FACULTAT
DE CIÈNCIES
I TECNOLOGIA
UVIC | UVIC·UCC

Treball de Fi de Grau

Lector de partitures de bateria amb maqueta simuladora de sons

Martí Solà Planagumà

Grau en Enginyeria Mecatrònica
Tutor/a: Ramón Jerez Mesa
Vic, Juny de 2019

M'agradaria dedicar aquest treball a totes aquelles persones que, d'alguna manera o altra, m'han ajudat i recolzat al llarg del projecte, i que de segur que sense elles no hagués estat possible:

A en Gerard Vallverdú, per tenir tanta paciència amb mi i imprimir-me tants prototips dels suports dels motors, així com les ajudes aportades en els aspectes mecànics.

A en Sergi Bernet, per a oferir-se també a imprimir-me peces quan corrien pressa.

A en Toni Vilatimó, per ajudar-me a solucionar els problemes que vaig tenir amb la connexió a Internet de la Raspberry.

Al Dr. Jan Jacobus Janse, de Matlab, per ajudar-me amb la compilació del programa, i per posar-me en contacte amb en Pablo.

A en Pablo Romero, de Matlab, per trobar temps per escriure'm la presentació que porta aquest treball.

A en Jordi Serra, per tota la paciència que ha tingut amb mi, i les múltiples ajudes a taller.

A en Ramón Jerez, per acceptar ser tutor d'aquest projecte, pel seguiment que m'ha portat, per totes les hores d'ajuda que m'ha ofert, i per exigir-me sempre l'excel·lència.

A la meva família, per tots aquests anys d'alegries i patiments que ara donen els seus fruits.

I per acabar, però no per això menys important, a aquella persona tant "*espacial*" que em recordava, dia rere dia, que jo podia amb el que fos.

A tots vosaltres, us dono les meves més sinceres gràcies.

"Never stop learning,

because life never stops teaching."

RESUM TREBALL DE FI DE GRAU

GRAU EN ENGINYERIA MECATRÒNICA

Títol : *Lector de partitures de bateria amb maqueta simuladora de sons*
Paraules clau : *Bateria*, *Maqueta*, *Processat d'Imatge*, *Protocol MQTT*
Autor : Martí Solà Planagumà
Tutor : Dr. Ramón Jerez Mesa (UVic - UCC)
Data : Juny de 2019

En aquest *Treball de Fi de Grau* s'exposen els resultats de la realització d'un programa capaç de llegir partitures de bateria (arxius en format imatge), i enviar les notes que cal tocar, en el moment precís, a una maqueta que simula els sons que faria l'instrument en qüestió. Enfocat a ser una eina que pugui servir per a practicar amb algun instrument, o per ensenyar a tocar-ne un, l'objectiu principal d'aquest projecte és fer una aplicació totalment funcional, senzilla, i entenedora per l'usuari estàndard, i que permeti l'esmentat anteriorment. L'aplicació s'ha desenvolupat en l'entorn de programació Matlab, i s'ha utilitzat un protocol MQTT per comunicar-la amb el microcontrolador encarregat de fer moure la maqueta (Raspberry Pi 3).

Com a resultats del projecte, s'han obtingut una aplicació lectora prou senzilla per l'usuari, tot i que millorable en els aspectes estètic i de disseny ; i una maqueta/prototip bastant més lenta de l'esperat, de dimensions 666 x 40 x 30 $[mm]$, a la qual no se li ha implementat l'opció de poder tocar notes que es realitzarien amb el peu esquerre, al no ser part de l'abast d'aquest projecte.

FINAL DEGREE PROJECT ABSTRACT

DEGREE IN MECHATRONICS ENGINEERING

Title : *Drums score reader with sound simulator scale model*
Keywords : *Drums*, *Scale Model*, *Image Processing*, *MQTT Protocol*
Author : Martí Solà Planagumà
Tutor : Dr. Ramón Jerez Mesa (UVic - UCC)
Date : June of 2019

In this *Final Degree Project* are exposed the results of making a program capable of reading drums scores (files in image format) and sending those notes that must be played, at the right moment, to a scale model that simulates the sounds that would be done with the actual instrument. Focused on being a tool that can be used to practice with some instrument, or to teach to play one, the main goal of this project is making an application fully functional, simple and understandable to the standard user, enabling the functionalities mentioned above. The application has been developed with the Matlab programming environment, and an MQTT protocol has been used to communicate it with the microcontroller in charge of moving the scale model (Raspberry Pi 3).

As a result of this project, a user-friendly application has been obtained, although it can be improved in the aesthetic and design aspects ; and a scale model/prototype quite slower than the expected, 666 x 40 x 30 $[mm]$ in dimension size, to which has not been implemented the option of being able to play notes that would be made with the left foot, due to the fact that is not part of this project scope.

PRÒLEG

Este proyecto "Lector de partituras de batería con maqueta simuladora de sonidos" representa un buen ejemplo de la complejidad y de la necesidad de un enfoque multidisciplinar de los retos y proyectos a los cuales se enfrenta la ingeniería y el desarrollo en el día de hoy.

Atrás quedan los tiempos en los que equipos de diseñadores y desarrolladores trabajaban de forma independiente, en base a unas especificaciones y requisitos, con escasa comunicación entre ellos aparte de la entrega o intercambio de una serie de elementos según dichos acuerdos, y sobre los cuales había apenas información hacia etapas anteriores del proceso de desarrollo, haciendo los ciclos de desarrollo lentos, farragosos y costosos, especialmente en el caso de detección de un error o necesidad de cambio detectados en etapas finales del proceso.

Hoy en día, los equipos y los propios ingenieros y desarrolladores que los forman, están más especializados que nunca, pero del mismo modo, la necesidad de ser cada vez más competitivos, eficientes y rápidos en los desarrollos, promueven la colaboración entre equipos multidisciplinares, las revisiones en equipo, el dominio de diferentes herramientas y métodos, y las iteraciones o procesos de mejora continua, dando alas a metodologías como Agile o el Diseño Basado en Modelos, que aplicándose de forma adecuada permiten aumentar la productividad, eficiencia y calidad en el desarrollo de servicios y productos para un mercado cada vez más exigente.

Aquí se puede apreciar a una escala menor pero representativa, la necesidad de abordar un problema de diseño e ingeniería desde diferentes ámbitos y con un enfoque multidisciplinar, como se verá, tratando temas técnicos como el procesamiento de imagen, la robótica, el Internet de las Cosas y diversos protocolos de comunicaciones; y buscando hacerlo útil y atractivo de cara al usuario final con una aplicación en el ámbito de la música, y destacando la importancia de la usabilidad y el diseño de interfaces y la experiencia de usuario.

Todo ello queda plasmado en un trabajo fin de grado, que refleja un proyecto de ingeniería actual y de interés, y llevado a cabo con éxito gracias, entre otros motivos, a la calidad de la formación universitaria española, muchas veces injustamente infravalorada respecto a otras de su entorno, la motivación y esfuerzo de los estudiantes y profesorado, y el uso de herramientas modernas como MATLAB y Simulink de uso extendido en la industria y los organismos de investigación, que permiten a los ingenieros y científicos abordar flujos completos de trabajo, como en este caso, desde la captura y procesamiento de datos e imágenes, hasta el control y prototipado físico, proporcionando un entorno de trabajo ágil, completo y de calidad.

Finalmente, me gustaría destacar la importancia de tender puentes entre la formación teórica académica, y la práctica y empresarial o industrial, que proyectos de este tipo permiten realizar, haciendo uso de herramientas y métodos de uso extendido y de aplicación práctica, para mejorar las habilidades y empleabilidad de los estudiantes de cara a un futuro profesional cada vez más exigente y competitivo.

Pablo Romero Cumbreras

Ingeniero de aplicación especialista en simulación
en tiempo real en MathWorks GmbH.

Nota: este texto refleja exclusivamente la opinión personal de su autor y no está vinculado a una declaración o postura oficial de MathWorks GmbH ni de sus empresas asociadas.

ÍNDEX

ÍNDEX de FIGURES

Capítol 1. - INTRODUCCIÓ

En aquest primer capítol s'introdueix el treball realitzat: es descriu la motivació que ha donat peu a que es porti a terme aquest projecte, es defineixen els objectius del mateix, es dóna una visió global del seu abast i quin lloc té actualment al mercat, i es fa un breu resum dels capítols que el precedeixen.

1.1. - Motivació

Els que toquem algun instrument sabem que quan volem aprendre una cançó nova, ajuda molt tenir-ne el ritme de fons (per aprendre's i marcar bé els temps). Això, a vegades, és una mica complicat d'obtenir ja que, a menys que coneguem a algú que toqui algun instrument de percussió i ens vulgui acompanyar a cada assaig, no es troben aquests tipus de melodies aïllades. Sempre podem aconseguir la cançó sencera per *Youtube*, però hi trobarem també superposats tots els altres instruments, incloent-hi el nostre amb el qual volem aprendre a tocar-la.

D'aquí neix aquest projecte; d'aquesta necessitat de tenir el ritme d'una cançó sense la superposició d'altres instruments, de manera que el músic pugui treballar i entrenar-se sobre aquesta percussió.

1.2. - Objectius

L'objectiu principal del projecte és crear una aplicació totalment funcional, senzilla, i entenedora per a l'usuari estàndard, i que estigui enfocada a ser una eina que pugui servir per a practicar amb algun instrument, o per ensenyar a tocar-ne un.

Com a objectius secundaris es planteja:

- Connectar de forma totalment inal·làmbrica l'aplicació dissenyada amb una maqueta que es construirà a posteriori, controlada per un microcontrolador.

- Construir un prototip de bateria musical amb materials reciclats, per tal de reduir costos, i que aquesta pugui simular els sons que li enviï la pròpia aplicació. Es vol basar en una distribució semblant a la de la **Figura 1**:
    - Dos baquetes (com a la vida real) mogudes per dos motors, que simularan el moviment dels canells. Estaran composades de bastonets xinesos.
    - Dos motors que simularan els moviments dels braços per tocar les diferents parts de la bateria.
    - Dos motors que simularan els moviments dels peus.
    - Un *bombo* (timbal del peu dret), una *caixa* (timbal redoblant), un *goliat* i dos *tom*'s (timbals amb sons més greus), un *hi-hat* o *xarles* (plats del peu esquerre), i un *crash* i un *ride* (plats més grans)[1]; estaran implementats amb llaunes d'olives de bar de diferents mides i les seves respectives tapes.

Figura 1.- Partesdel. (2019). *Parts d'una bateria* [Imatge]. Recuperat de *https://www.partesdel.com/partes_de_la_bateria.html*

Tot i això, el projecte se centrarà majoritàriament en la programació de l'app i no pas amb la maqueta, que serà útil sols per acabar de demostrar la funcionalitat de la primera.

[1] Més informació de les bateries musicals a: https://ca.wikipedia.org/wiki/Bateria_(instrument_musical)

1.3. - Estat de l'Art

Actualment no existeix cap aplicació igual al mercat que, a través d'una imatge de partitura de bateria, pugui arribar a fer sonar la cançó escrita. Tot i això, existeixen certes aplicacions o llocs web mitjançant els quals es poden fer accions semblants:

- Inserint les notes d'una partitura manualment, fet que pot ser molt feixuc i implicar molt de temps, o bé carregant-les des d'arxius *MusicXML*[2] o *MIDI*[3], els quals no són gaire comuns de trobar i costen d'obtenir, es pot escoltar el so de la melodia introduïda, com mostra la **Figura 2**.

