ellos en base a su dificultad.

4.3.2. Comparativa por géneros

El siguiente paso dentro de la comparativa se basa en diferenciar los resultados entre los distintos géneros.

Los algoritmos se basan en técnicas de procesado de señal capaces de reconocer, analizar y definir resultados en base a, principalmente, frecuencias y ganancias que caracterizan los extractos del set de datos. Es por ello que la distinción por géneros es un indicador muy real de la eficacia debido a que cada uno está definido por una serie de características musicales. Por ejemplo, el *progressive house*, genero de música electrónica muy rítmico, compuesto por un patrón bombo-caja muy marcado y secuencias melódicas periódicas, o la *música clásica*, conformada por conjuntos de instrumentos de diferentes clases que componen secuencias que evolucionan en el tiempo, con o sin patrones, con o sin ritmos marcados y con cambios en la amplitud del sonido.

Esta no simultaneidad es la que dificulta la acción de los algoritmos a la hora de reconocer el ritmo, y se observa claramente en la figura 7, donde se comprueba la facilidad que tienen a la hora de detectar géneros musicales más rítmicos, como es el *progresive house*, mientras que no ocurre los mismo en otros géneros como la *música clásica* o el *jazz*.

En conclusión, los 3 algoritmos predicen mejores resultados de géneros con patrones más rítmicos, y con menores cualidades cambiantes en el tiempo.

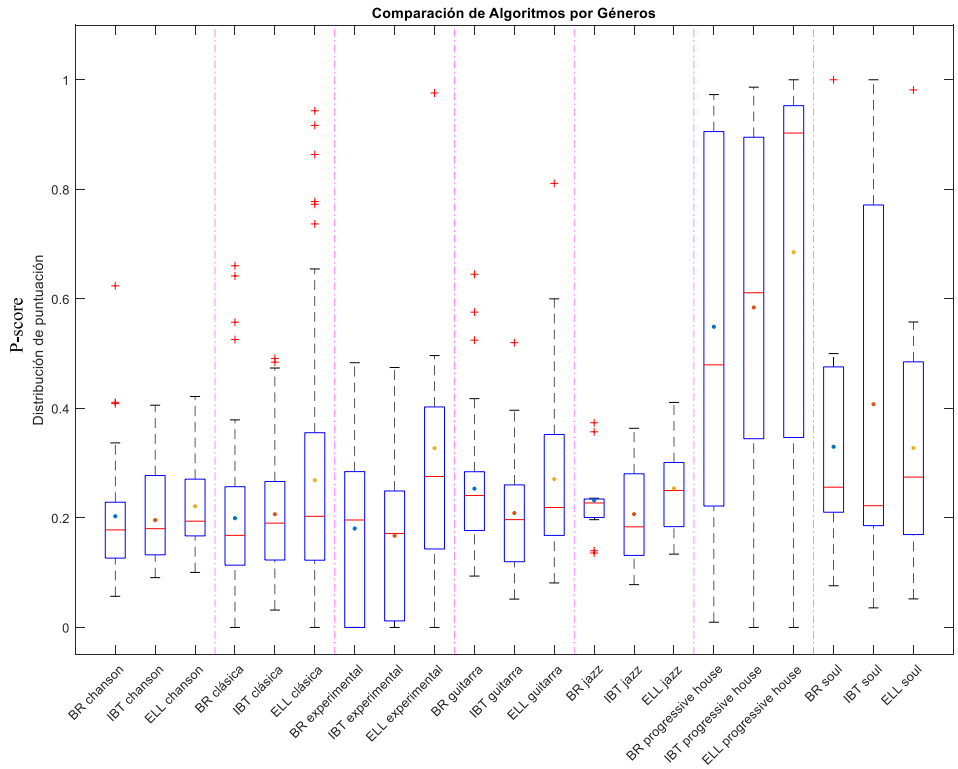

Figura 7 Distribución de valores P-Score diferenciados por algoritmos mediante las barras punteadas y representados en diagrama de cajas. Los valores de las medias son los puntos en azul, para BeatRoot, en naranja para IBT y en amarillo, para Ellis. Están agrupados de 3 en 3 por géneros.

4.3.3. Comparativa por pistas con mayor predominancia de bombo y caja

Siguiendo con la dinámica del apartado anterior, donde se comprueba que los algoritmos obtienen mejores resultados con características no cambiantes, se propone la hipótesis que valores rítmicos marcados permiten un buen funcionamiento del algoritmo, por lo que el objetivo de este apartado va a ser intentar validar dicha hipótesis.

Este pensamiento extrapolado al ámbito humano es la realidad. Si un usuario cualquiera escucha una canción con un sonido característico de tambor de forma periódica, va a intuir que ese es el ritmo de la canción, es decir, los propios instrumentos dan la información sobre lo que se busca. Es por ello que se ha propuesto realizar una diferenciación entre los pistas con una mayor predominancia de bombo y caja por ser los elementos más representativos en las canciones a la hora de marcar el pulso.

Se han seleccionado las 30 pistas con la mayor y la menor predominancia de bombo y caja de la base de datos y se ha representado la puntuación obtenida mediante el P-Score en cada uno de los dos grupos, en la figura 8.

Con estos resultados, se comprueba la hipótesis. Tanto Ellis como IBT son capaces de obtener un resultado perfecto en alguna de estas pistas con el bombo muy predominante y, a su vez, BeatRoot obtiene, aunque no unos resultados tan buenos, una considerable mejora tanto respecto del global de los datos, como respecto de las 30 pistas con menor predominancia (Tabla 5).

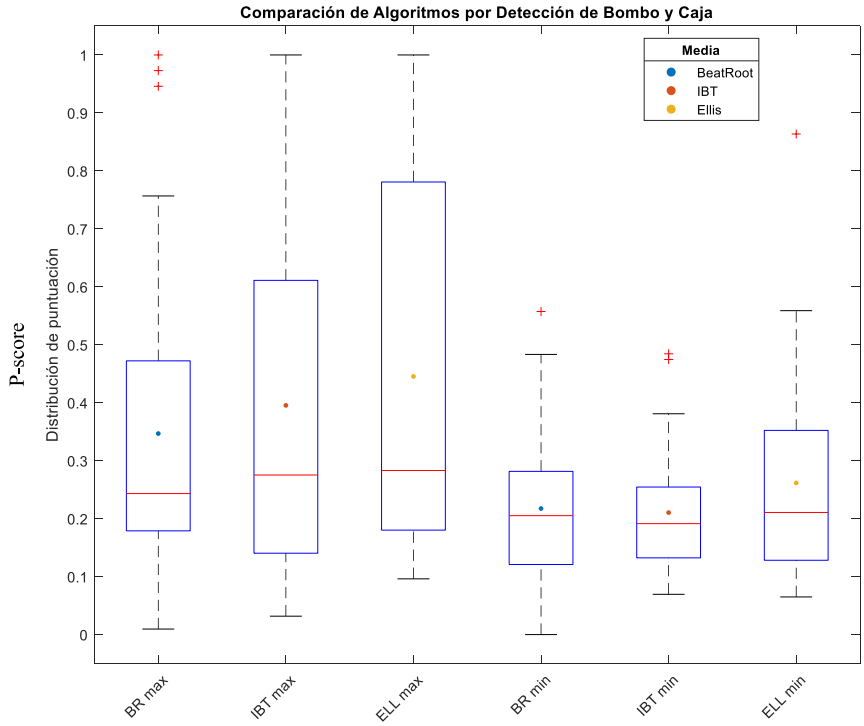

Figura 8 Distribución de puntuaciones P-Score de la evaluación de cada algoritmo sobre 30 pistas con mayor y menor predominancia de bombo y caja.

	BeatRoot	IBT	Ellis
Media y desviación estándar P-Score. Pistas con mayor predominancia	0.3468 ± 0.2652	0.3954 ± 0.3136	0.4453 ± 0.3285
Media y desviación estándar P-Score. Datos globales	0.2435 ± 0.1737	0.2425 ± 0.1880	0.2934 ± 0.2182
Media y desviación estándar P-Score. Pistas con menor predominancia	0.2174 ± 0.1255	0.2104 ± 0.1116	0.2616 ± 0.1721

Tabla 5 Resultados de las medias y desviaciones estándar obtenidas de la figura 8. Se visualiza de forma fácil la mejora de los resultados cuando los datos tienen instrumentos rítmicos más marcados.

4.4. Comparación Tempo

Los algoritmos de beat-tracking no solo se aplican para determinar la localización de los pulsos sino que algunos estiman métricas de tempo de cada pista. Es el caso de IBT, que además, permite obtener el resultado en valores de mediana, media… entre otros.

Esta característica no es común a los 3 algoritmos, solamente a IBT y a Ellis. BeatRoot no permite de primera mano la obtención de tempo, por eso se ha decidido trabajar con calculados a partir de los intervalos medios entre los pulsos detectados por su sistema de Beat Tracking.

4.4.1. Comparativa entre algoritmos

El método para comparar la estimación de tempo en los algoritmos es más simple, ya que solo contamos con un dato por pista, el **tempo**, y la puntuación dada a cada algoritmo será mejor cuanto más cerca esté de la realidad definida por el anotador experto del set de datos.

En este caso, la forma de puntuar se basara en un sistema lineal (3), diseñado para este trabajo, que devolverá una puntuación en base a la lejanía del valor de tempo estimado respecto del real.

$$Puntuación = \begin{cases} \frac{-1}{FC * GT} * |VP - GT| + 1, & \frac{|VP - GT|}{GT} \leq FC \\ 0, & otro\ caso \end{cases} \quad (3)$$

Donde VP es el valor estimado por el algoritmo, GT es la verdad de referencia (de su forma en inglés, Ground Truth), y FC es factor de confianza. Este valor FC es el error relativo respecto del cual se va a asignar una puntuación de 0. Cuanto mayor sea, mayor error se permitirá al algoritmo sin asignarle una puntuación nula.

Por ejemplo, para una pista de 128BPM a la que se le supone un CF del 10% donde un algoritmo ha estimado un resultado de 135BPM, es decir, un error en valor absoluto de 7, se obtendría la gráfica que aparece en la figura 9. La puntuación obtenida será de 0.454 sobre 1.

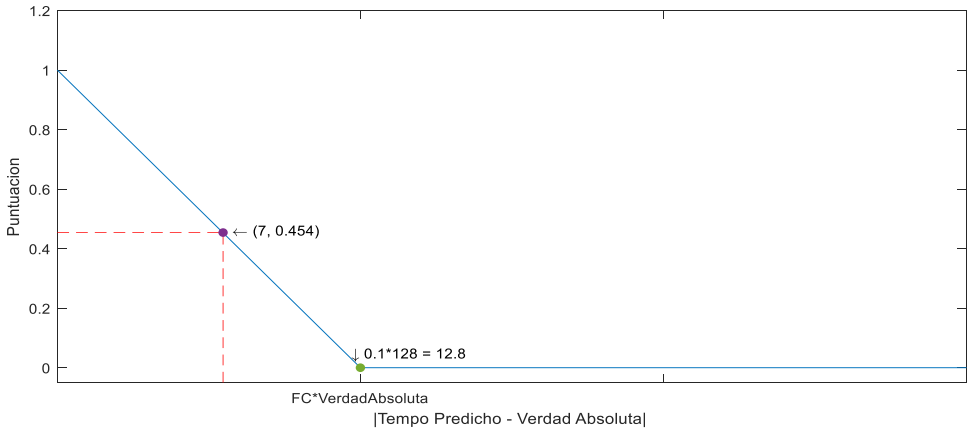

Figura 9 Representación gráfica de la función a trozos variable para el caso concreto de evaluación de una pista con un tempo real de 128 BPMs. El punto morado corresponde al error del resultado estimado por el algoritmo mientras que el punto verde correspondo al final del umbral de puntuación.