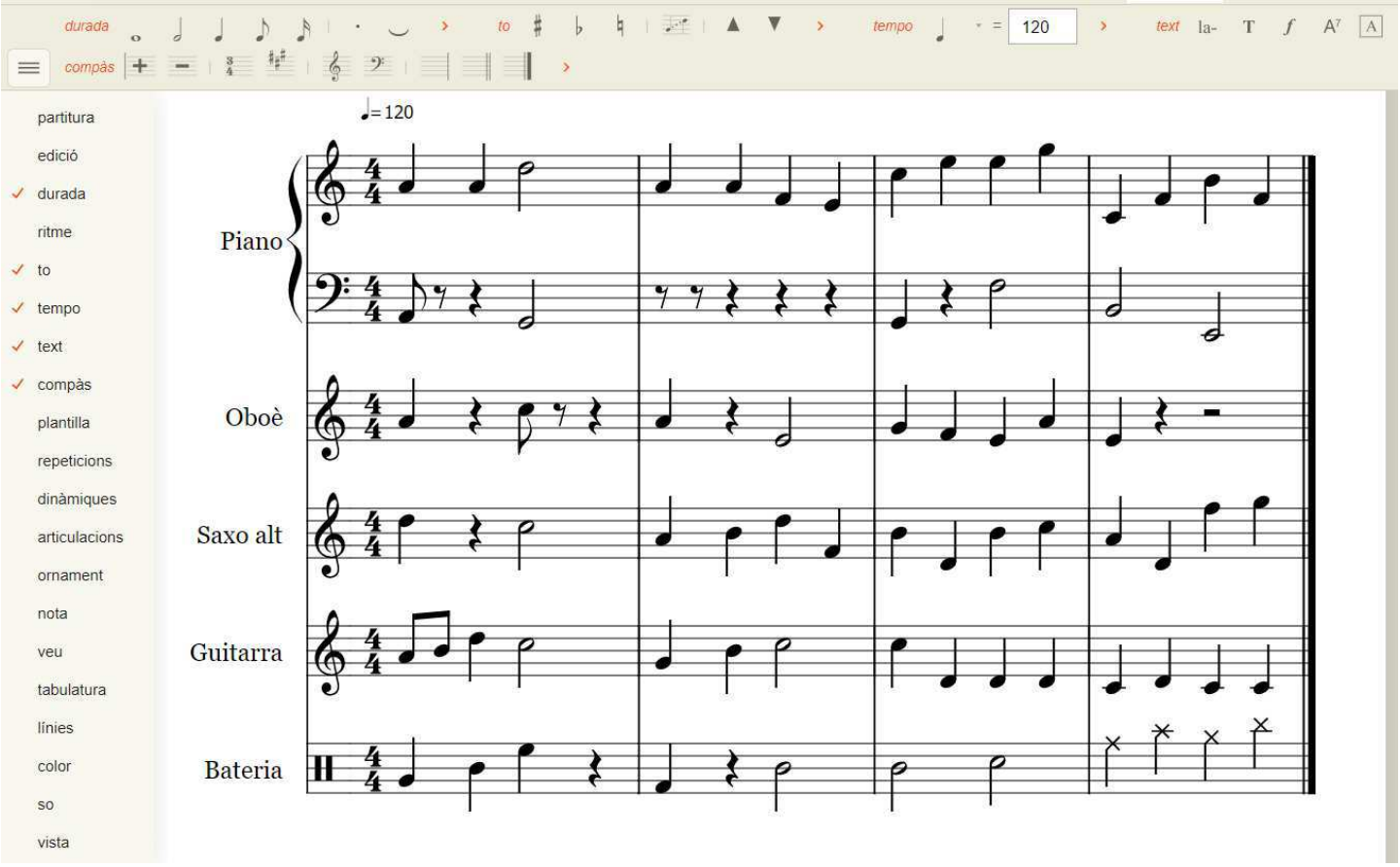

Figura 2.- Noteflight. (2019). *Introduint i configurant una partitura de música* [Captura de Pantalla Pròpia]. Recuperat de https://www.noteflight.com/

[2] *Music Extensible Markup Language.* Més informació a: https://ca.wikipedia.org/wiki/MusicXML
[3] *Musical Instrument Digital Interface*. Més informació a: https://es.wikipedia.org/wiki/MIDI

- A través d'una imatge d'un instrument qualsevol inserida en la aplicació o lloc web, com es mostra més avall en la **Figura 3** i **Figura 4**, es pot escoltar el so de la nota corresponent a on es clica amb el ratolí. És a dir, si es prem on hi ha col·locat virtualment el bombo (en el cas d'una bateria), s'escolta el so greu característic fet pel peu dret en pressionar la membrana; i si es clica la quinzena nota blanca del piano virtual, s'escolta un DO en quarta octava[4].

Figura 3.- Virtual Drumming. (2019). *Bateria virtual* [Captura de Pantalla]. Recuperat de https://www.virtualdrumming.com/es/como-tocar-la-bateria/juegos-de-bateria-virtual/tocar-la-bateria-virtual.html

Figura 4.- Recursive Arts. (2019). *Piano Virtual* [Captura de Pantalla]. Recuperat de https://recursivearts.com/es/virtual-piano/

Tot i que ambdós recursos tenen els seus avantatges (amb el primer es poden reproduir sons d'una gran varietat d'instrument alhora), també tenen els seus inconvenients (amb el segon és impossible poder assajar amb el propi instrument, serveix només per escoltar momentàniament).

De manera que, quan es planteja el projecte, s'agafen les parts bones de cadascun d'ells, i s'encara cap a la realitat que cap dels dos aborda: utilitzar el format de partitura amb el que actualment més treballen els músics[5], el d'imatge, ja siguin partitures físiques passades a l'ordinador per escàner, o simplement baixades d'Internet.

[4] Més informació a: https://ca.wikipedia.org/wiki/Octava_(m%C3%BAsica)
[5] Preguntat i corroborat per l'escola de música de Sant Celoni.

1.4. - Organització/Estructura de la memòria

Per a dur a terme l'elaboració de la memòria del projecte, aquesta s'ha desglossat en varis capítols que porten, pas a pas, cap la solució final del treball:

Capítol 2: Metodologia

En aquest segon capítol s'expliquen l'organització del projecte portada a terme i els mètodes utilitzats per arribar als resultats actuals. Així mateix, també es dóna una visió global del treball.

Capítol 3: Solució Tècnica

En aquest tercer capítol es detallen les cinc parts en què es divideix el treball, i s'entra més en matèria en cadascuna d'elles per aprofundir i ampliar la informació donada en el capítol anterior.

Capítol 4: Resultats i Conclusions

En aquest quart capítol es resumeixen els resultats obtinguts en aplicar la metodologia explicada en el capítol 2, i s'extreuen les conclusions finals del treball fet.

Capítol 5: Bibliografia

En aquest cinquè, i últim, capítol es llisten les fonts utilitzades per realitzar el projecte i citades dins del treball (ordenades alfabèticament) en format APA, de manera que els lectors puguin aprofundir més en matèria si es desitja.

Apèndixs i Annexes

Aquest segon volum serveix per a documentar, justificar i recolzar tot el que s'explica en aquesta memòria del projecte, on s'inclouen el codi del programa, el circuit elèctric o les justificacions dels càlculs dels motors, entre d'altres.

Capítol 2. - METODOLOGIA

En aquest segon capítol s'expliquen l'organització del projecte portada a terme i els mètodes utilitzats per arribar als resultats actuals. Així mateix, també es dóna una visió global del treball.

2.1. - Organització del Projecte i Mètodes Emprats

Per a dur a terme una correcta elaboració del projecte, s'ha emprat una metodologia de treball de manera que es pogués desenvolupar cada apartat de forma continuada:

1. Cerca d'informació

Primerament, un cop decidit el tema en el qual es basaria aquest projecte, esdevingut per la motivació explicada anteriorment, es va fer una recol·lecta d'informació necessària per a poder desenvolupar la memòria i el treball.

Es va anar a l'Ateneu de Sant Celoni (on s'imparteix música) per recordar els fonaments bàsics en els quals es basa una partitura de bateria, així com les parts que componen l'instrument, per tal de rebre una informació actualitzada i de primera mà. Es va decidir anar a aquest espai perquè l'autor d'aquest projecte va ésser alumne d'un professor que feia classes de bateria, fa uns anys.

Fruit de les converses, es constata que no totes les partitures de bateria són iguals, sinó que en algunes el *Bombo* pot estar a la primera línia del pentagrama, mentre que en algunes altres pot estar al segon espai. Aquesta circumstància ha esdevingut un punt clau que s'ha volgut introduir a l'aplicació: un apartat on es poguessin configurar la col·locació de les notes en la partitura.

2. Programació de l'aplicació

Ja que des d'un començament es planteja que el desenvolupament de l'aplicació es farà des del punt de vista del tractament i processat d'imatge (una partitura de bateria), s'ha optat per realitzar l'app en l'entorn de programació Matlab[6] (versió R2018b).

A més de comptar amb nombroses funcions que permeten aquest tractament d'imatge (algunes utilitzades en assignatures anteriors), Matlab possibilita les opcions de realitzar aplicacions GUI[7], i arxius instal·lables[8] per a aquestes, de manera que dóna valor afegit a l'app desenvolupada perquè permet poder ésser utilitzada des de qualsevol ordinador.

La programació es va realitzar en dues etapes:

- En la primera, es van desenvolupar totes les funcions necessàries per a realitzar el tractament de la imatge de partitura, cridades des d'un programa principal en el qual no hi havia cap part gràfica, sols es comprovava que les funcions realitzessin la seva tasca corresponent i, per tant, s'obtingués el resultat esperat (una variable amb el conjunt de notes a tocar, ordenades per ordre de ser tocades).
- En la segona etapa es va dissenyar la part gràfica de l'aplicació, s'hi van introduir les funcions creades i provades a la primera etapa, i es van afegir altres funcionalitats que podien dotar a l'aplicació d'un major control (edició de paràmetres) i funcionalitat (l'opció d'editar les notes en el pentagrama, mencionat en el punt previ).

3. Configuració de la xarxa de comunicacions

El principi en el qual es volia basar la xarxa de comunicacions per entrellaçar i connectar l'aplicació amb el microcontrolador encarregat de moure la maqueta era que fos una xarxa *wireless*[9]. Per aquest motiu, s'ha optat per utilitzar un protocol MQTT per enviar les dades necessàries per a transmetre les ordres corresponents al prototip.

A més, aquest protocol permet que ambdós dispositius (l'app controlada des d'un ordinador, i la maqueta) estiguin connectats a diferents *Wi-Fi*, fet que treu el límit de proximitat entre un i l'altre i possibilita la interacció entre ells des de llocs totalment separats.

[6] Més informació a: https://es.mathworks.com/products/matlab.html
[7] *Graphical User Interface*. Més informació a: https://ca.wikipedia.org/wiki/Interf%C3%ADcie_gr%C3%A0fica_d%27usuari
[8] Arxius *setup.exe* que permeten instal·lar l'aplicació i utilitzar-la sense la necessitat de tenir Matlab.
[9] Inal·làmbric, sense fils.

4. Disseny d'un prototip mecànic

La construcció de la maqueta s'ha dividit en dues parts:

La primera part, la més important, és la dels actuadors. S'ha decidit treballar amb servo motors, ja que de la gran varietat que el mercat ofereix, aquests són el tipus de motors més adients pel projecte, ja que aporten respostes ràpides (requisit necessari per a l'aplicació del treball) i un control de posició bastant precís. A més, els que no poden fer gaire força no solen ser cars, fet que és important en un prototip d'aquestes característiques.

D'altra banda, per controlar aquests motors i poder rebre les ordres de l'aplicació mitjançant un protocol MQTT, s'ha optat per utilitzar la microcontroladora Raspberry Pi 3[10], la qual a més de poder comptar amb accessibilitat a Internet (fet indispensable per connectar-se al protocol) mitjançant un port *Ethernet* o una targeta de *Wi-Fi*, incorpora dos pins que poden generar polsos de senyal PWM[11], que es creen internament per *hardware* amb els oscil·ladors que conté.