Se parte de la misma forma de actuar que en el apartado 4.3. Se va a realizar una comparativa inicial de los 3 algoritmos para ver unos primeros detalles sobre cuál puede estar dando mejores resultados.

En la figura 10 se muestran los resultados de las puntuaciones de los 3 algoritmos con cada una de las pistas evaluadas con la técnica anterior y con 5 diferentes factores de confianza, para poder ver cómo evolucionan los resultados.

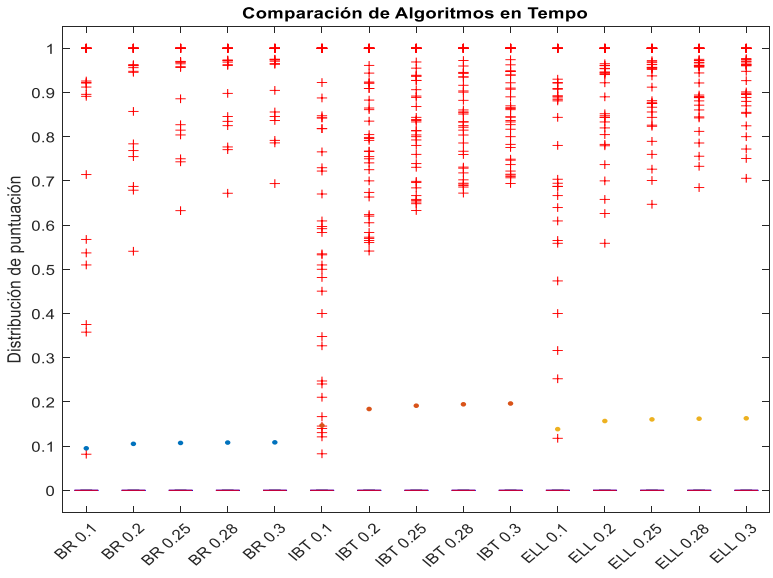

Figura 10 Diagrama de cajas de la distribución de puntuaciones con el sistema lineal de puntuación. Las medias de la distribución se expresan con puntos de color azul (BeatRoot), naranja (IBT) y amarillo (Ellis). FC esta entre 0.1 y 0.3 porque definen un rango de error posible entre los que se encuentran los 70ms explicados anteriormente.

Lo que se puede observar de dicha figura es como la mediana es 0 en todos los casos para todos los factores de confianza, lo que indica que al menos la mitad de las pistas han sido predichas con un error mayor al factor de confianza mayor, es decir, un 30% de error.

Entre los tres algoritmos, el que mejor controla esta prueba es IBT, con medias de puntuación por encima de sus competidores para todos los niveles de confianza. Estos resultados concuerdan con los obtenidos en el apartado 4.3, donde se demostraba que gran parte de los extractos se estimaban con métricas duplicadas o incluso triplicadas.

4.4.2. Comparativa por géneros

Como se ha indicado y comprobado anteriormente, características musicales como la predominancia del bombo o patrones musicales muy periódicos facilitan la actuación de los algoritmos. Estos parámetros mejoran la detección de eventos de comienzo, que es la etapa previa a toda la detección tanto de pulsos como de tempo. Por ello, la hipótesis del apartado 4.3.2 será la misma que ahora, es decir, los géneros con las características comentadas seguirán obteniendo los mejores resultados.

En la figura 11, observamos que géneros como el *progresive house* o la música *experimental* obtienen valores de la distribución relativamente altos. Sí que es cierto que el claro vencedor es el *progresive house,* con más de la mitad de pistas estimadas perfectas.

Es curioso el caso particular de la música *experimental* que a la hora de la puntuación de Beat Tracking no había obtenido grandes resultados y en este caso, ha mejorado considerablemente, sobre todo con IBT. El sistema de IBT que estima el tempo está compuesto por una autocorrelacción del flujo espectral donde los valores más altos de dicha operación se proponen como posibles hipótesis de tempo. Esto hace pensar que el sistema de autocorrelacción y detección de eventos de comienzos conjuntamente a la evolución de los agentes, funciona realmente bien para dicho género.

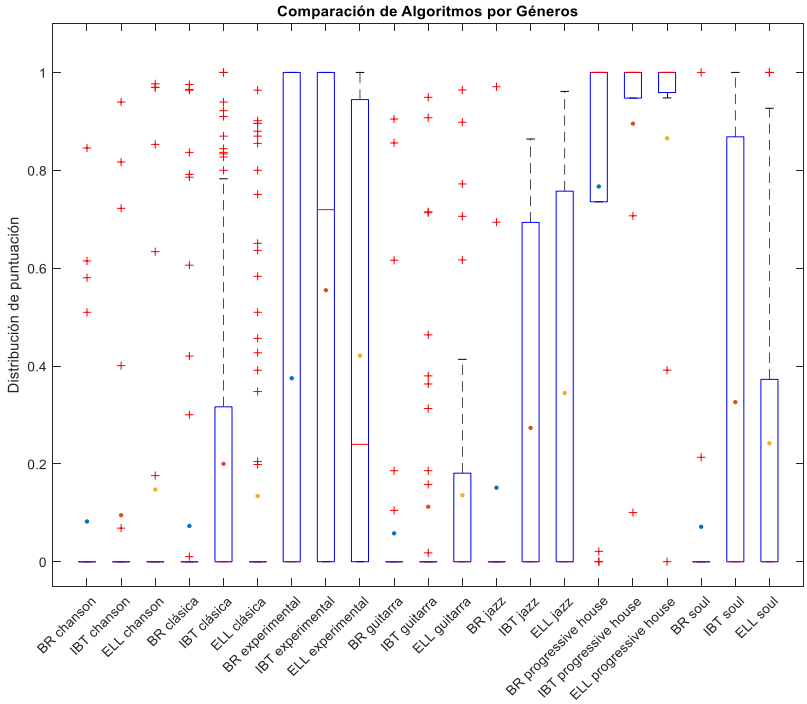

Figura 11 Distribución de puntuaciones diferenciadas por algoritmos y representados en diagrama de cajas. Los valores de las medias son los puntos en azul, para BeatRoot, en naranja para IBT y en amarillo, para Ellis. Se usa el factor de confianza mayor, 0.3.

El claro perdedor de esta prueba es el género de *guitarra*, caracterizado por cambios musicales muy abruptos, amplitudes de sonido cambiantes, silencios largos… Toda una mezcla de elementos que como se puede ver, no son nada favorables para detectar ni sus pulsos ni sus

tempos. La música *clásica* obtiene unas puntuaciones mayormente bajas debido a su relación con los aspectos descritos del género *guitarra*

4.4.3. Comparativa por pistas con mayor predominancia de bombo y caja

En cuanto a la distinción por niveles de bombo y caja, se va a repetir el mismo proceso que en el apartado 4.3.3. La figura 12 muestra los resultados.

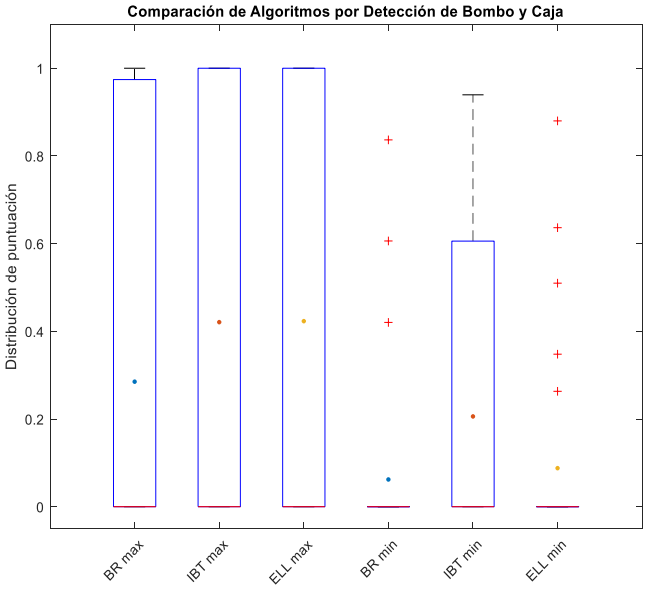

Figura 12 Distribución de puntuaciones de la evaluación de cada algoritmo sobre 30 pistas con mayor y menor predominancia de bombo y caja. Coeficiente de confianza de 0.3.

	BeatRoot	IBT	Ellis
Media P-Score. Pistas con mayor predominancia	0.2851	0.4209	0.4231
Media P-Score. Datos globales	0.1129	0.2007	0.1662
Media P-Score. Pistas con menor predominancia	0.0620	0.2058	0.0879

Tabla 6 Resultados de las medias obtenidas de la figura 12. Se visualiza de forma fácil la mejora de los resultados cuando los datos tienen instrumentos rítmicos más marcados.

De nuevo se refuerza la hipótesis de que instrumentos como el bombo o la caja, con su carácter marcador de pulso, facilitan la actuación de los algoritmos, pero en este caso, se observa como la mediana sigue en una puntuación de 0, incluso en las pistas donde predominan dichos instrumentos, es decir, como mínimo, la mitad de las pistas se han estimado más de un 30% de error, como se comentaba en apartados anteriores, pero la distribución llega a valores de puntuación perfecta (1) con relativa facilidad, y con medias cercanas a 0.5 en el caso de IBT y Ellis, que son los que mejor actúan en este caso, lo que quiere decir que los resultados están muy polarizados, o son 0 o son 1, o el algoritmo lo hace realmente mal, o el algoritmo lo hace realmente bien.

4.5. Comparación con anotadores humanos

La comparación de los algoritmos entre ellos da una visión muy buena sobre cómo actúa cada uno y permite obtener conclusiones sobre el mejor en cada aspecto pero falta una base

sobre la que apoyarse para poder definir la buena actuación de los algoritmos. Para ello, se necesita algo que permita tener una referencia a la hora de juzgar cuanto de bien se comportan. Para ello, se propone la obtención de resultados de la mano de anotadores humanos.
Van a existir 4 categorías que agrupan a los diferentes anotadores que realizan la prueba.

- Grupo A: Músicos
- Grupo B: Personas que tocan instrumentos pero que no se consideran a ellas mismas músicos.
- Grupo C: Personas que alguna vez han tocado un instrumento o tienen conocimientos musicales básicos.
- Grupo D: Personas que nunca han tocado un instrumento y que no tienen apenas conocimientos musicales.

El dato obtenido de los anotadores anónimos es el tempo, por lo que el primer resultado que se va a comprobar es la puntuación obtenida mediante la técnica explicada en el apartado 4.4. En la figura 13 se reflejan dichos resultados, donde comprobamos que en general, lo anotadores obtienen mejores distribuciones que los algoritmos, por ejemplo, la peor media, la del anotador 8 es más alta que la media de la puntación del mejor algoritmo, IBT. Con las medianas y los valores atípicos ocurre lo mismo.