Tot i això, amb el prototip que es va pensar (dos motors per les mans, dos motors per girar aquests, i dos pels peus) no eren suficients dos pins, sinó que se'n necessitaven sis. I en aquest punt és on va ser decisiva la tria de la Raspberry, ja que incorpora la opció que permet generar aquests polsos PWM de manera programada, amb *software*, fet que (a priori) va semblar que seria suficient.

En la segona part, l'acústica de la maqueta, s'ha intentat crear una estructura semblant a l'instrument, per tal que els sons s'assemblessin el més possible als reals. Per fer-ho, en la segona part s'han utilitzat llaunes d'olives de diferents mides (per als timbals), les tapes d'aquestes (per als plats), un dispensador de tovallons de bar (per al so redoblant de la caixa), i una caixa de puros (per al bombo). A més, s'han utilitzat bastonets de menjar xinès amb els quals s'aconsegueix un so bastant semblant al de les baquetes reals.

[10] Tot i que és catalogada com a un ordinador monoplaca. Més informació a: https://ca.wikipedia.org/wiki/Raspberry_Pi
[11] *Pulse width modulation*. Senyal necessari per controlar els servo motors. Més informació a: https://ca.wikipedia.org/wiki/Modulaci%C3%B3_per_amplada_de_polsos

5. Muntatge

Un cop realitzat tot el disseny mecànic, s'ha implementat aquest fent ús de llistons de fusta (per a la base, les potes d'aquesta, i els suports elevadors), cargoleria, cola blanca i cola calenta, per tal de construir plenament el prototip de bateria musical.

A més, s'ha utilitzat la tecnologia FDM[12] per tal d'imprimir suports pels motors, dissenyats amb l'eina CAD-CAM[13] *PTC Creo Parametrics*, ja que aquesta eina permet fer prototipatges ràpids, funcionals i barats.

Finalment, s'ha implementat el circuit electrònic corresponent per controlar la maqueta, basat en la Raspberry Pi 3 i en una placa PCB, que serveix com a pont entre una font d'alimentació externa i els motors, per tal de no demanar massa càrrega al microcontrolador.

6. Anàlisi dels resultats i Conclusions

Per acabar, s'han realitzat les proves i els ajustaments adients finals (tant a l'aplicació com a la maqueta) per tal de posar en correcte funcionament el sistema, i poder valorar si s'han aconseguit els objectius plantejats o, en cas contrari, poder-ne conèixer les causes.

A més, s'han planificat quines serien les possibles línies de treball futur, les quals han anat sorgint durant l'execució d'aquest projecte.

[12] *Fused Deposition Modelling*. Altrament dita impressió en 3D. Més informació a: https://en.wikipedia.org/wiki/Fused_filament_fabrication#Fused_deposition_modeling
[13] Disseny i Fabricació Assistits per Ordinador. Més informació a: https://es.wikipedia.org/wiki/CAD/CAM

2.2. - Visió Global del Sistema

Un cop entesa l'organització i l'estructura del projecte, el projecte queda interrelacionat de la manera en què es mostra a la següent **Figura 5**.

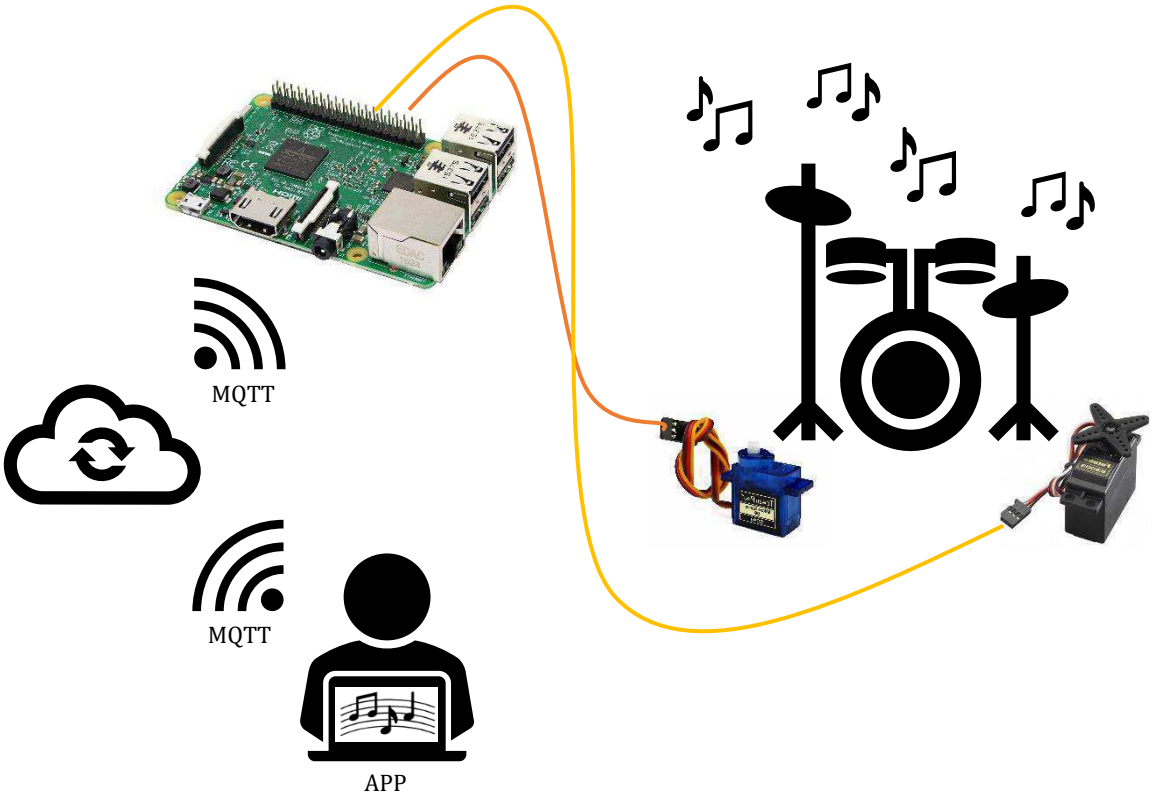

Figura 5.- Solà, M. (2019). *Esquematització del projecte* [Recurs Propi].

En cas de voler tenir una visió més complexa del sistema, es pot consultar l'**Apèndix I** del volum d'**Apèndixs & Annexes** per tal de visualitzar el mapa conceptual del treball, on s'han plasmat les diferents parts en què es divideix, així com també la interrelació que hi ha entre cadascuna d'elles.

Capítol 3. - SOLUCIÓ TÈCNICA

En aquest tercer capítol es detallen les cinc parts en què es divideix el treball, i s'entra més en matèria en cadascuna d'elles per aprofundir i ampliar la informació donada en el capítol anterior.

3.1. - Programació – Programa Lector de Partitures

En aquest apartat s'explica el funcionament de la part gràfica de l'aplicació. És a dir, com s'ha d'utilitzar per tal de llegir una partitura amb el programa, connectar-se a la Raspberry, i enviar-li les notes a tocar. En cas de voler saber què hi ha darrere la interfície gràfica (com treballen i estan estructurades les funcions), i voler veure el codi sencer encarregat de processar una imatge de partitura de bateria, es pot consultar la informació pertinent a l'**Apèndix II** del volum d'**Apèndixs & Annexes**, on s'expliquen individual i detalladament cadascuna d'elles.

Un cop obert el programa, apareixen tres "subfinestres" que permet realitzar tres accions diferents, però interrelacionades entre elles:

1. Lectura de Partitures:

Tant bon punt està arrencada l'aplicació, el primer que es veu és aquesta subfinestra. Tal com es mostra a la **Figura 6**, és una finestra buida, a excepció d'un botó a dalt a la dreta, el qual si es prem farà que s'obri un quadre de diàleg (**Figura 7**) per buscar i seleccionar la partitura que es desitja llegir.

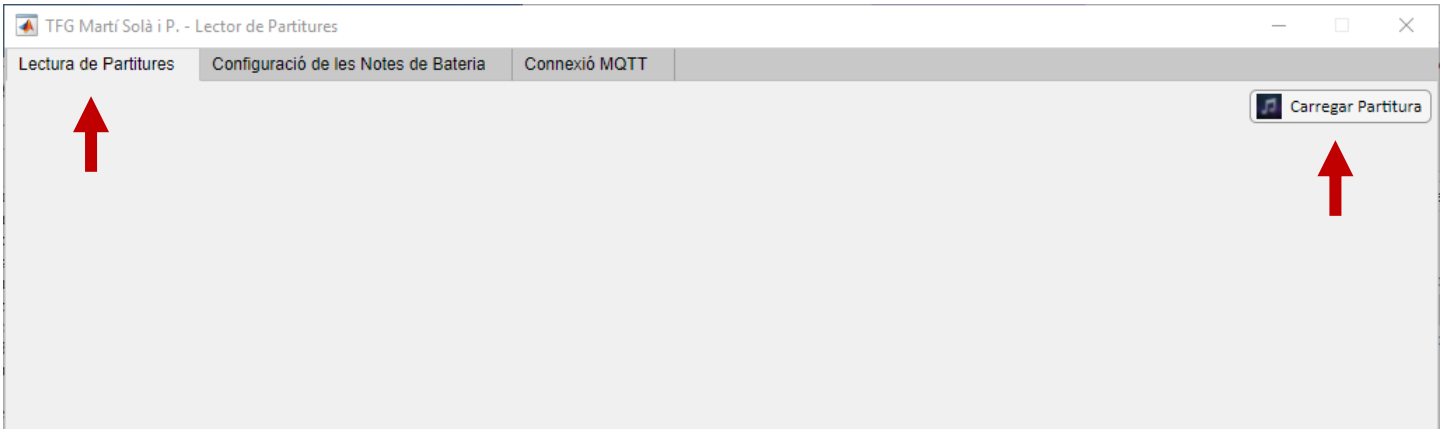

Figura 6.- Lector de Partitures. (2019). *Subfinestra 1- Lectura de Partitures, i botó per carregar una imatge de partitura de bateria* [Captura de Pantalla].

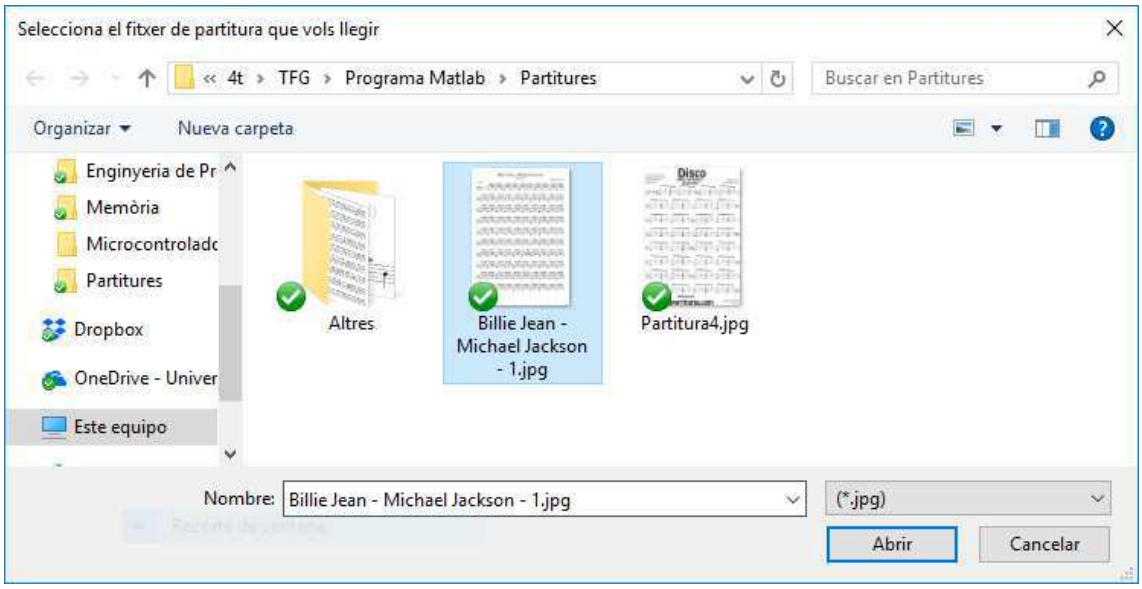

Figura 7.- Lector de Partitures. (2019). *Quadre de diàleg que s'obra al prémer el botó per carregar una imatge de partitura* [Captura de Pantalla Pròpia].

Un cop triada la partitura que es desitja llegir, es realitza automàticament un primer tractat d'imatge amb els valors dels paràmetres de lectura per defecte, com mostra la **Figura 8**.

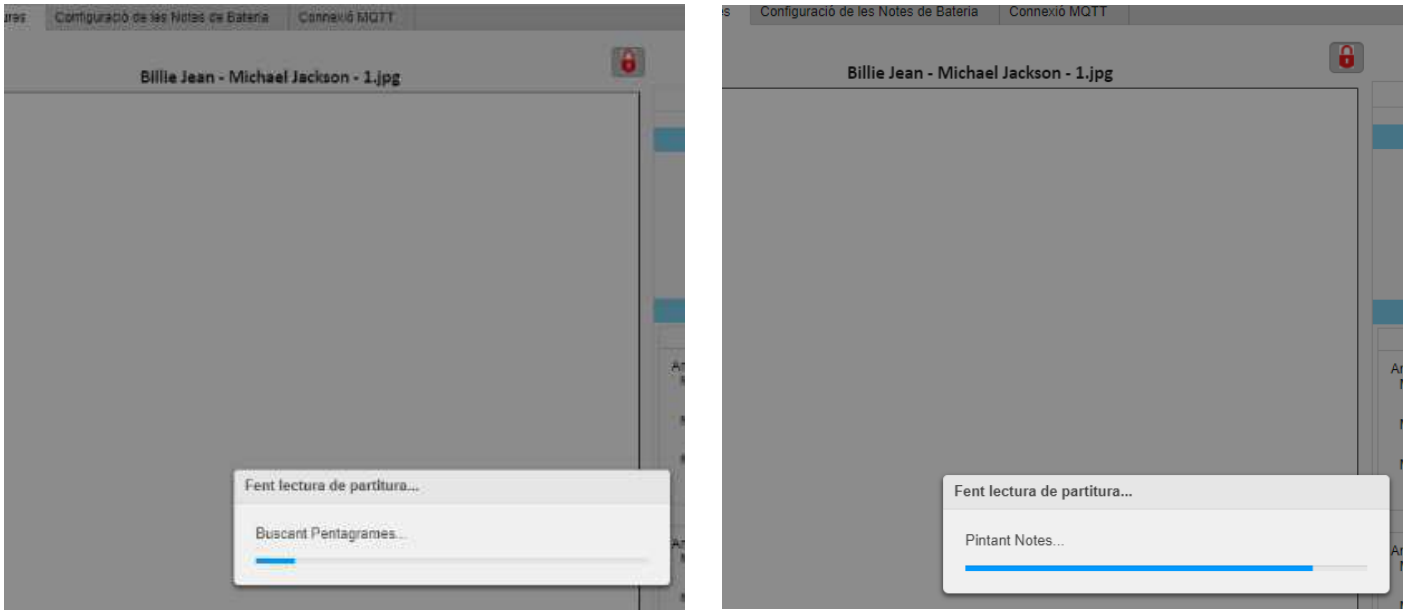

Figura 8.- Lector de Partitures. (2019). *Realitzant un 1r processat d'imatge* [Captura de Pantalla Pròpia].

Quan es llegeixi una partitura per primera vegada, és possible que es no es faci un correcte processat d'aquesta, ja que cada imatge té associat uns paràmetres adients per a aquesta, que vindran donats per la resolució que pugui tenir, la mida de les notes i del pentagrama, la quantitat d'aquests...[14] Per a modificar els paràmetres, es podrà fer des de dos llocs de la subfinestra, que permeten opcions diferents:

[14] Aquest fet és comentat a les conclusions, i n'és proposada una possible millora.

- Si es volen modificar **pocs** paràmetres, o ajustar **poc** els valors d'aquests, es podrà fer des del menú que hi ha a la dreta, com mostra la **Figura 9**, amb el qual es realitzarà tot el procés de tractament d'imatge per cada cop que es variï algun paràmetre.

Figura 9.- Lector de Partitures. (2019). *Menú per modificar individualment els valors dels paràmetres de processat* [Captura de Pantalla Pròpia].

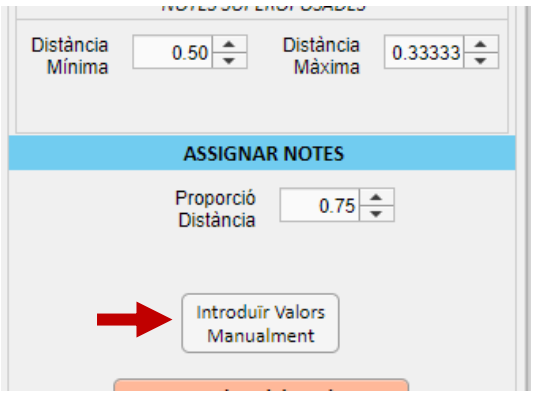

Figura 10.- Lector de Partitures. (2019). *Botó per obrir la finestra modificadora de paràmetres* [Captura de Pantalla].

- En canvi, si es volen modificar **bastants** o **tots** els paràmetres, o introduir **manualment** els valors d'aquests, es pot fer des de la nova finestra que s'obrirà en prémer el botó (**Figura 10**) que hi ha més avall del menú comentat en el punt anterior.

  Des de la nova finestra, es poden veure els resultats (**Figura 12**) de canviar els valors dels paràmetres prement els botons que tenen a la dreta (**Figura 11**), els quals només mostren per pantalla la part del processat que és afectada per a aquells canvis.

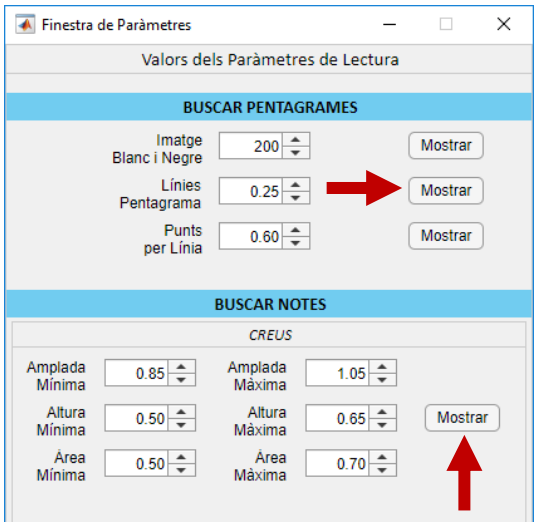

Figura 11.- Lector de Partitures. (2019). *Menú per modificar globalment els valors dels paràmetres de processat* [Captura de Pantalla].

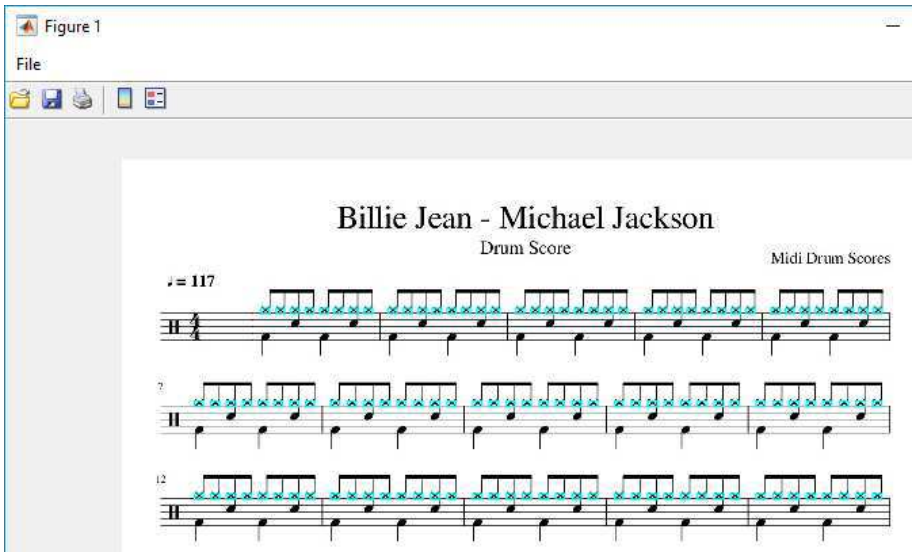

Figura 12.- Lector de Partitures. (2019). *Imatge que s'obra per pantalla on es mostren els resultats de canviar els paràmetres per la detecció de notes Creu* [Captura de Pantalla Pròpia].

Per a les explicacions de com trobar els paràmetres adients per a cada tipus d'imatge de partitura per fer-ne un correcte processat (**Figura 13**), es pot consultar la informació pertinent a l'**Apèndix III** del volum d'**Apèndixs & Annexes**, on s'expliquen individual i detalladament cadascun dels paràmetres, i com afecten al tractament de la imatge.

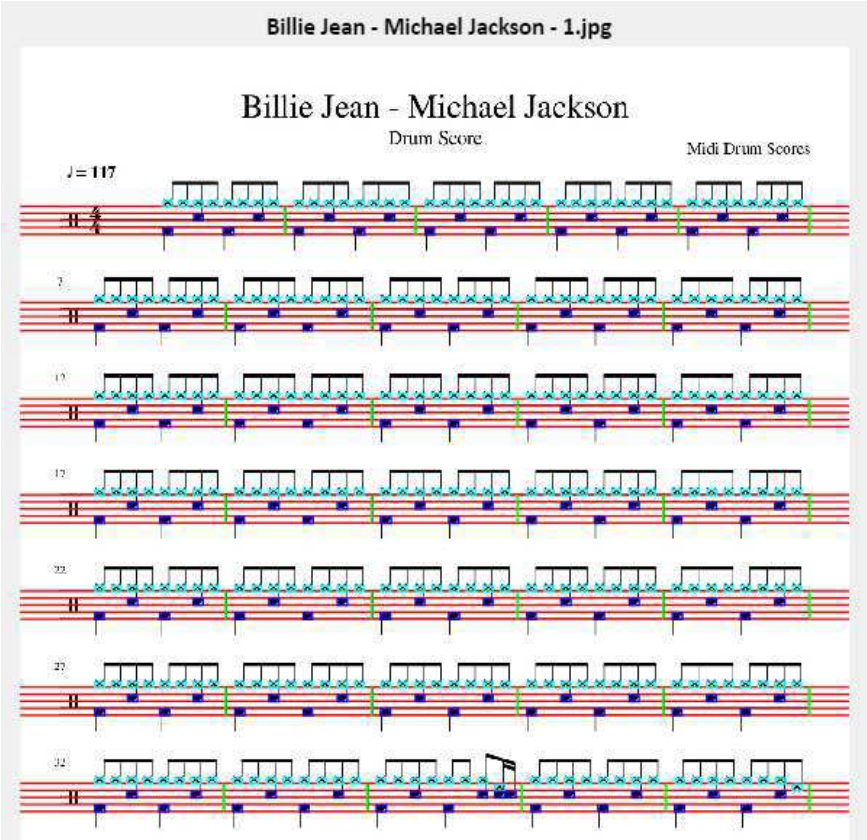

Figura 13.- Lector de Partitures. (2019). *Imatge que mostra un correcte tractament d'imatge i, per tant, la lectura de la partitura* [Captura de Pantalla Pròpia].