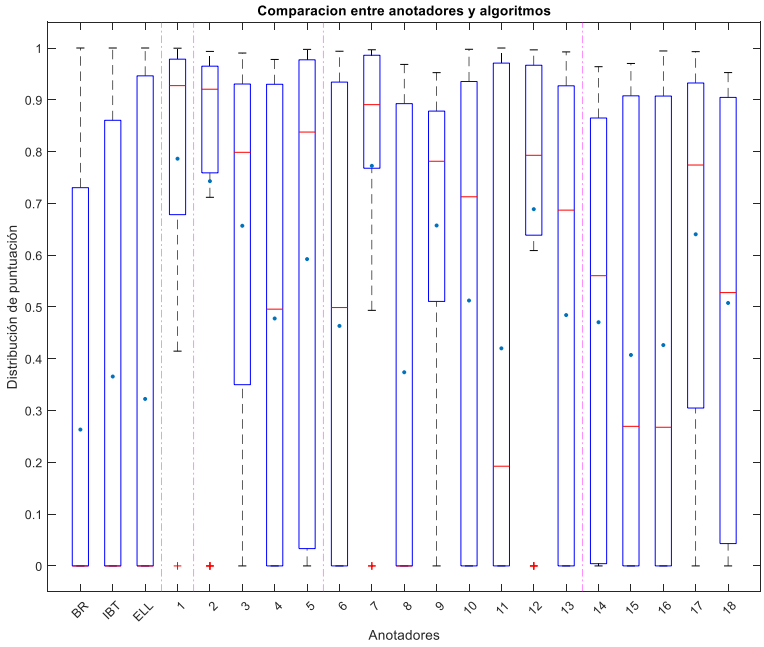

Figura 13 Diagrama de cajas referente a la distribución de puntuación de cada anotador. Se muestra la puntuación que obtienen los 3 algoritmos con los mismos 15 pistas y, a continuación, los anotadores, separados por experiencia musical por las barras punteadas.

	Puntuación media	Nº anotadores
Mejor puntuación media de los algoritmos (IBT)	0.3657	----------
Mejor puntuación media de los anotadores “Grupo A”	0.7863	1
Mejor puntuación media de los anotadores “Grupo B”	0.7429	4
Mejor puntuación media de los anotadores “Grupo C”	0.7725	8
Mejor puntuación media de los anotadores “Grupo D”	0.6403	5

Tabla 7 Puntuación media más alta de entre todos los anotadores de cada grupo (IBT).

De la tabla 7 podemos concluir que los anotadores humanos tienen un mejor desempeño en la tarea de estimar el tempo.

Además, se puede ver una clara diferenciación entre el mejor anotador de la clase A, expertos y la clase D, personas que nunca han tocado un instrumento, entre los que se encuentra una diferencia de 0.146.

La puntuación del mejor anotador de los grupos B y C es la inversa de la que cabría esperar. Aunque, en general, la capacidad rítmica pueda irse desarrollando con el tiempo y la experiencia, siempre existirán personas que sean muy capaces en esta disciplina, como ocurre en el grupo C, con el anotador número 7, que obtiene una puntuación más alta que el mejor de los anotadores del grupo B.

Una puntuación media de cada usuario es un valor representativo sobre lo bien que efectúa la tarea dicho anotador pero, por supuesto, se debe conocer el error que comete a la hora de estimar los tempos. En este apartado no se tiene en cuenta que los anotadores tiendan a estimar mayores o menores tempos, sino que se va a obtener el error absoluto para poder ver la realidad de su actuación. Esto es porque a la hora de estimar el pulso, en las personas intervienen varios factores tanto internos como externos (velocidad de respuesta mental, física, ambiente exterior, ruido, aprendizaje…) que pueden hacer variar la prueba en función del momento en que se realice haciendo que, para una misma pista, el anotador pueda estimar distintos tempos cada vez.

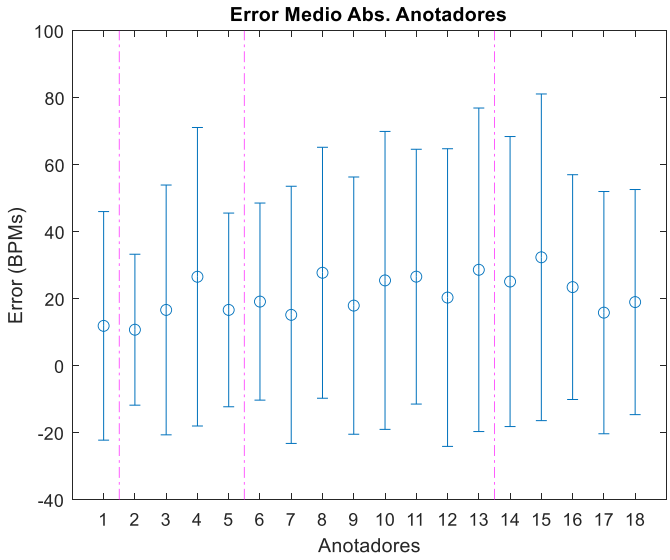

Figura 14 Error medio absoluto y desviación estándar de los anotadores en la tarea de estimar el tempo. Están diferenciados por experiencia musical de la siguiente forma, de izquierda a derecha, grupos A-B-C-D.

En la figura 14, se representa el error medio y la desviación estándar de cada anotador al estimar el tempo. La diferencia del error entre los anotadores no es muy elevada y aunque tiende a subir cuanta menos experiencia tienen estos, no es algo muy significativo. Al final, lo más representativo en dicha grafica es la desviación estándar, que interesa sea lo menor posible. El mejor resultado en este caso lo obtiene el anotador 2 (Grupo B), con una desviación estándar de ± 22.52 BPMs con una media de 10.76BPMs, lo que significa que para, por ejemplo, una pista a 128BPM, para el peor caso estaría estimando 161 pulsos cuando únicamente habrían aparecido 128, lo que equivale a un error del 25.78%, en este caso.

El objetivo que se propone ahora es el de responder a por qué las desviaciones estándar de todos los anotadores son relativamente altas.

La posibilidad de que esto ocurra puede derivar de 2 factores. El primero y algo que se vio a la hora de realizar las pruebas es que los anotadores con poca experiencia musical tienden a anotar no solo el pulso sino cualquier característica musical prominente, es decir, anotan cualquier pulso de forma no equiespaciada, aun cuando el tempo fuese constante. Esto provoco que obtuvieran unos valores medios de tempos que se alejaban del valor real. El segundo factor y como ocurría con los algoritmos, es que se tiende a estimar un tempo a un múltiplo de su valor real. En la figura 15 se visualiza como esto ocurre. Es curioso que al contrario que en el caso de los algoritmos donde por lo general duplicaban o triplicaban el ritmo, las personas tienden a disminuirlo. En concreto, la pista número 10 es un vals con un tempo de 199BPM que engaña a varios anotadores, los cuales estiman un tercio del valor real.

Para finalizar, se debe comprobar qué errores obtienen los extractos seleccionados respecto a la media del total de anotadores, es decir, cuál se les da mejor, cuál peor…

Como en todas las gráficas de error, a una menor media de error y una menor desviación estándar, una mejor capacidad de detección del tempo. En este caso (figura 16), el mejor resultado lo obtiene la pista 1, con muy baja desviación estándar, es decir, todos los anotadores han estimado prácticamente el mismo tempo. En cambio y como se veía en la figura 15, la pista 10 es la que más incertidumbre genera, y es el que ha obtenido los peores resultados.

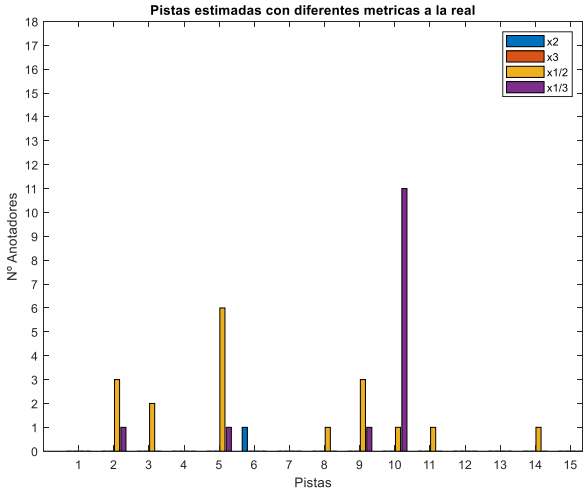

Figura 15 Representación de las pistas que han sido anotadas con un múltiplo de su valor real. Estos múltiplos son los indicados en la leyenda de esta misma gráfica. Se ha permitido un umbral de ± 7% del valor real para que los resultados de los anotadores fuesen interpretados como un múltiplo de su valor real y no como una selección aleatoria de pulsos.

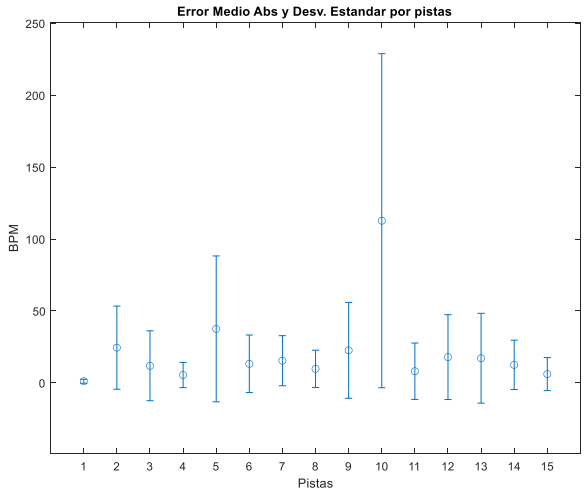

Figura 16 Representación del error medio y la desviación estándar con la que se ha estimado cada una de las pistas seleccionadas.

Por supuesto, aunque los extractos se han seleccionado aleatoriamente, se han escogido al menos 1 de cada género, por lo tanto, existen pistas de algún genero de los que se ha comprobado que para los algoritmos es más fácil, como el *progresive house*. De este caso, hay 4 pistas (3, 4, 11 y 15) que obtienen resultados de error más bajos que otros.

4.6. Conclusiones

Bajo todas las pruebas realizadas, es una realidad el mediocre desempeño de los algoritmos sobre la base de datos que se ha utilizado. Haciendo un breve repaso a los primeros apartados de cada comparativa se obtiene, en la parte de Beat-Tracking, 0.2934 sobre 1, para Ellis como mejor resultado en la valoración global o 0.1963 sobre 1 para IBT, en la misma valoración, en la parte de tempo.

Aunque las prestaciones sean relativamente mala, se comprueba en los apartados de comparativa entre géneros y predominancia de bombo y caja que las características musicales de cada pista cobran una especial importancia en el correcto funcionamiento de los algoritmos que, en cierta parte, trabajan de forma parecida a los humanos. Los algoritmos obtenían resultados muy buenos en pistas con ritmos muy marcados y patrones rítmicos que ayudasen al procesado de señal que se realiza en los algoritmos, como puede ser en la función del flujo espectral para detectar los eventos de comienzo, en BeatRoot o la autocorrelaccion para encontrar la hipótesis de tempo, en IBT.