Un cop aconseguits els paràmetres correctes per la partitura i, per tant, un bon processat de la imatge, es poden guardar aquests valors prement el botó que hi ha sota el menú, que els guardarà en un fitxer de text amb el nom de l'arxiu de la partitura més el sufix "__ValorsParametres" dins la mateixa carpeta on es troba la partitura. D'aquesta manera, quan es vulgui llegir una partitura ja parametritzada, no caldrà re-configurar-la, sinó que simplement caldrà carregar l'arxiu de configuració mitjançant l'altre botó sota el menú (ambdós botons mostrats a la **Figura 14**).

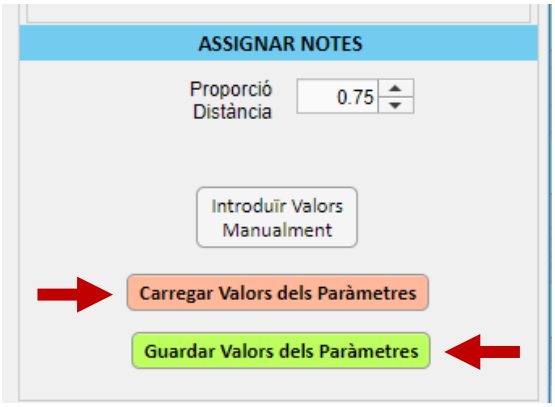

Figura 14.- Lector de Partitures. (2019). *Botons per guardar i carregar la configuració dels valors dels paràmetres del processat d'imatge* [Captura de Pantalla].

Com a detall estètic, s'ha afegit un botó amb forma de cadenat, mostrat a la **Figura 15**, el qual permet bloquejar i desbloquejar la imatge mostrada de la lectura de la partitura. Si està desbloquejada, es pot moure i ampliar, per si calgués fer *zoom* en alguna zona per comprovar el correcte funcionament de l'aplicació.

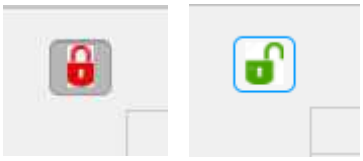

Figura 15.- Lector de Partitures. (2019). *Botó cadenat per bloquejar i desbloquejar la imatge de la partitura* [Captura de Pantalla].

2. Configuració de les Notes de Bateria:

Aquesta opció va aparèixer degut al que es comenta a l'apartat 2.1.1.- Cerca d'informació, quan es va descobrir que no totes les partitures de bateria tenen les mateixes notes col·locades a la mateixa posició, en contraposició a com sol ser amb altres instruments (a menys que es canviï de clau de pentagrama). Per tant, va sorgir la necessitat d'aquesta part de l'aplicació, on es poguessin editar les posicions de les diferents notes, i que aquest fet quedés reflectit al processat i tractament de la imatge de partitura.

Per fer-ho, sols cal prémer el botó que hi ha dalt a la dreta (**Figura 16**), i a continuació aniran sortint els diferents noms de les notes per a que se'ls assigni una posició clicant la nota corresponent dins del pentagrama (**Figura 17**). Primer se sol·liciten les que se simbolitzen amb boles (tambors), i després les que ho fan amb creus (plats), per temes d'ordre.

Figura 16.- Lector de Partitures. (2019). *Subfinestra 2- Configuració de les Notes de Bateria, i botó per editar les posicions dels diferents sons de l'instrument* [Captura de Pantalla].

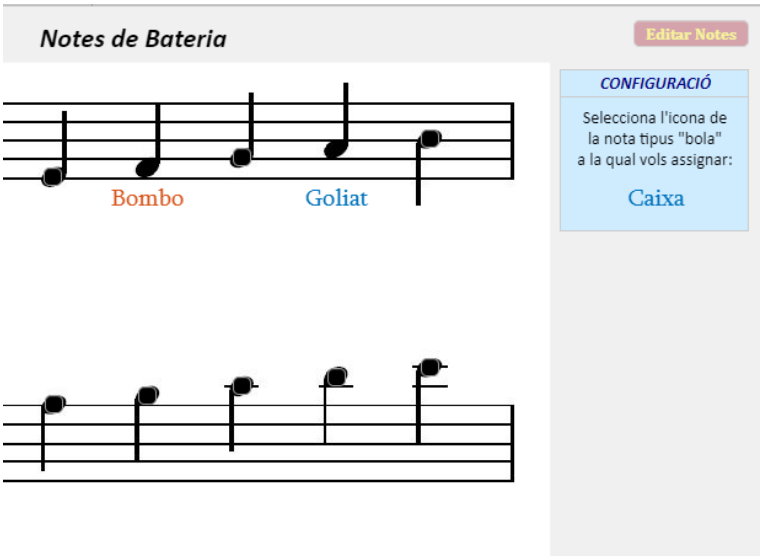

Figura 17.- Lector de Partitures. (2019). *Decidint la posició correcta de cada nota* [Captura de Pantalla Pròpia].

El resultat final que s'obté és una distribució de les diferents notes de l'instrument, tal i com es mostra a la **Figura 18**.

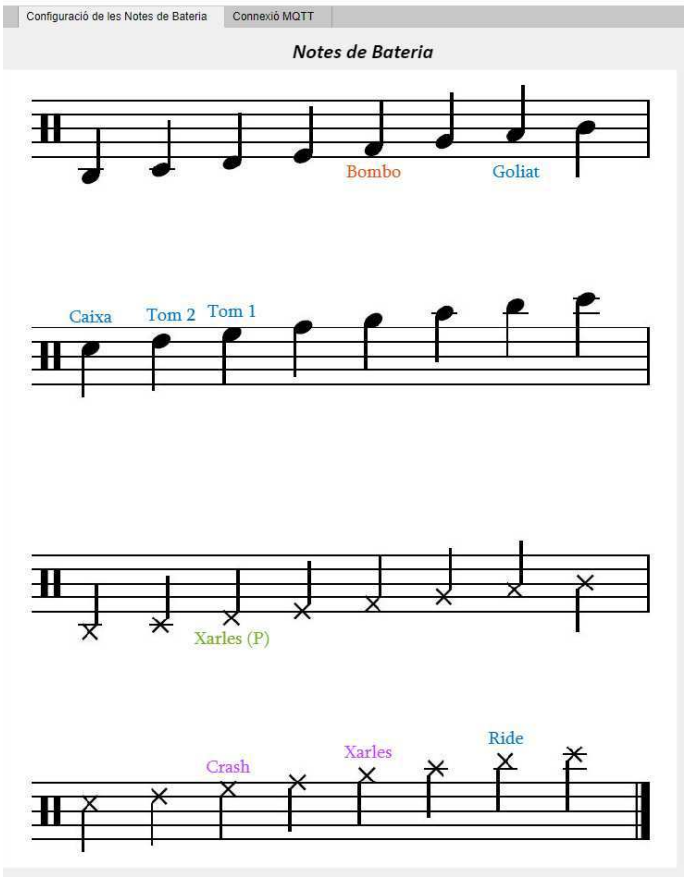

Figura 18.- Lector de Partitures. (2019). *Posicions finals dels diferents sons de l'instrument* [Captura de Pantalla].

3. Connexió MQTT:

I finalment aquesta última subfinestra (**Figura 19**) permet realitzar la connexió necessària per comunicar-se amb l'encarregat de fer moure i sonar la maqueta, la microcontroladora Raspberry Pi 3. El motiu pel qual s'ha utilitzat un tipus de protocol MQTT, i com funciona aquest, està explicat més avall, ja que aquest apartat se centrarà en explicar què s'ha de tenir en compte per tal de realitzar aquesta connexió des de l'aplicació.

Figura 19.- Lector de Partitures. (2019). *Subfinestra 3- Connexió MQTT, i menú de configuració per fer la connexió MQTT* [Captura de Pantalla].

Per a les explicacions de què significa cada paràmetre del menú de configuració de l'esquerra, es pot consultar la informació pertinent a l'apartat 3.3.- Xarxes – Protocol MQTT, on s'expliquen individual i detalladament cadascun d'ells, i com es poden identificar o trobar per tal de realitzar la connexió.

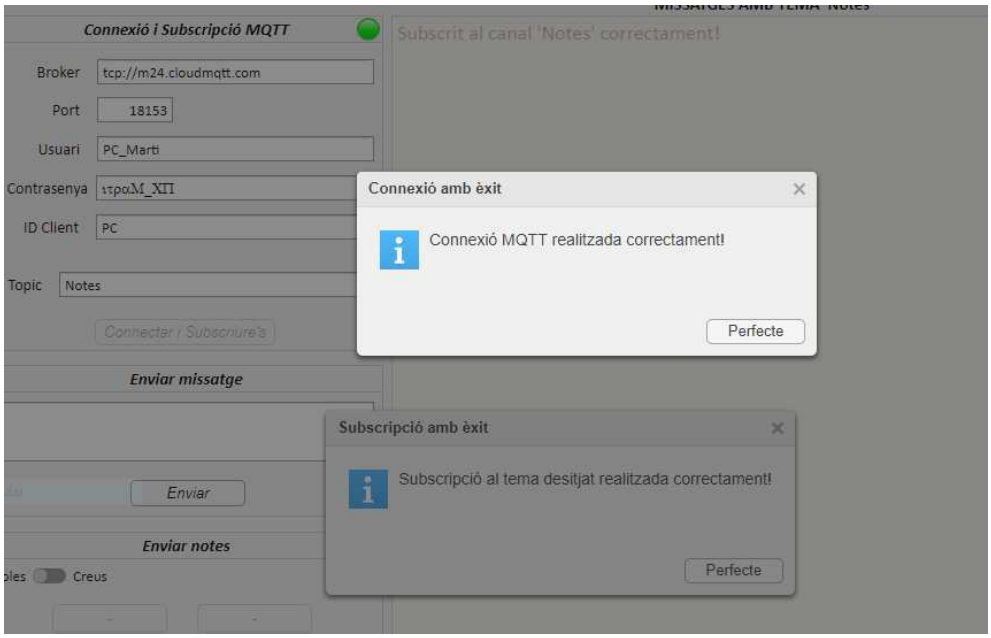

Figura 20.- Lector de Partitures. (2019). *Connexió establerta correctament amb la Raspberry Pi 3* [Captura de Pantalla].

Un cop configurada la comunicació corresponent (mostrat a la **Figura 20**), s'obren dos menús inferiors que permeten a l'usuari diverses opcions, amb diferents funcionalitats:

- Enviar missatge: Bàsicament serveix per a comprovar la connexió establerta. Envia el missatge que s'hagi escrit en format JSON[15] amb paraula clau "Text" (exemple mostrat en la **Figura 21**), i així es pot comprovar a la pàgina web que fa de Broker que la comunicació funciona correctament (**Figura 22**). Ara bé, al final del projecte es van implementar en el codi de la Raspberry unes funcions per tal de controlar el seu funcionament a distància, de manera que s'executessin certes ordres que s'enviessin des d'aquest menú, com ara "Reboot" (resetejar) o "Shutdown" (apagar).