Haciendo el cómputo global de resultados, el ganador es IBT, que sale vencedor en más de la mitad de las pruebas y que, si bien es cierto que no es el que mejor precisión tiene, sí que es el que mejores puntuaciones de tempo consigue y es el más estable, constante y no tiende a duplicar ni triplicar el tempo como sus rivales. Además, es el más flexible, pudiendo modificar ciertos parámetro numéricos, como el tiempo de la ventana de inducción y es el que más funcionalidades ofrece con sus diferentes modos de inducción. Aun así, no es ni de lejos capaz de alcanzar los resultados humanos donde incluso el peor de sus anotadores obtiene una puntuación ligeramente superior en la tarea de estimar el tempo (Figura 13).

5. Implementación del prototipo

El último objetivo de este trabajo es el de crear una herramienta tangible que pueda servir en el aprendizaje musical. La idea es motivada por el conocimiento de la existencia de ciertas personas incapaces de seguir el ritmo musical pero que quieren desarrollar su capacidad de tocar un instrumento o mejorar en el baile.

5.1. Definición del prototipo

El prototipo tiene que ser como una especie de metrónomo adaptable automático que sea capaz de automodificarse durante la reproducción de la canción.

El actual metrónomo es capaz de establecer un ritmo estático habiéndose previamente seleccionado por el músico. Cabe la posibilidad de que el ritmo de la canción pueda ir cambiando o que el usuario seleccione mal el ritmo inicialmente, haciendo que el metrónomo falle. Por otro lado, como se ha visto en el desarrollo de este trabajo, los algoritmos también yerran el ritmo, asique ninguno de estos instrumentos son infalibles.

El prototipo debe ser lo más simple posible de utilizar. Además, debe ser capaz de funcionar en tiempo real, escuchando la canción mientras se reproduce. Por supuesto, se debe incurrir en los menores costes dentro de lo posible. Se ha optado por una implementación Plug & Play, donde el prototipo pueda conectarse al ordenador y pueda ser usado casi instantáneamente.

Ante estas ideas, la implementación sobre un microcontrolador que se encargue de la comunicación entre ordenador y periféricos será la óptima. Los microcontroladores funcionan con un consumo mínimo y la capacidad de cómputo es suficiente. Además, el algoritmo que se debe utilizar es IBT dada su funcionalidad en tiempo real y su buen desarrollo general en las pruebas contra sus competidores.

El algoritmo IBT se ejecutara en el ordenador correspondiente en tiempo real, el cual hace uso del micrófono del ordenador obteniendo la información acústica exterior. El algoritmo se comunicara vía USB con el microcontrolador mandando una señal en los instante donde IBT estime un pulso, haciendo que este realice las operaciones necesarias, entre las que se encuentran el procesamiento de dicha información y la comunicación con el modulo LED, que servirá para "*hacer visibles los* pulsos" para el usuario.

Respecto al algoritmo, este se apoya en Marsyas, un framework de código abierto escrito en C++ dirigido al procesado de audio que provee de las distintas funcionalidades (MarSystems) que aparecen en gran parte de los algoritmos de procesado digital de señal actuales, permitiendo así un rápido prototipado. Un breve recuento de Marsystems así como el uso de estos realizado en IBT se muestra en la figura 22.

Por supuesto, el algoritmo no se ha desarrollado para la función que se requiere aquí, por lo que se necesita de su modificación. Lo que se ha realizado es, por un lado, un vaciado de funcionalidades no útiles para su uso en tiempo real, minimizando así su tamaño y, por otro, una adaptación para conseguir que sea capaz de comunicarse mediante USB. Esta comunicación se realiza mediante la librería propia de Windows, incluyendo la cabecera *windows.h,* donde coexisten todas las declaraciones de funciones de la biblioteca Windows API, entre las que se encuentra la plataforma COM (de sus siglas en inglés, *Component Object Model*), con la que se permite acceder a un puerto serie USB del dispositivo.

5.2. Diseño del hardware

Para el funcionamoento del prototipo se necesita una serie de dispositivos de hardware que permitan desarrollar las funcionalidades explicadas en el anterior apartado.

5.2.1. Placa integrada ESP32

Todo proyecto electrónico digital de un cierto nivel que necesite de órdenes programadas requiere de un microcontrolador, un circuito integrado capaz de desarrollar las instrucciones que contiene en su interior de forma continuada con un muy bajo consumo de energía. En este caso, se hace uso del ESP32-WROOM 32D del fabricante Espressif, un microcontrolador muy barato con funcionalidades muy amplias, desde control GPIO (de sus siglas en inglés, Entrada-Salida de Proposito General) hasta Wi-Fi o Bluetooth.

Para facilitar tanto la escritura en memoria flash (flashing code) como su comunicación entre el USB y la UART (de sus siglas en inglés, Transmisor-Receptor Asíncrono Universal), el EPS32 debe ser acompañado por un módulo puente USB-UART, que permite realizar las anteriores acciones. El fabricante proporciona una placa de desarrollo que alberga el microcontrolador, el puente USB-UART y diferentes elementos de acondicionamiento y control de tensión con la que se puede hacer un uso más rápido del microcontrolador y a un coste más reducido. Esta placa es la denominada ESP-DevKitV4.

La documentación sobre este producto está expuesta en [6].

Figura 17 Microcontrolador EPS32 WROOM 32D

5.2.2. Matriz LED 8x8

Es necesario un dispositivo de hardware que sea capaz de mostrar dichos pulsos musicales. Un periférico fácilmente visible que permita al usuario visualizar la información del ritmo cuando conecta el aparato.

En primera instancia se podría pensar en utilizar un dispositivo acústico, imitando al metrónomo con su característico tic-tac, pero hay que tener en cuenta que el algoritmo no debe recibir ruido externo que pueda conllevar a errores.

Así, la mejor solución y la más visual es la de utilizar un *display* en forma de matriz formada por diodos LED. Tienen una disposición 8x8 por ser un tamaño compacto pero fácil de ver.

Esta matriz LED es de ánodo común, como se observa en la figura 18 y cada uno de los LEDs de la matriz se activa poniendo su fila a '1' lógico y su columna a '0' lógico.
Se requiere de resistencias limitadoras de corriente, en el caso de que se use de forma cruda pero no es lo que va a ocurrir en este prototipo sino que se va a utilizar un integrado de control, que se introduce en el siguiente apartado.

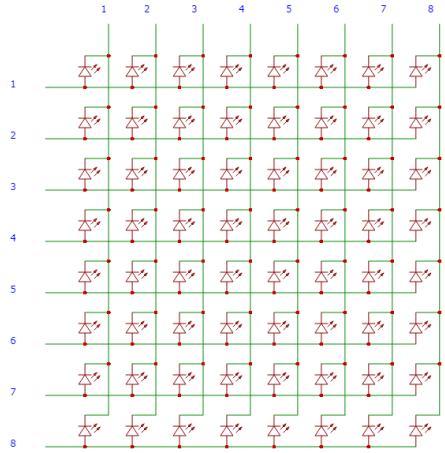

Figura 18 Matriz LED de dimensiones 8x8, 64 LEDS de ánodo común de color rojo.

5.2.3. Controlador MAX7219

El controlador MAX7219 es un circuito integrado utilizado en el control de displays de 8 dígitos o matrices de 64 LEDs individuales como la que se usa en este proyecto.

Los LEDs se activan mediante la creación de un pulso PWM con el que se puede ajustar la intensidad. Además, cuenta con una función de gasto mínimo (shutdown mode) y la capacidad de retener los datos hasta la próxima recepción de datos nuevos, mediante una SRAM de 8x8 bits.

Este dispositivo reduce el número de pines a utilizar para controlar la matriz, de 16 a únicamente 3, un pin de datos, DIN, donde se mandan una serie de 16 bits que contendrán toda

la información para activar los LEDs necesarios (Tabla 8), un pin de control CS, que se fija a '0' cuando se requiera leer la información del pin DIN y un último pin de reloj, CLK, que permite añadir los datos enviados al registro cada flanco ascendente. La comunicación con el microcontrolador se realizara mediante SPI (de sus siglas en inglés, Interfaz Serie de Periféricos).

El circuito debe alimentarse a 5v y ser conectado a masa adecuadamente siendo capaz de proporcionar una corriente nominal de 330mA.

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
X	X	X	X	ADDRESS				MSB			DATA				LSB

Tabla 8 Formato de los datos serie que se envían hacia el pin DIN y que el integrado computa. En los 4 bits de ADDRESS se escribirá o bien el número de la fila a activar o la orden a ejecutar por el microchip y en DATA, se escribirá los LEDs de dicha fila a activar.

5.2.4. Switch Configurable

Como implementación adicional, se conectara un selector de modos en formato *switch* mecánico que permitirá seleccionar como se quiere visualizar la matriz.
Existen 2 modos, el **modo usuario** que ilumina los 64 LEDs durante algunas centenas de milisegundos, permitiendo visualizar "a todo color" el pulso musical y el **modo animado**, que iluminara la pantalla con una serie de dibujos aleatorios seleccionados y creados por el autor.

En el Anexo 2 se muestran y explican características relevantes sobre la construcción del circuito y sobre elementos incluidos en el.

5.3. Diseño del algoritmo central para ESP32

Para el buen funcionamiento del prototipo es necesario que el microcontrolador sea capaz de ejecutar una serie de órdenes que permitan comunicarse con los periféricos y recibir información de IBT. Es necesario redactar esta serie de órdenes en formato de código C, un lenguaje de alto nivel con control de variable a nivel de bits y registros, que permite una mayor velocidad de procesamiento y una mejor capacidad de debug.

Espresiff es una empresa fabricante y distribuidora de hardware que produce sus propios sistemas operativos y frameworks y el ESP32 que se utiliza en este proyecto lleva instalado su propio firmware que permite al usuario realizar operaciones de bajo nivel mediante funciones de alto nivel, con las que incrementar la velocidad de prototipado, la legibilidad del código y, por tanto, el debug posterior.

Más información sobre las capacidades de este firmware se muestran en [15] donde se explican todas las posibilidades que tiene el EPS32 y sus variantes en cuanto a los periféricos que tiene integrados.

En el caso que se aborda se tratan 4 de ellas, la comunicación con la UART donde se hará uso del chip puente USB-UART, la comunicación serie SPI, que como se ha explicado, será utilizada para el envió de datos hacia el integrado MAX7219, los pines de propósito general E/S,

para permitir leer datos binarios y los contadores integrados, para realizar contajes de tiempo muy precisos.

5.3.1. FreeRTOS

FreeRTOS es un sistema operativo desarrollado para microcontroladores que permite exprimir todas las funcionalidades de tiempo real que el proyecto necesita. Este se distribuye libremente bajo la licencia de código abierto del MIT.

Los microcontroladores por lo general cuentan únicamente con 1 núcleo de procesado que les permite efectuar solo una operación a la vez. La optimización de los procesos que se ejecutan en este núcleo es completamente obligatoria en muchos casos, premiando la eficacia y el pulido de nuestro algoritmo. FreeRTOS permite hacer una secuenciación óptima de tareas que mandamos al procesador, obteniendo así la mejor secuencia y que permite que dichas tareas no se queden a mitad de ejecución.

En el algoritmo se han definido 2 tareas con 2 niveles de prioridad[1] que se ejecutaran cíclicamente

- UART_event_Task: Donde se trata todo el proceso de interpretación de datos enviados a la UART. Además, se tratan diferentes errores que pueden suceder en este dispositivo. Es la tarea más prioritaria ya que, en caso de sobrecarga de tiempos, sería la tarea que se ejecutase, no permitiendo que se pierda ningún pulso musical.