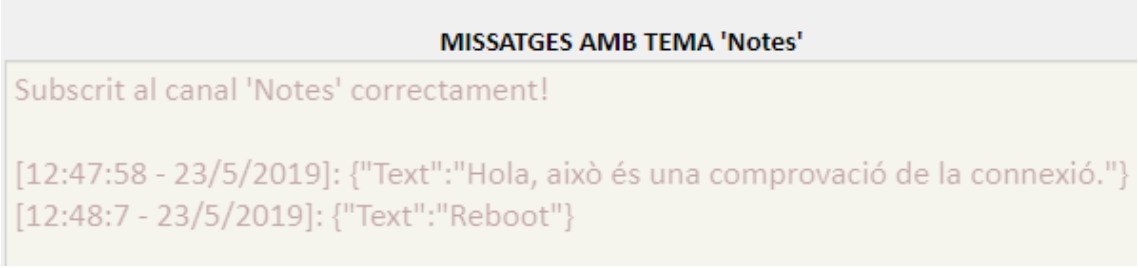

Figura 21.- Lector de Partitures. (2019). *Text mostrat per pantalla a l'enviar un text introduït dins l'editor de text* [Captura de Pantalla Pròpia].

[15] *JavaScript Object Notation*. Més informació a: https://ca.wikipedia.org/wiki/JSON

Figura 22.- CloudMQTT. (2019). *Text rebut i mostrat per pantalla* [Captura de Pantalla Pròpia]. Recuperat de *https://api.cloudmqtt.com/console/82595785/websocket*

- Enviar notes: Menú mostrat a la **Figura 23** que serveix per a comprovar que la maqueta funciona correcta i adequadament. És a dir, aquesta funció és utilitzada per assegurar que cadascuna de les diferents parts de l'instrument serà tocat, i només tocat, quan s'enviï a la Raspberry l'ordre de tocar la nota corresponent. Per exemple, quan s'enviï l'ordre de tocar la nota "Bombo", una baqueta ha de realitzar el moviment corresponent al que faria el peu dret, i fer sonar la caixa corresponent a aquesta part de la bateria.

  Aquesta funció, a més, ha estat utilitzada per l'autor d'aquest projecte per tal de comprovar que la programació feta a la Raspberry era la correcta, i que el posicionament que s'havia fet de la distribució de les diferents parts de l'instrument era l'idoni perquè el so fos el millor possible, permetent la possibilitat de fer ajustos ràpids per tal de trobar-lo.

Figura 23.- Lector de Partitures. (2019). *Menú per enviar notes individualment* [Captura de Pantalla].

- Enviar seqüència de notes: Botó gran sota els menús (**Figura 25**) que permet realitzar la segona funció principal del programa (la principal n'és el tractament de la imatge de partitura): permet enviar la variable llista que s'hagi generat, durant el processat, amb les notes que cal tocar de la melodia introduïda, i les envia per protocol MQTT cap a la Raspberry, encarregada de reproduir els sons (exemple mostrat en la **Figura 24**).

  Amb l'objectiu de mantenir un ritme constant, i alhora donés temps al prototip de posicionar-se per tocar correctament el que pertocaria, es va dissenyar una metodologia d'enviament de dades en la qual primer s'envia un valor on s'indica si cada extremitat toca o no en aquella franja de temps (i en cas de les mans, en quina part de l'instrument en concret), i seguidament un senyal en què es fa sonar als quatre alhora (en cas de no tenir un valor 0). D'aquesta manera, tot i no ser potser la forma més efectiva, s'aconsegueix un senyal rítmicament constant, com el que podria ser el d'un metrònom[16], per tal de permetre la possibilitat de tocar vàries parts de l'instrument de bateria alhora.

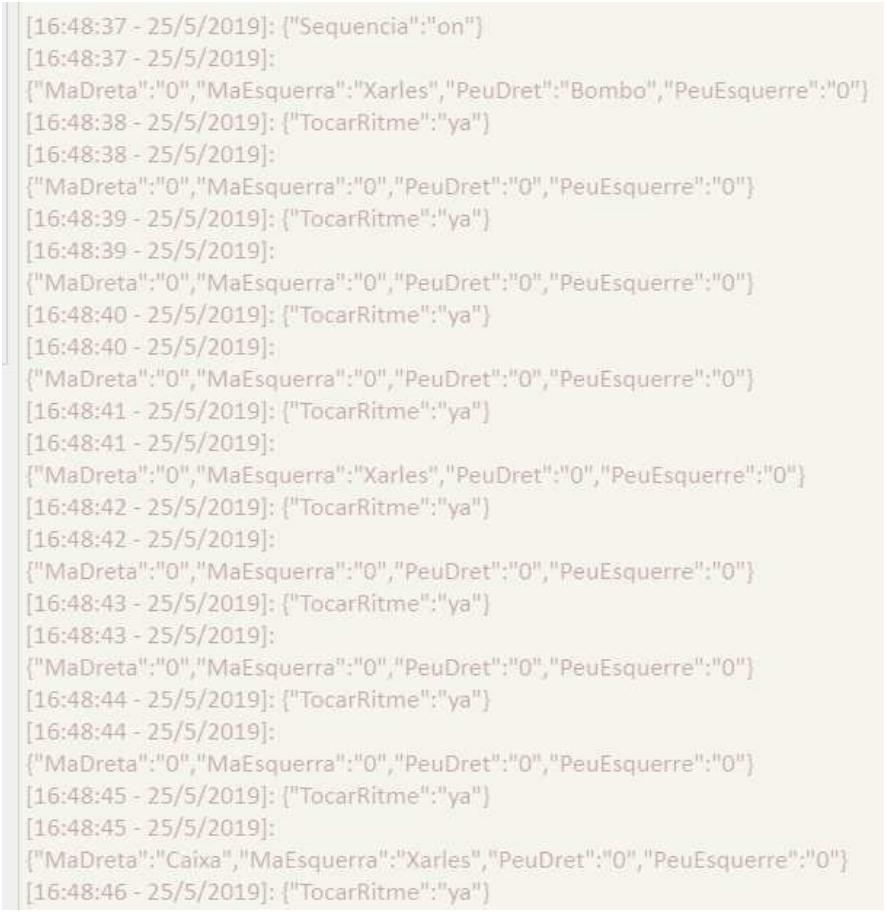

Figura 24.- Lector de Partitures. (2019). *Text mostrat per pantalla a l'enviar la seqüència de notes a tocar de la melodia introduïda* [Captura de Pantalla Pròpia].

[16] Més informació a: https://ca.wikipedia.org/wiki/Metr%C3%B2nom

- Veure seqüència: Botó petit sota l'anterior (**Figura 25**) que permet l'opció de visualitzar la *variable llista* generada pel programa, comprovar que s'ha realitzat un bon processat de la imatge, i corroborar que el resultat és l'esperat corresponent a la partitura seleccionada (a través d'un fitxer de text, mostrat a la **Figura 26**).

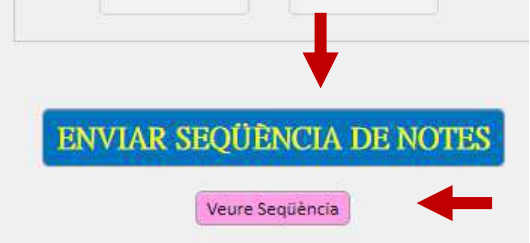

Figura 25.- Lector de Partitures. (2019). *Botons per enviar melodia i per veure la variable llista generada* [Captura de Pantalla].

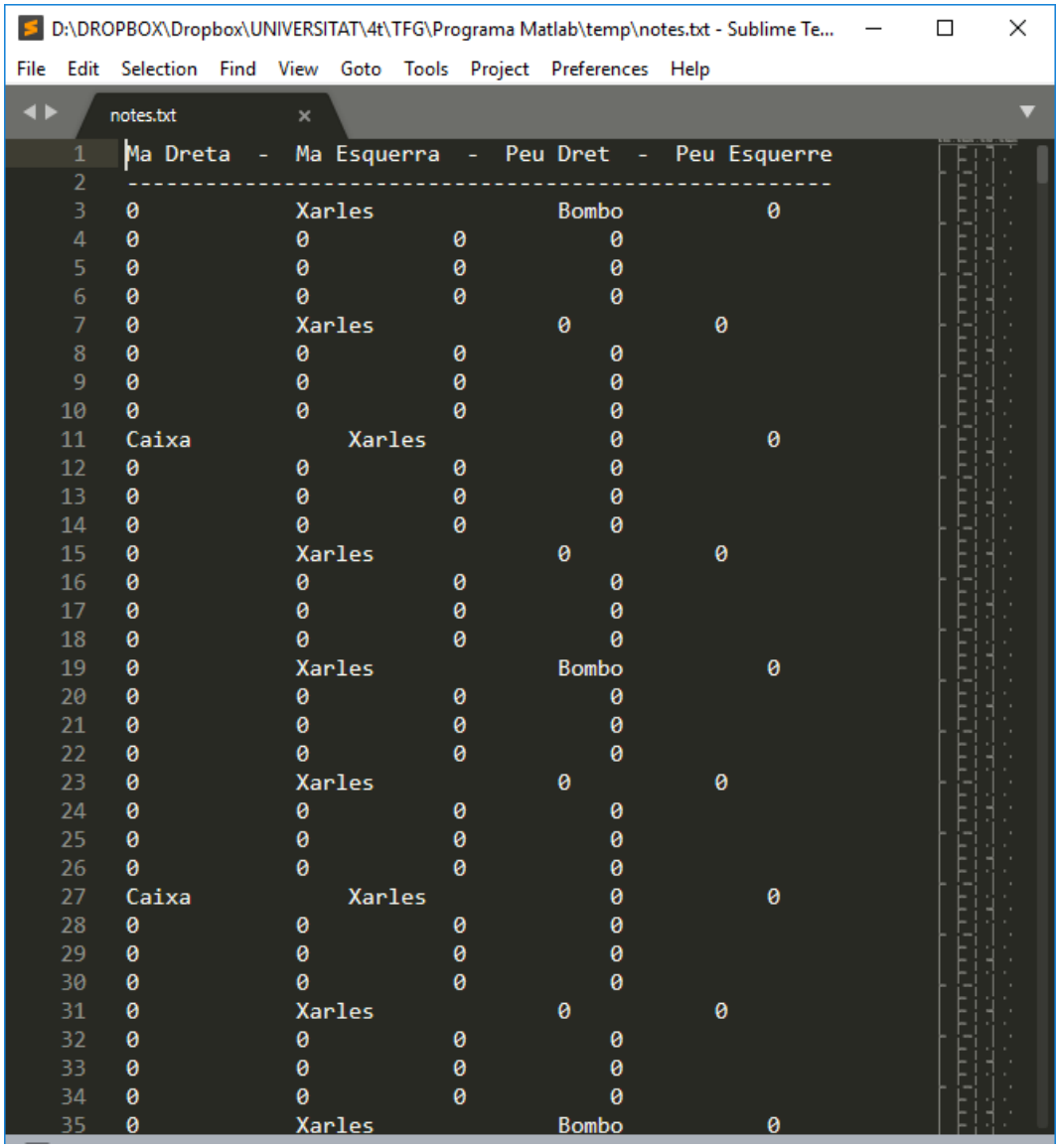

Figura 26.- Lector de Partitures. (2019). *Fitxer de text generat amb les dades de la variable llista de notes a tocar, subdividit en intervals de semifuses* [Captura de Pantalla Pròpia].

Totes aquestes funcions, menys la última, tenen la característica que mostren alguna cosa per pantalla, un text que es visualitza a la pàgina en blanc que hi ha a la dreta. Tot i això, aquesta característica sempre es pot desactivar amb el *switch* que hi ha sobre d'aquesta, mostrat a la **Figura 27**.

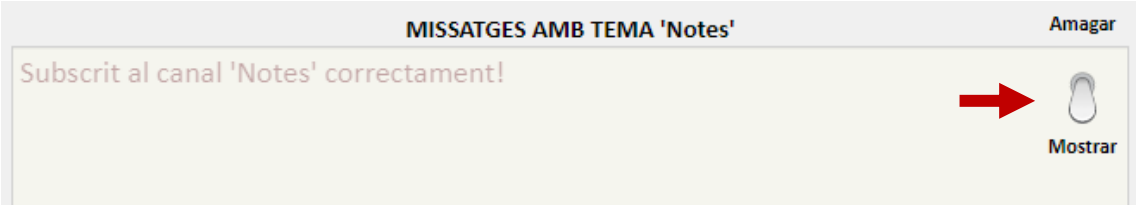

Figura 27.- Lector de Partitures. (2019). *Switch per amagar o mostrar per pantalla els missatges impresos per les diferents funcions* [Captura de Pantalla].