- Beat_Light: Donde se ejecuta el proceso de dibujado en la matriz LED tratando tanto el modo usuario como el modo animado mediante lectura del *switch*. Tiene una prioridad menor la perdida de una ejecución de esta tarea no supone una gran pérdida, es decir, si se da el caso en el que la tarea no se puede ejecutar por falta de tiempo en un periodo de ejecución pero si al siguiente o incluso 10 periodos después, en una muy mala situación, el retraso que supondría este percance a la visualización en pantalla sería despreciable.

5.3.2. UART

La UART es un dispositivo integrado en el microcontrolador que maneja los puertos y dispositivos serie. Para el caso que acontece, efectúa la función de control de interrupciones de los dispositivos conectados al puerto serie (TX y RX), lo cuales están interconectados al puente USB-UART que permite una interpretación de los datos mandados por el algoritmo IBT para que la UART sea capaz de utilizarlos de forma consistente.

[1] La prioridad es usada por los sistemas de tiempo real para ordenar y secuenciar tareas. Por ejemplo, una tarea con un plazo de ejecución muy corto debería ser más prioritaria que otra debido a la posibilidad mayor que tendría de no cumplir dicho plazo.

UART ira poco a poco recibiendo los datos mandados por el algoritmo IBT que serán almacenados en un registro interno esperando a ser leídos, mediante una disposición FIFO (de sus siglas en inglés, Primero en Llegar, Primero en Salir), donde los primeros en llegar son los primeros que se leen mediante la tarea definida en el apartado anterior. En caso de que no haya ningún dato, la tarea será procesada en un menor tiempo para maximizar así la eficiencia.

El dato a recibir será el byte correspondiente a un 1 en ascii (00110001). El envió de un byte completo, en vez de un solo bit, permite disminuir los errores de envió por causas externas como el ruido, habilitando únicamente el bloque de transferencia SPI en el caso de que el tren de bits recibido sea dicho '1'.

Por supuesto, otros tratamientos de errores se han propuesto y los más relevantes se muestran a continuación:

- Ring Buffer Full: detecta que el buffer de almacenamiento cíclico se ha llenado y se va a comenzar a sobrescribir los datos más antiguos.
- Parity Error: detecta un error en el bit de paridad de los datos.
- Frame Error: detecta un error en la trama de bits.
- Break Error: detecta una rotura en él envió de la trama.

En el primer caso, se borra el buffer y la cola se resetea, en los demás casos, únicamente se muestra el error. En todos los casos, una vez tratado el error, se continúa con la ejecución del programa (Anexo 3).

La trama de bits que se envía desde el ordenador anfitrión al microcontrolador debe tener la misma forma que la que se debe recibir para que entre ambos exista una comunicación. En este caso la trama está conformada por un bit de START, 8 bits de información o DATA y un bit de STOP. La comunicación entre ambos se realiza a 115200 baudios. Se usara el canal 0 de UART, existiendo un total de 2 canales en el ESP32.

5.3.3. SPI

Como se ha explicado en anteriores apartados, el SPI es un método de comunicación serie que permite enviar información a distintos elementos externos al microcontrolador. Es este caso en el que solo se pretende enviar información, se hará uso del microcontrolador como MOSI (de sus siglas en inglés, *Master Out, Slave In*) a una velocidad de reloj de 10MHz, propuesta por la documentación.

Como se veía en el apartado 5.2.3 del controlador MAXIM, los datos que se deben enviar deben tener la dirección para su colocación en la matriz (4 bits) y los datos a encender en dicha dirección (8 bits). La acción se realizara mediante una función creada llamada *spi_transfer*. Cada bit será enviado de forma síncrona con cada flanco ascendente del reloj.

Antes de realizar ninguna función, se hace una inicialización del integrado MAXIM, para comprobar que todo funciona correctamente. Una vez realizado esto, se procede a dibujar en la matriz lo que corresponda dependiendo del modo en el que se encuentre el programa

(usuario o animado). Se mandarán los bits necesarios para dibujar la matriz cada vez que la UART detecte un pulso mandado por IBT y, a continuación, se borrará la matriz una vez que haya pasado un cierto tiempo que permita su visualización correcta (200 ms). Este tiempo no se realiza mediante una espera como tal, ya que esto supondría un alargamiento del tiempo que dura la ejecución de la tarea, imposibilitando la acción del sistema FreeRTOS y eliminando la posibilidad de que se lean los datos que recibe la UART en el momento correcto. La lectura de dicho tiempo se explica en el apartado 5.3.4.2.

Aun así, es completamente necesario hacer una espera de alrededor de 10 ms después de cada envió de datos SPI por necesidad computacional del integrado MAXIM. Aun así, al ser un tiempo tan pequeño, no conlleva al problema comentado en el párrafo anterior.

5.3.4. Funcionalidades secundarias

Además de las funcionalidades relevantes que se han ido explicando existen otras que aunque no sean realmente complejas, son completamente necesarias para la ejecución del prototipo.

5.3.4.1. GPIO

La función básica y principal de todo microcontrolador es la de las GPIO. Este bloque integrado en el microcontrolador permite efectuar operaciones de entrada y salida en base a las órdenes escritas. Al final y al cabo, todas las funciones del microcontrolador están basadas en la operativa de dicho modulo que permite situar la tensión de sus salidas en 2 estados binarios, '1' o '0'. Estos valores no son tensiones exactas a VDD y 0v, sino que oscilan entre VDD y 0.8*VDD, para '1' y 0.1*VDD y 0v para '0'.

En este caso, se hará uso de la GPIO 34 como entrada, con la que se podrá leer el estado de *switch*. Es muy importante añadir una resistencia a modo *pull-up* para que exista un punto de medida que el microcontrolador pueda medir de forma eficaz, para ello, se hará uso de la integrada en el propio dispositivo ($45\ k\Omega$). En el caso de no incluirla, la lectura del '1' lógico sería únicamente ruido electromagnético que daría como resultado una lectura de '0'.

5.3.4.2. Timer

En muchos proyectos es muy importante el contaje del tiempo para medir el cómputo de una acción o definir el inicio de una tarea a futuro, por ejemplo. En este caso, se requiere de realizar un conteo de tiempo en el que la matriz LED esta activa, así como realizar esperas de tiempo, para permitir que el controlador MAXIM funcione correctamente.

Los microcontroladores suelen tener la capacidad de contar pulsos de reloj almacenando el valor acumulado en un registro. El ESP32 no es distinto y cuenta con 2 grupos de bloques *Timer* con 2 contadores cada uno, con un registro máximo de 64 bits.

Por supuesto, dicho registro es de un tamaño limitado y la velocidad del reloj es muy alta. Para limitar que el registro se llene muy rápido y que el microcontrolador este continuamente actualizando dicho registro se impondrá un valor de preescalado de 32, que corresponde al máximo valor que minimiza las operaciones de escritura en registro pero que a su vez, evita perdidas de precisión.

6. Conclusiones y futuras mejoras

6.1 Conclusiones

La labor que realiza el Beat-Tracking en la sociedad actual no es más que una simple herramienta o un útil divertimento utilizado en conciertos o salas de música. El desarrollo de las técnicas ha evolucionado hasta nuestros días con una mejora sustancial gracias a diversos autores y expertos en la materia.

Aun así, el desempeño de estas técnicas dista mucho aun de una buena operatividad para cualquier tipo de canción y dista mucho más aun de la perfección, dada la complicación que implicaría crear bases de datos con una verdad absoluta.

Es muy probable que los algoritmos basados en procesado digital hayan tocado techo en sus posibilidades y sea necesario un cambio de disciplina. El futuro probablemente esté en el uso de redes neuronales.

En los distintos apartados de la comparativa se ha mostrado el deficiente desempeño de los 3 algoritmos frente a una base de datos tan compleja como el que se utiliza [3], pero para que el lector tenga otra referencia de resultados, en [1] se realiza una comparativa parecida con una base de datos de menor dificultad donde se aprecia una considerable mejora.

En este documento se aportan, además de una nueva comparativa de la técnica, una aplicación web donde verificar la capacidad rítmica y la descripción de un prototipo completamente original y funcional que pueda ser utilizado en clases de música o en personas que carezcan de cualidades rítmicas.

Aunque este fuese el motivo principal del desarrollo, los datos que arrojan los resultados son que el uso del dispositivo no proporcionaría una ayuda real debido al mal desempeño de los algoritmos y que su uso podría estar mayormente orientado a un control LED meramente recreativo. En una ampliación podría orientarse hacia el uso de nuevas técnicas con redes neuronales intentado mejorar la precisión.

Las dificultades que se han encontrado han sido muy variadas pero radicaban principalmente en el no conocimiento de los temas y herramientas que se usaban y que, por tanto, han supuesto la fuente de aprendizaje mayor, tanto a nivel teórico como de cara a ser una persona con la capacidad suficiente a encontrar la información para resolver un problema, y que realmente es una de las características más relevantes a la hora de ser un profesional en cualquier campo de la ingeniería.

6.2 Mejoras futuras

En este documento se han expuesto una serie de técnicas Beat-Tracking y se ha propuesto un prototipo funcional para suplir una necesidad pero dado el carácter de un Trabajo Final de Grado, con su limitación de tiempo y recursos, ha habido ciertos aspectos que no se han tratado y que se exponen a continuación.

- En este trabajo se ha buscado únicamente la comparación de los algoritmo tal y como se encuentran en sus repositorios. En un futuro, sería interesante modificar ciertos parámetros internos para ver si se puede optimizar su funcionamiento.

- En los últimos años ha aparecido una corriente muy amplia con la implementación de redes neuronales adaptadas a procesos como los que aquí se describen. En una posible ampliación de este documento se pretende hacer hincapié en dichas técnicas por ser el futuro tanto en este campo como en muchos otros.

- Las modificaciones realizadas en el algoritmo IBT para su correcto funcionamiento sobre el prototipo se realizaron únicamente en Windows, produciendo así que solo funcione en este sistema operativo. En un futuro se pretende expandir su uso a otros sistemas operativos como Mac OS o Linux.

- Adaptar la aplicación web de forma que sea capaz de guardar los resultados sin tener que ejecutarse de forma local, es decir, adaptarla a internet.

7. Bibliografía

[1] M.F.McKinney, D.Moelants, M.E.P.Davies and A.Klapuri. Evaluation of Audio Beat Tracking and Music Tempo Extraction Algorithms. In *Journal of New Music Research,* 2007, Vol. 36. No. 1, pages 1-16, The Netherlands, Belgium, London, Finland.

[2]Sebastian Böck and Markus Scheld. Enhanced Beat Tracking with Context-Aware Neural Networks. In *Proc. Of the 14th Int.Conference on Digital Audio Effects, 2011,* Johannes Kepler University, Linz, Austria

[3] Holzapfel, A.; Davies, M.E.P.; Zapata, J.R.; Oliveira, J.L.; Gouyon, F.; , "Selective Sampling for Beat Tracking Evaluation," Audio, Speech, and Language Processing, IEEE Transactions on , vol.20, no.9, pp.2539-2548, Nov. 2012 doi: 10.1109/TASL.2012.2205244
In: URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6220849&isnumber=6268383

[4] In: URL: *https://en.wikipedia.org/wiki/Opus_(Eric_Prydz_song)*, 30 December 2020

[5] Handel, S. (1989). Listening: An Introduction to the Perception of Auditory Events. Bradford, MIT Press.