Això és útil sobretot amb la tercera funció esmentada amb anterioritat, ja que es va notar que el programa alentia el ritme a mesura que s'anaven acumulant els textos de les diferents notes enviades, i per tant aquest botó millora considerablement la velocitat a la qual treballa i és capaç de transmetre les dades pel protocol MQTT l'aplicació.

Per a fer possible que totes aquestes funcions explicades anteriorment es desenvolupin d'acord amb l'esperat, i estiguin estructurades dins la GUI d'una forma fàcil i entenedora per l'usuari, s'ha utilitzat l'entorn de desenvolupament *App Designer*, que incorpora el propi Matlab, i s'han consultat les pàgines web referenciades a la bibliografia com a punts [2] i [3].

En cas de voler veure el codi final escrit per generar la interfície gràfica, es pot consultar la informació pertinent a l'**Apèndix IV** del volum d'**Apèndixs & Annexes**.

3.2. - Programació – Raspberry

En aquest apartat s'explica l'estructura de la programació feta en la microcontroladora encarregada de fer sonar la maqueta. És a dir, quin procediment segueix per connectar-se al protocol MQTT i escoltar-hi, o com es controlen els motors de tal manera que es pugui fer sonar alhora. En cas de voler saber què hi ha darrere (funcions que hi treballen), i voler veure el codi sencer, es pot consultar la informació pertinent a l'**Apèndix V** del volum d'**Apèndixs & Annexes**, on s'expliquen individual i detalladament cadascuna d'elles.

S'ha desenvolupat el comportament de la Raspberry amb el llenguatge de programació d'alt nivell Python[17], ja que és un dels llenguatges de programació més potents, i a més és *open-source*, fet que obre les portes a la imaginació del que es vulgui programar.

Per tal de poder aconseguir que diferents notes sigui tocades en el mateix instant s'ha utilitzat una llibreria que permet a la microcontroladora fer *multithreading/multitasking*[18], de manera que varis programes "corren" alhora, on cadascun d'ells controla un motor diferent. Aquests diferents subprogrames queden a l'espera d'una ordre del programa principal, la qual els avisa quan han de tocar, i per tant ho fan de manera simultània i sincronitzada. Per a poder implementar aquesta característica dins del programa, i que funcioni d'acord amb l'esperat, s'han consultat les pàgines web referenciades a la bibliografia com a punts [9], [11] i [13].

L'estructura que segueix el programa és la següent:

1. S'importen totes les llibreries necessàries, i es declaren les variables globals que s'utilitzaran durant tot el programa, com ara les posicions dels timbals/plats, entre d'altres.
2. Es defineixen les diferents funcions que seran cridades pel programa principal, com la de moure el motor del peu dret.
3. S'inicia el programa principal, que segueix el següent procediment:
   3.1. S'inicialitzen *variables esdeveniments* que seran utilitzades en les crides als subprogrames comentats anteriorment i que realitzaran la seva funció específica.
   3.2. Es creen els diferents subprogrames, cadascun dels quals controla un motor diferent (ja que s'hi associa la funció de moure el motor corresponent), i s'inicien per tal que ja quedin a l'espera de rebre ordres.
   3.3. Es declaren els pins per on es controlaran els motors (quatre per *software* i dos per *hardware*), i s'inicien enviant-los un primer senyal PWM amb la posició inicial.

[17] Més informació a: https://ca.wikipedia.org/wiki/Python ; https://www.python.org/about/
[18] Més informació a: https://en.wikipedia.org/wiki/Multithreading_(computer_architecture)

3.4. Es crea una connexió MQTT i subscripció al canal (funcionament explicat a l'apartat 3.3.- Xarxes – Protocol MQTT. I es queda a l'espera de rebre missatges.

Un cop es rep un missatge, es re-converteix de format JSON a variable, i depenent de la paraula clau, s'actua de la següent forma:

- Si és un text, i és una de les ordres establertes per seguretat i control (com apagar o reiniciar la placa), s'executa la funció que compleix l'ordre.
- Si és una sola nota, el programa crida una 1a funció per tal de posicionar la baqueta, i a continuació una segona funció que la fa moure, generant el so. En cas que la nota sigui per ser tocada amb un peu, només es crida aquesta segona funció.
- Si és una nota que pertany a la seqüència, es distingeix entre si és tocada amb una mà o un peu, i amb quin, i es posicionen els motors, en cas que el valor rebut sigui diferent a 0 (de forma paral·lela, utilitzant les *variables esdeveniments* que criden als subprogrames). Un cop col·locats, queden a l'espera de rebre l'ordre rítmica per tocar.
- Si és l'ordre de tocar (variable rítmica explicada en un secció dins l'apartat anterior), s'activen unes altres variables esdeveniments que fan accionar els diferents motors alhora, en cas de no tenir valors de 0.

Aquestes dues últimes funcions es repeteixen durant el procés d'enviament de seqüència des de l'aplicació, amb el qual s'aconsegueix una metodologia de: posicionar, tocar, posicionar, tocar, etc.

Finalment, es configura la microcontroladora per tal que arrenqui el programa explicat de forma automàtica cada cop que s'engegui, de manera que no s'hi hagi d'interactuar directament cada cop que s'alimenta la placa per fer tocar la maqueta.

Per a poder configurar la Raspberry d'aquesta forma, s'ha consultat la pàgina web referenciada a la bibliografia com a punt [12].

Comentar que en aquest projecte, per tal de poder fer funcionar la programació desenvolupada per fer sonar el prototip, s'ha connectat la Raspberry al Wi-Fi que és capaç de generar el portàtil de l'autor d'aquest projecte (que bàsicament fa de punt de cobertura, com un encaminador[19]), ja que les diferents xarxes que subministra la universitat o bé són inaccessibles, o bé hi restringeixen l'accés.

[19] *Router* en anglès.

3.3. - Xarxes – Protocol MQTT

En aquest apartat s'explica què és i en què es basa el protocol utilitzat per comunicar l'aplicació amb la Raspberry, i quines funcions s'han utilitzat a ambdós dispositius per connectar-s'hi.

L' MQTT[20] és un protocol M2M[21] de missatges *publicar-subscriure's* basat en el protocol TCP/IP[22]. Segueix una topologia de tipus estrella (**Figura 28**), en la qual hi ha incorporat un node central que fa de servidor o *broker*, amb una capacitat de fins a 1000 clients. El *broker* és l'encarregat de gestionar la xarxa i de transmetre els missatges. Per tal de mantenir actiu el canal, els clients envien paquets cap aquest, i esperen la resposta del servidor.

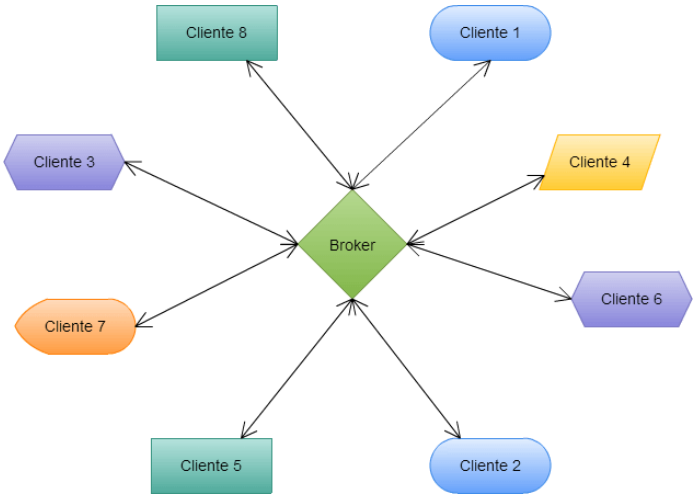

Figura 28.- Geeky Theory. (2019). *Estructura amb tipologia estrella del protocol MQTT* [Imatge]. Recuperat de *https://geekytheory.com/que-es-mqtt*

La comunicació es basa en temes o *tòpics*, els quals són creats pel clients que publiquen un missatge, i on han d'estar subscrits els nodes que desitgin rebre la informació enviada per aquest. Es representen mitjançant una cadena i tenen una estructura jeràrquica, separada per " / " (exemplificat a la **Figura 29**).

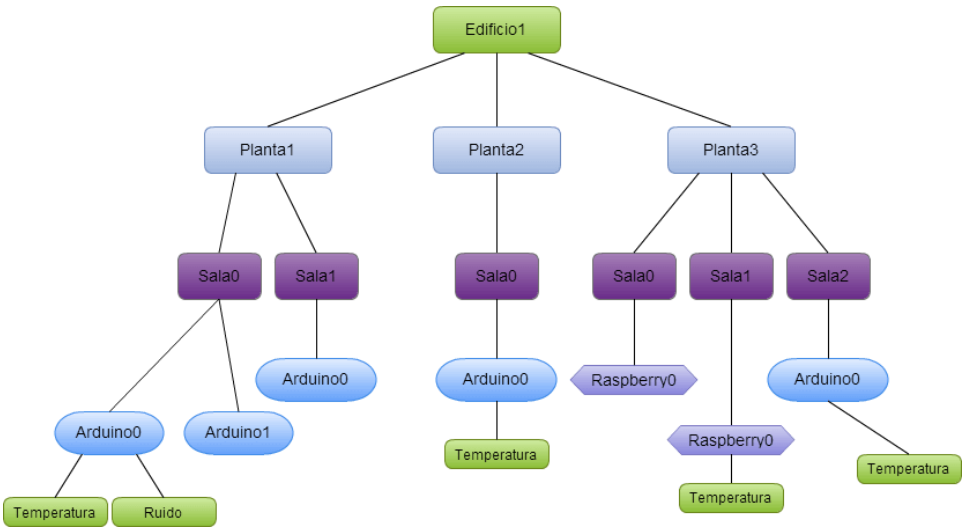

Figura 29.- Geeky Theory. (2019). *Possible estructura jeràrquica d'un canal MQTT* [Imatge]. Recuperat de *https://geekytheory.com/que-es-mqtt*

[20] *Message Queuing Telemetry Transport*. Més informació a: https://ca.wikipedia.org/wiki/MQTT
[21] *Machine to Machine*.
[22] *Transmission Control Protocol & Internet Protocol*. Més informació a: https://en.wikipedia.org/wiki/Internet_protocol_suite

En aquest projecte s'ha utilitzat un servidor de broker MQTT online, de manera que no calgués que tant l'aplicació com la maqueta estiguessin connectats entre ells directa o indirectament, sinó que fos indistinta la seva localització sempre i quan ambdós tinguessin accés a Internet. S'ha emprat la pàgina web CloudMQTT[23] per tal de realitzar la funció de comunicar els dos dispositius de manera inal·làmbrica, tot i utilitzar-se amb una llicència gratuïta bastant limitada[24].

A continuació es detalla què significa cada paràmetre del menú de configuració de la connexió MQTT de l'aplicació, i on es poden identificar o trobar dins la pàgina web (aquests dos primers mostrats en la **Figura 30**):

- Broker: ID/URL del servidor al qual ens volem connectar, que en aquest cas és la pròpia pàgina web. És únic per connexió, i està basat en una adreça TCP/IP.
- Port: Ve donat pel servidor un cop s'ha registrat un compte i escollida una llicència de treball.