[6] ESP-32 documentation. Espressif Programming Guide.
In: URL: https://docs.espressif.com/projects/esp-idf/

[7] In: URL: *https://www.music-ir.org/mirex/wiki/2006:Audio_Beat_Tracking_Results Dic.2020.*

[8] S. Dixon. Evaluation of the Audio Beat Tracking System BeatRoot. In *Preprint for Journal of New Music Research,* 36, 2007/8, London, UK.

[9] S.Dixon. Onset detection revisited. In *in Proceedings of the 9th International Conference on Digital Audio Effects,* pages 133-13, Montreal, Canada, 2006.

[10] Marsyas, Music analysis, retrieval and synthesis for audio signal. Software Framework for IBT algorithm. In: URL: *http://marsyas.info/index.html*

[11] João Lobato Oliveira, Fabien Gouyon, Luis Gustavo Martins and Luis Paulo Reis. IBT: A Real-Time Tempo and Beat Tracking System. In *11th International Society for Music Information Retrieval Conference (ISMIR 2010).*

[12] João Lobato Oliveira, Fabien Gouyon, Matthew E. P. Davies and Luis Paulo Reis. MIREX 2012 Audio Beat Tracking Sumission: IBT.

[13] J. L. Oliveira, M. E. P. Davies, F. Gouyon and L. P. Reis, "Beat Tracking for Multiple Applications: A Multi-Agent System Architecture With State Recovery," in *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 10, pp. 2696-2706, Dec. 2012, doi: 10.1109/TASL.2012.2210878.

[14] D. P. W. Ellis and G. E. Poliner, "Identifying `Cover Songs' with Chroma Features and Dynamic Programming Beat Tracking," 2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07, 2007, pp. IV-1429-IV-1432, doi: 10.1109/ICASSP.2007.367348.

[15] D. Ellis, “Beat Tracking by Dinamic Programming” J.New Music Research, vol 36 no. 1 , pp. 51-60, March 2007

8. Anexos

Anexo I

- **Algoritmo BeatRoot**

Fue diseñado por Simon Dixon en 2006 en el lenguaje de programación Java. Cuenta con una interfaz gráfica con la que obtener y visualizar resultados, como se muestra en la Figura 19. El algoritmo dio muy buenos resultados en el momento de su salida en el concurso MIREX [7] y es por ello, el precursor de muchos otros.

Dixon estudió la posibilidad de utilizar entradas MIDI [2] y Match [3] como input del algoritmo de forma offline pero no es el uso típico que se realiza.

BeatRoot se basa en la técnica **Flujo Espectral** (SF) (4), que define un valor compuesto por la suma de la variación de la magnitud de la señal en cada intervalo de frecuencia, en la que se ha dividido el espectro frecuencial. Previamente, se realiza una Transformada de Fourier Localizada (de sus siglas en inglés, STFT (5)) con un enventanado **Hamming**, cada 10ms seguida de un pretratamiento de filtrado y, posteriormente, se aplica un algoritmo de máximos locales con umbrales definidos por el autor, obtenidos empíricamente [8], que obtendrán los eventos de inicio los cuales son la base del tratamiento posterior.

$$SF(n) = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} H(|X(n,k)| - |X(n-1,k)|) \qquad (4)$$

Donde H es una función de rectificación de media onda para tratar únicamente aumentos positivos y $X(n,k)$ representa la k-esimo intervalo de frecuencia de la ventana n.

$$X(n,k) = \sum_{m=-\frac{N}{2}}^{\frac{N}{2}-1} x(hn+m)w(m)e^{-\frac{2j\pi mk}{N}} \qquad (5)$$

Una descripción más extensa de esta técnica se encuentra reflejada en el documento correspondiente del autor [8], que además, incluiría mejoras de la misma, proponiendo ponderaciones en base a la magnitud o incluso arreglando problemas [9].

El algoritmo obtiene hipótesis de tempo mediante la puntuación de distintos clústeres que se crean de 25ms (entre 50ms a 2s), donde se albergan todos los posibles resultados de

[2] Musical Instrument Digital Interface es un protocolo de comunicación entre instrumentos digitales y/o ordenadores.
[3] Es una combinación entre la representación de las notas MIDI y su asociación en la partitura

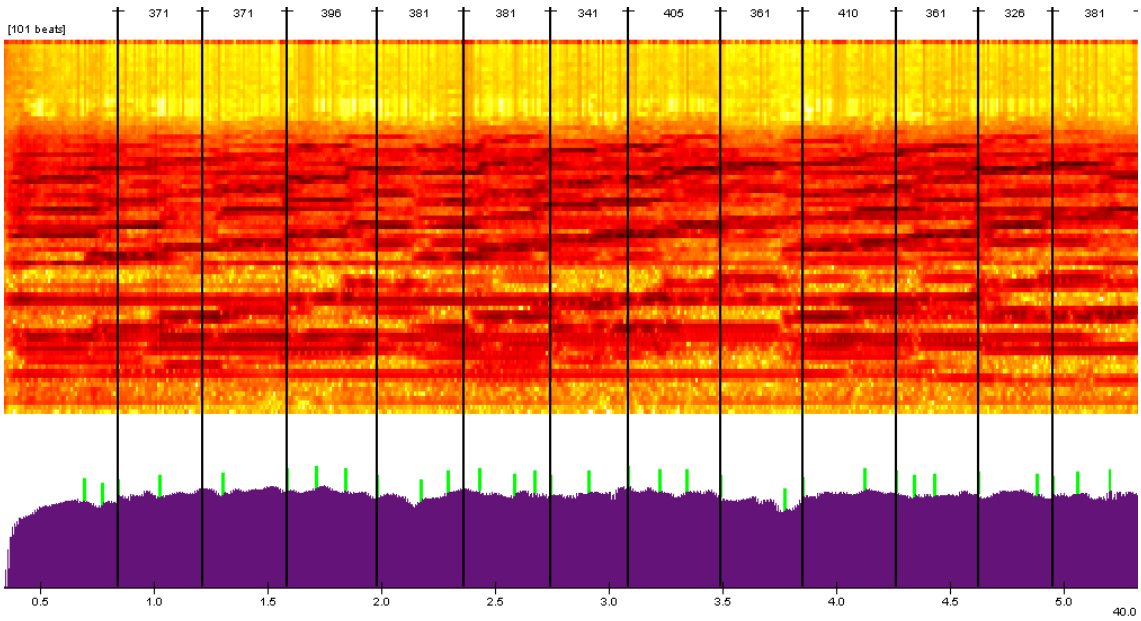

*Figura 19 Interfaz gráfica de BeatRoot, donde se muestran un extracto de 5 segundos de "*C'est La Moindre Des Choses*". Se observa el espectrograma en colores rojizos, la envolvente de amplitud, en morado, los eventos de comienzo detectados, marcados en verde y los pulsos seleccionados en negro.*

intervalo entre eventos (de las siglas en inglés, IOI) (Figura 20). Las puntuaciones y cómo se tratan los resultados está extensamente explicado en el siguiente documento [8].

Para finalizar, una vez que se obtiene la hipótesis de tempo se necesita la hipótesis de fase asociada que permita componer el pulso musical. Dixon propuso una implementación con pool de agentes en la que cada agente obtiene una hipótesis de tempo y una fase, que ira autopuntuandose según ciertos márgenes de tolerancia interiores y exteriores impuestos.

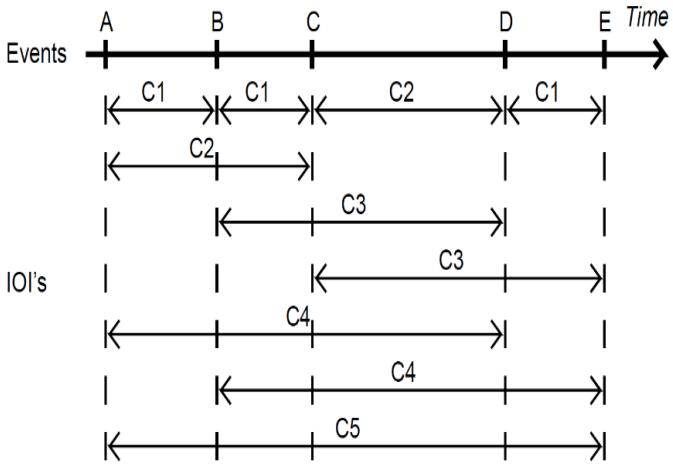

Figura 20 Agrupamiento de todos los IOI's posibles en los 5 clústeres existentes. Extraída de [8].

Estas puntuaciones se regulan en base a los eventos de comienzos detectados. Si estos entran dentro de los umbrales de selección, los agentes obtendrán puntuaciones o incluso llegarán a modificar sus tempos iniciales en base al error medido entre el pulso estimado y el evento de comienzo aparecido.

Dixon define en su algoritmo exclusiones, duplicados, eliminaciones y todo tipo de tratamiento de estos agentes para evitar la sobrecarga y mejorar el funcionamiento del sistema [8] [9].

- **Algoritmo INESC Porto Beat Tracker o IBT**

Este algoritmo fue diseñado por cuatro investigadores de diferentes instituciones de ciencias e ingeniería, en Portugal. Está escrito en lenguaje C++ e integrado dentro del framework MARSYAS [10], que alberga múltiples funcionalidades derivadas del procesamiento de señal. Basado en la metodología que seguía BeatRoot y modificado para mejorar tanto el tiempo de cómputo como la robustez frente al ruido [1] permite, como novedad, una actuación en tiempo real, ciertamente útil en aplicaciones prácticas.

La técnica en la que se basa es, de nuevo, el flujo espectral, como su predecesor BeatRoot, pero con ciertas mejoras como un trabajo de filtrado posterior de fase 0, que reduce falsos positivos. Los parámetros de audio y procesamiento se encuentran explicados en [11].

A partir de los eventos de comienzo recogidos de la etapa anterior, se calcula la autocorrelación de la función de flujo espectral, a lo largo del tiempo definido para la ventana de inducción. En esta, el algoritmo hace el tratamiento previo donde se buscan hipótesis iniciales de tempo y pulso.

Cada hipótesis de periodo se define como la diferencia entre los máximos de dicha autocorrelación (6) que serán obtenidos mediante un algoritmo de detección de máximos global (7), en base a un umbral definido por el autor.

$$A(\tau) = \sum_{n=0}^{m} SF(n)SF(n+\tau) \qquad (6)$$

Donde $SF(n)$ es la función de flujo espectral, m es la longitud de la ventana de inducción, medida en frames[4].

$$P_i = \arg\max_i\big(A(\tau)\big), i = 1, \dots, N \qquad (7)$$

Donde P_i es cada una de las hipótesis de fase que se obtendrán como el desfase temporal de cada uno de los máximos globales.

El tratamiento que usa a posteriori se basa en la misma idea de pool de agentes que BeatRoot a los que se les proporciona una hipótesis inicial de tempo y fase. La fase asociada a cada tempo se obtiene mediante la creación de trenes de pulsos con diferentes fases. La fase que junto con el tempo asociado mejor coincida con los eventos de comienzo obtenidos del flujo espectral, será la que se adjudique a esa hipótesis de tempo (Figura 21).

Además, a los agentes se les proporciona una puntuación ponderada entre la suya, mayor cuanto más similares resultados se hayan obtenido en la fase anterior, y entre la de sus compañeros [12].

[4] Se refiere al número de enventanados en las que se ha dividido el tiempo de inducción.

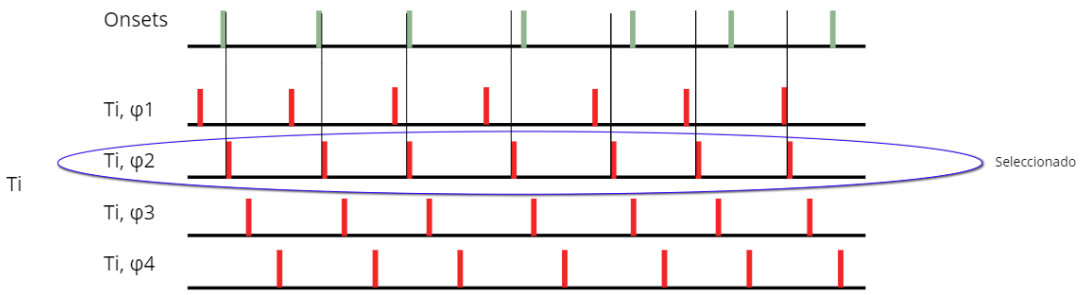

Figura 21 Representación de la selección de fase para una hipótesis de tempo concreta. Se elige la que obtenga una mayor correlación con los eventos de comienzo detectados en el flujo espectral.

Todo lo anterior se realiza en la ventana de inducción, un tiempo de alrededor de 5 segundos, donde se realizan las estimaciones iniciales. El algoritmo tiene incorporado un modo de reinducción donde se vuelven a estimar los resultados anteriores y que, en algunos casos, permite mejorarlos.

A partir de aquí, entrará en juego el control de agentes, que compara las hipótesis con la realidad en función de unos umbrales predefinidos y modifica las puntuaciones de cada agente basándose en esta comparación. También se encarga de todas las operaciones posibles de gestión de agentes, crear, eliminar…, en base a ciertos parámetros modificables ([12] y ampliado en [13]).

Todo lo anterior se implementa dentro de Marsyas con su sistema característico de bloques, de la forma en la que se muestra en la figura 22.

- **Algoritmo Ellis**

Diseñado por Daniel P.W.Ellis en la Universidad de Columbia, como parte de una investigación. Desarrollado en el entorno y lenguaje MatLab, alberga una metodología relativamente diferente a los anteriores algoritmos presentados.

El algoritmo comienza con la creación de una señal envolvente de la fuerza de los eventos de inicio, que representa los aumentos de la energía de la señal entre las bandas de frecuencias.

La señal inicial es mapeada en 40 bandas de Mel[5], mediante una suma ponderada de los valores del espectrograma [14]. Después, se realiza la diferencia de los valores del espectrograma entre cada muestra, recogiendo únicamente aquellos valores ascendentes, para posteriormente realizarse un filtrado paso alto y una convolución gaussiana para obtener media 0 y suavizar la señal, respectivamente. Construida la señal base, se pretende determinar el tempo. Este se obtendrá mediante una autocorrelacción de la envolvente descrita, enaltecida mediante una función de ponderación gaussiana, que permite tratar mejor la señal correlada.

[5] La escala Mel es una transformación logarítmica de la frecuencia de una señal en la que se busca que los sonidos que los humanos perciben a una misma distancia frecuencial se representen en la escala Mel con el mismo espaciado entre ellos.

El algoritmo es capaz de dar 2 soluciones de tempo. La primera será la estimada por el algoritmo y la segunda será una ponderación de la primera puntuación con dos grupos de métricas de tempo ((x2, x1/2) y (x3, x1/3)), siendo la elegida la que obtenga mejor puntuación.

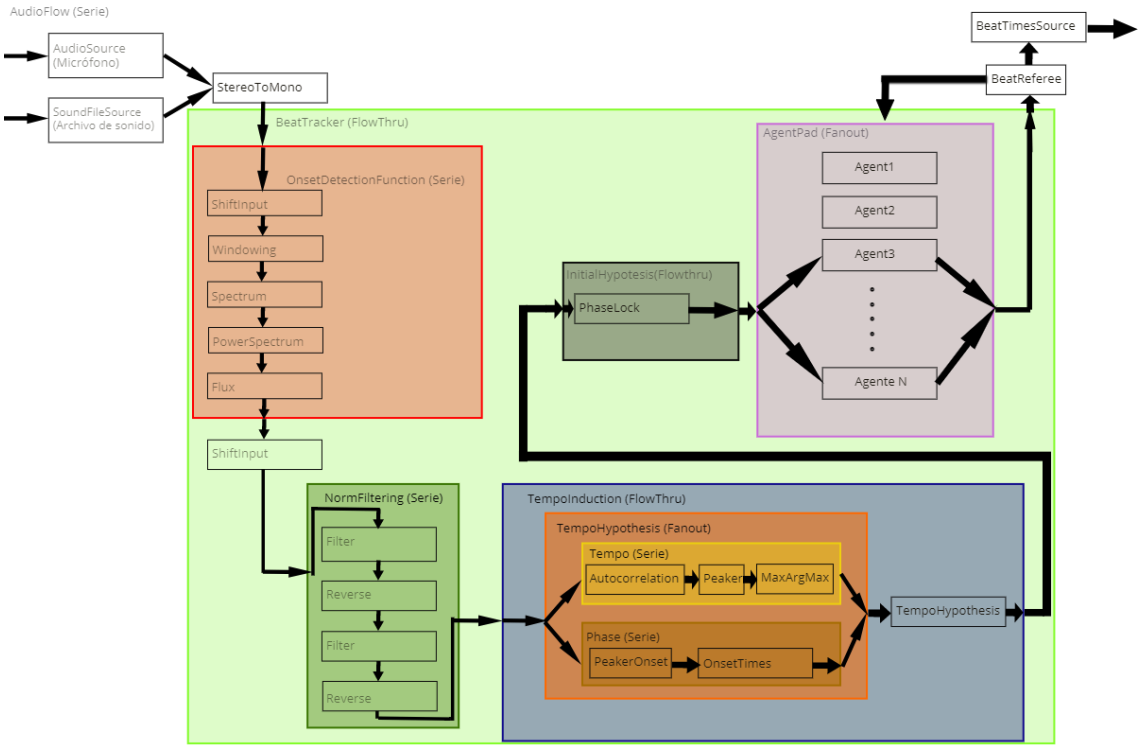

Figura 22 Descripción del sistema Marsyas de IBT, donde se muestra cada etapa del algoritmo con los nombres de cada función. Realmente es un esquema completo de cómo se ejecuta el algoritmo en tiempo real.

A continuación, se realizara una programación dinámica, descrita en [15], con la cual se optimiza el uso recursivo de la Ecuación 8, utilizada para valorizar las diferentes muestras de la señal y poder así, encontrar los pulsos.

$$C(\{t_i\}) = \sum_{i=1}^{N} O(t_i) + \alpha \sum_{i=2}^{N} F(t_i - t_{i-1}, \tau_p) \qquad (8)$$

Donde $O(t_i)$ es el valor de la envolvente en el instante t_i, α es un coeficiente de ponderación y F es una función que depende del tempo estimado anteriormente (9).

$$F(\Delta t, \tau) = -\left(\log \frac{\Delta t}{\tau}\right)^2 \qquad (9)$$

El desarrollo de la técnica está largamente descrito en [15], pero brevemente consiste en guardar los máximos valores que devuelve la ecuación 8 y realizar un “backtrace” a estos para definir los pulsos encontrados en base a parámetros.

En la tabla 1 se muestra un resumen de características de los 3 algoritmos comentados.

Anexo II

En la figura 23 se muestra el esquema general del circuito, que cuenta con partes bien diferenciadas que se van a explicar a continuación:

- **Microcontrolador**

  Se muestran todas las conexiones que existen entre el microcontrolador y los demás componentes. El ESP-32 cuenta con 39 pines con distintas funcionalidades, VDD, GND, GPIOs, PWMs…

  Los pines más relevantes conectados son el IO5 (CS), IO18(CLK), IO23(DIN) para la operación de comunicación SPI, IO34 (input) como lector digital de tensión y RXD y TXD, que permiten la comunicación con el microcontrolador y el puente USB-UART.

  La alimentación de este elemento se muestra también. Es necesaria aplicar una tensión de 3.3v en los pines VDD y EN.

- **Control Matriz LED**

  La matriz LED es un elemento único que agrupa 64 LEDs en serie-paralelo. Esta perfectamente conectada con el chip integrado MAXIM7219 que permite un control más sencillo de la misma.

  Tanto las columnas como las filas de la matriz tienen un pin establecido del controlador, al que se conectan. Por otro lado, los pines de control CS, DIN y CLK se conectan al microcontrolador como se ha mencionado en el punto anterior.

  Por supuesto, el integrado que será el responsable de aplicar la corriente necesaria a los LEDs deberá ser alimentado. El MAXIM7219 concretamente debe ser alimentado a 5v y la corriente que proporcionara, y con ello la intensidad lumínica que tendrán los LEDs estará directamente relacionada con la resistencia R11. El fabricante recomienda un valor de 9.53kΩ.

- **Puente USB-UART**

  De izquierda a derecha, de arriba a abajo se encuentra la conexión con el cable USB y su consiguiente adaptación y protección de tensión, el puente USB-UART, que permite la comunicación con el microcontrolador y una etapa de adaptación de las señales DTR y RTS (de sus siglas en inglés, Terminal de Datos Listo y Listo para Enviar).

  La alimentación de este dispositivo se realiza mediante el pin VDD, a 3.3v adaptados mediante la referencia de tensión. El pin SUSPEND negado debe conectarse a masa mientras que el pin RST negado debe conectarse a 3.3v, es decir, ‘1’ lógico. A su vez, el pin VBUS debe ser alimentado a una tensión aproximada de la mitad de la tensión de alimentación del USB, por requerimientos del fabricante.

Los pines DTR y RTS activan las funciones del microcontrolador que permiten grabar código o entrar en modo BootLoader.

- **Referencia de tensión**

  En multitud de circuitos no existen siempre los mismos requerimientos de tensión para alimentar cada elemento en ellos. Es por ello que se necesitan de varias tensiones o de una tensión muy estable de un valor muy concreto. Aquí es donde aparece la referencia de tensión que es un elemento que permite establecer una tensión fija con muy poco rizado y una gran estabilidad frente a los cambios de temperatura y/o alimentación.

  En este caso, gracias a la tensión que se obtiene del USB se fija otra tensión a 3.3v que permita alimentar ciertos elementos como el puente USB-UART, el microcontrolador…

  Es muy adecuada la utilización de condensadores que permitan eliminar ciertas variaciones de tensión que pudiesen darse, a modo de filtro paso bajo.

- **Botones útiles**

  Esta parte del circuito no es apta para el usuario y únicamente existe para facilitar la tarea del creador. El botón de FLASH permite grabar el código que se envía en la memoria del microcontrolador mientras que el botón BOOT permite entrar en el modo BootLoader. Estos botones están conectados a los pines EN y IO0 del microcontrolador.

  El puente USB-UART ya controla estos dos pines y por ello no es necesario pulsar los botones a la hora de grabar el código, pero por posibles futuros fallos en el futuro, se han incluido.

- **Switch**

  Como se ha explicado a lo largo del documento, dicho Switch permite la selección de la visualización de la matriz LED entre los dos modos existentes.

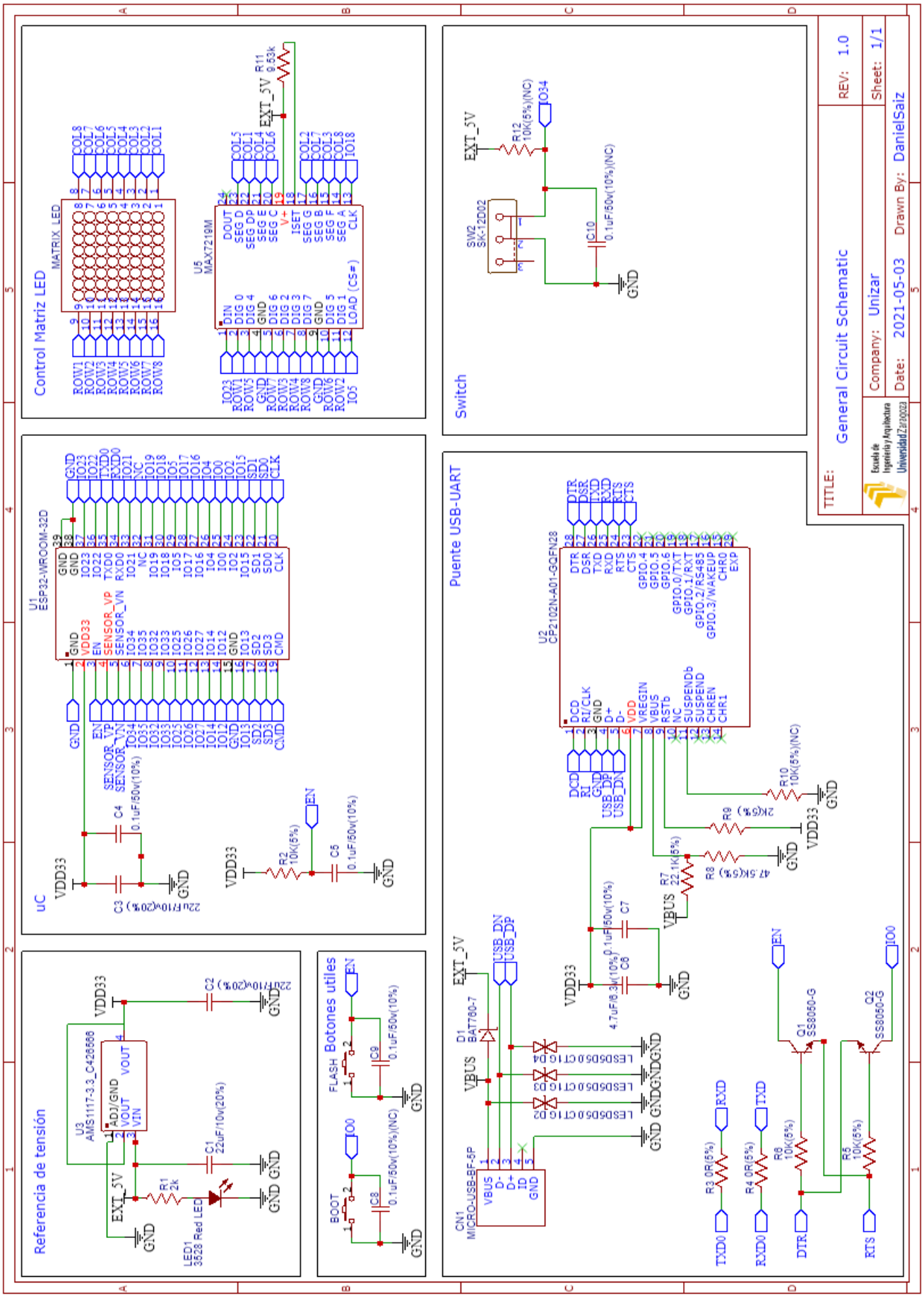

Figura 23 Esquema general del circuito del prototipo. Se muestran todas la etapas/partes descritas en el Anexo 2 de este documento. Se puede encontrar más información en el repositorio ***https://oshwlab.com/DanielSaiz/beattrackermachine***

Anexo III

A continuación, se muestran los diagramas de flujo del código que trata las interrupciones y errores del bloque UART y del código que controla la activación de la matriz LED.

El código completo del prototipo electrónico como el código fuente modificado y el programa ejecutable IBT se pueden encontrar en el repositorio de GitHub ***github.com/LeDanix/Beat_Tracking_IBT_Prototype***

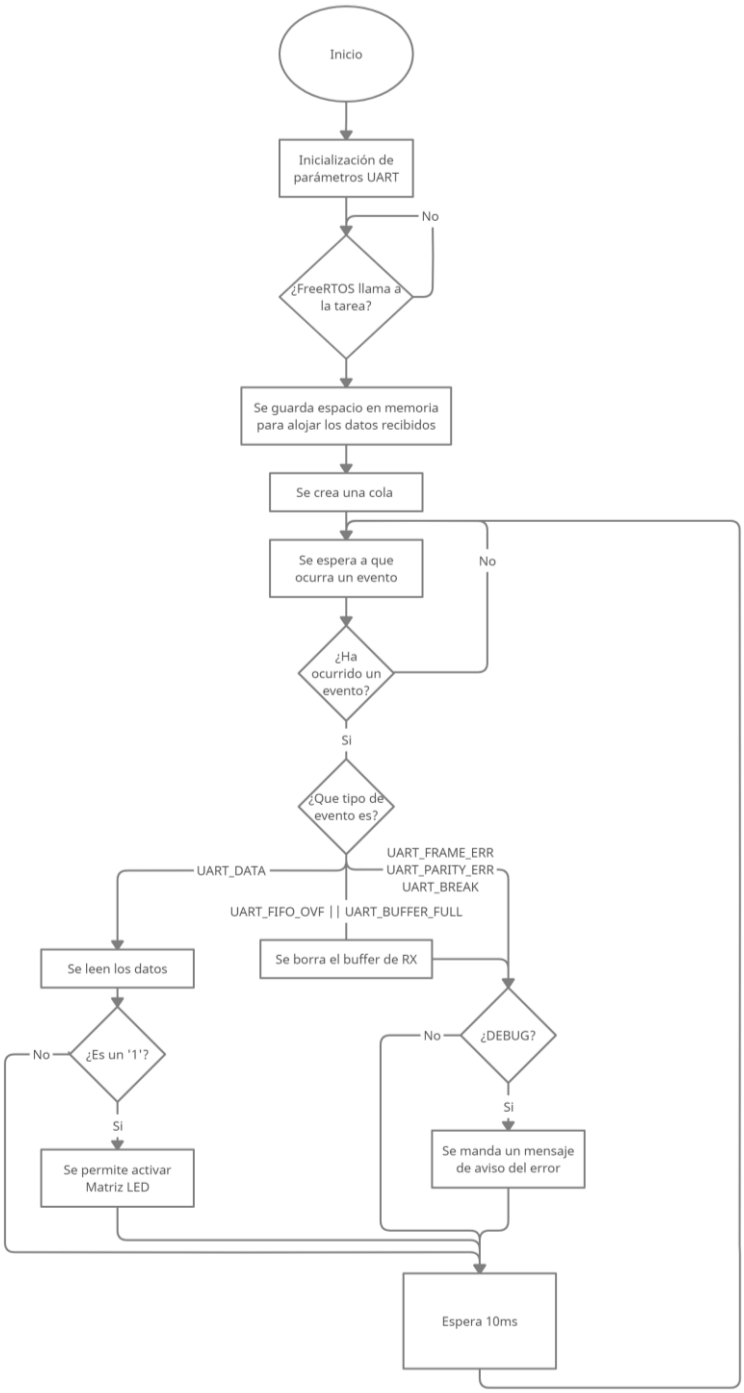

Figura 24 Diagrama de flujo de la tarea de control UART. Contiene información sobre cómo se tratan los datos recibidos y los errores posibles.

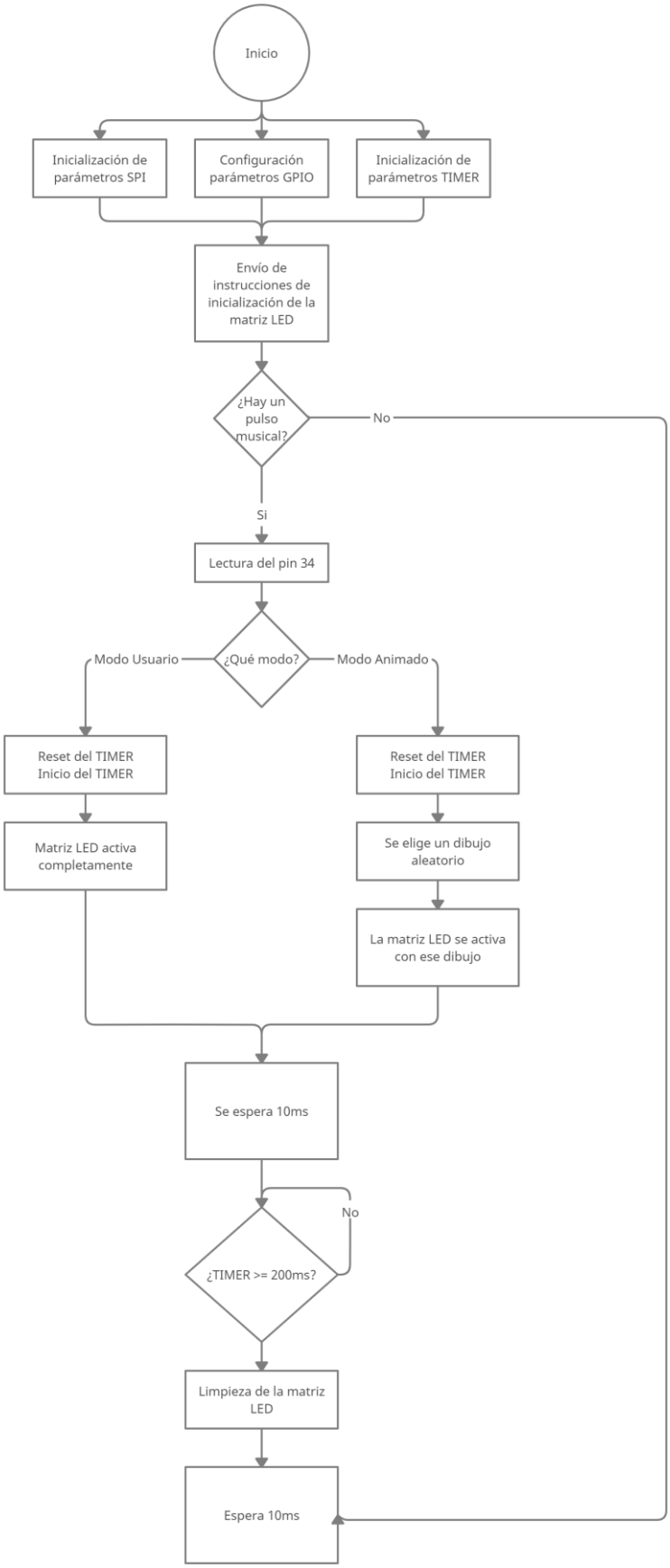

Figura 25 Diagrama de flujo de la tarea de control de la matriz LED.