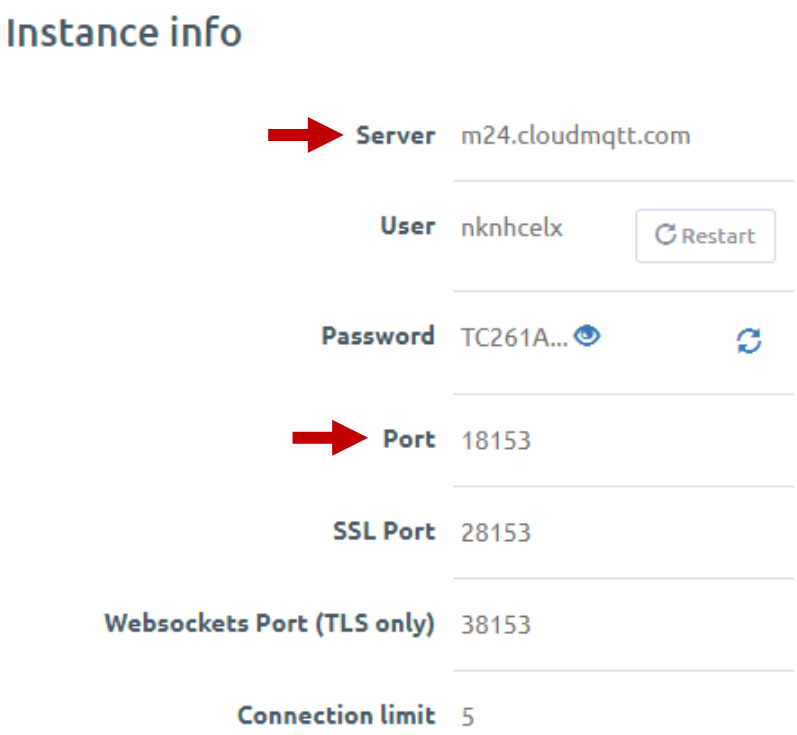

Figura 30.- CloudMQTT. (2019). *Paràmetres de configuració per realitzar la connexió MQTT* [Captura de Pantalla Pròpia]. Recuperat de *https://api.cloudmqtt.com/console/82595785/detailsbsocket*

[23] URL: https://www.cloudmqtt.com/
[24] En nombre de clients permesos (5 usuaris), i velocitat d'enviament de dades (10 Kbit/s).

- Usuari & Contrasenya: Claus per accedir al servidor. Es va decidir implementar aquesta funcionalitat (ja que hi ha l'opció d'establir el servidor públic, sense requeriment d'aquestes claus) per tal d'evitar accessos no desitjats i, per tant, un possible alentiment i saturació del servidor. Accessibles des de la pàgina web, com mostra la **Figura 31**.

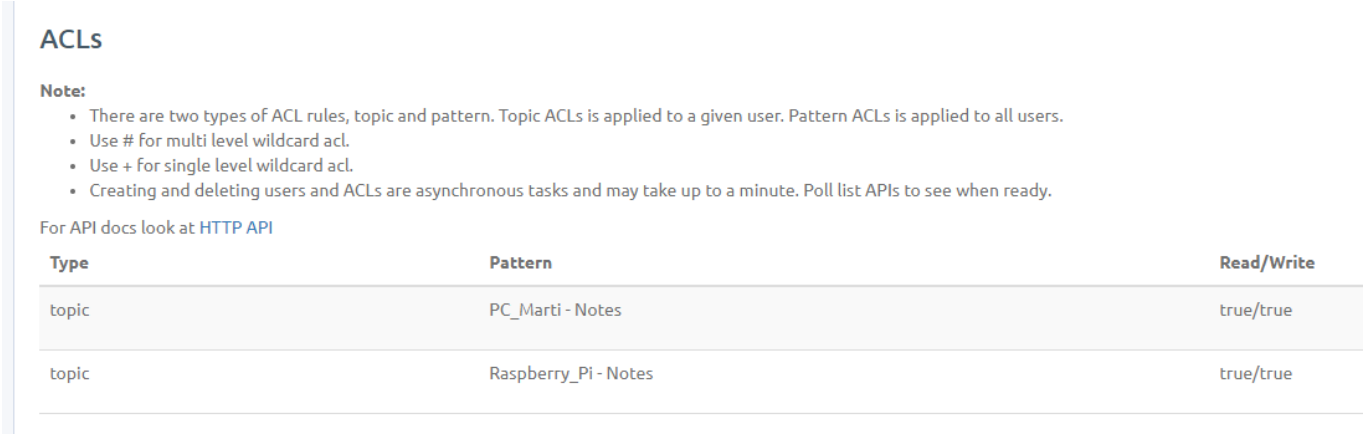

Figura 31.- CloudMQTT. (2019). *Usuaris admesos al servidor, amb els respectius permisos (d'escriptura i lectura)* [Captura de Pantalla Pròpia]. Recuperat de *https://api.cloudmqtt.com/console/82595785/users*

- ID Client: Nom amb el qual es veu al dispositiu dins del servidor. Introduït posteriorment per tal poder visualitzar d'una millor manera quan es realitzava correctament la connexió i qui enviava cada missatge.
- Topic: Com que no calia implementar una jerarquia[25] en aquest projecte, sinó una sola direcció a la qual adreçar els missatges de les notes a tocar, se'l va anomenar simplement "Notes".

Per a crear el servidor amb protocol MQTT, així com a restringir-hi l'accés a sols els dos usuaris creats, s'ha utilitzat la documentació que incorpora la pròpia pàgina web que fa de servidor, referenciada a la bibliografia com a punt [4].

Per a programar l'aplicació i la microcontroladora d'aquest projecte s'han emprat les funcions donades pel propi equip de Matlab i la documentació amb codi Python de la pròpia pàgina web CloudMQTT, referenciades a la bibliografia com a punts [7] i [5], respectivament.

[25] S'utilitzaria quan es vol que un dispositiu "escolti" a varis canals (Oficina1/#).

3.4. - Disseny Mecànic – Motors & Maqueta

Un cop desenvolupada l'aplicació i comprovada la seva funcionalitat, es desitja construir un prototip de bateria musical amb materials reciclats per tal de demostrar l'eficàcia d'aquesta d'una forma més visual. I en aquest cas, també més sonora. S'han agafat com a referents vídeos on es mostren dissenys de prototips o prototips finals tocant una bateria, referenciats a la bibliografia com a punts [1] i [10].

Es dissenya una distribució de les diferents parts de l'instrument de manera que amb sols quatre extremitats (quatre motors units a quatre baquetes, que simulen les dues mans i els dos peus) es puguin arribar a fer sonar totes, fent aquesta maqueta més propera a un model comercial. Per a moure les dues mans, s'utilitzen uns altres dos motors.

A l'hora de triar el tipus de motors que s'havien d'utilitzar d'entre la gran varietat que el mercat ofereix, es va seguir el criteri d'escollir aquells que oferissin una major resposta de moviment i un control de posició bastant precís. I els que complien ambdós requisits, a més de no ser gaire cars i de força no molt gran, eren els Servo Motors.

Per tal de decidir la força que havien de ser capaços de realitzar, així com la velocitat a la qual havien de poder arribar, es van extreure uns DSL[26] a partir dels esquemes que es mostren més avall, en la ¡Error! No se encuentra el origen de la referencia. i ¡Error! No se encuentra el origen de la referencia., per tal d'esquematitzar els sistemes en qüestió, i poder-

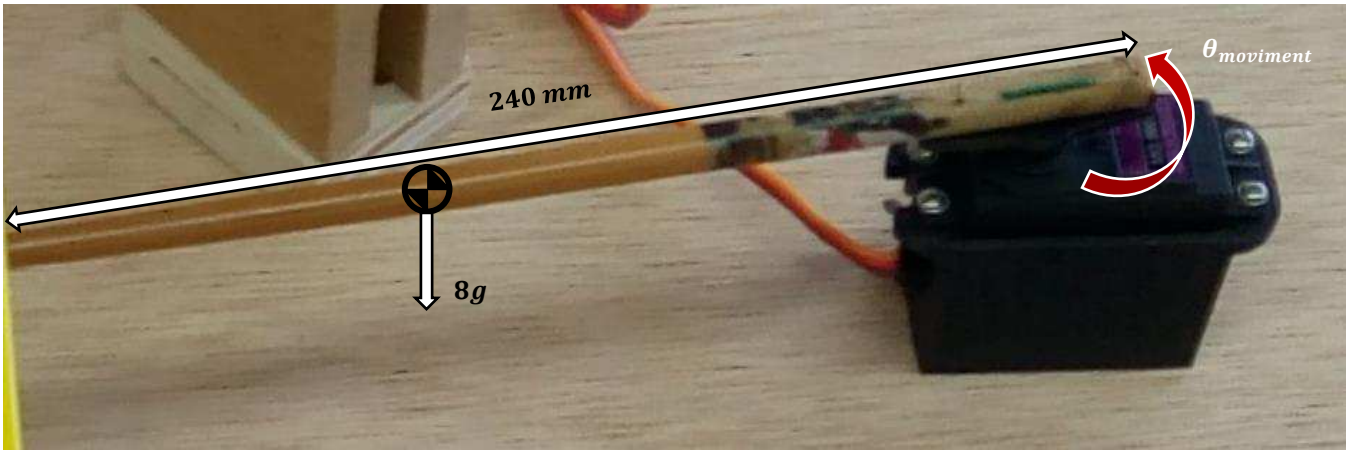

Figura 32.- Solà, M. (2019). *Esquema dels motors "extremitats"* [Imatge].

ne obtenir els paràmetres esmentats.

[26] Diagrama del Sòlid Lliure. Més informació a: https://ca.wikipedia.org/wiki/Diagrama_del_cos_lliure

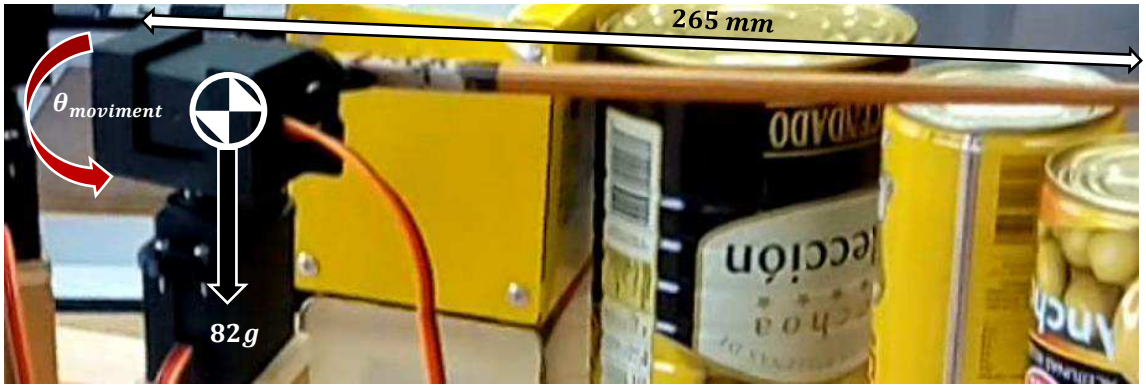

Figura 33.- Solà, M. (2019). *Esquema dels motors "mans"* [Imatge].

Per veure els DSL's o el procediment seguit per tal d'extreure els valors comentats, es pot consultar la informació pertinent a l'**Apèndix VI** del volum d'**Apèndixs & Annexes**.

Els motors que compleixen els requeriments explicats anteriorment es troben referenciats a la bibliografia com a punts [6] i [8], respectivament, i són els que es van comprar. Per veure els respectius *datasheets*, es poden consultar als **Annexes I** i **II** del volum d'**Apèndixs & Annexes**.

Tot i això, i com es comenta en l'apartat següent 3.5.- Electrònica – Circuiteria, la majoria dels servo motors estan controlats per senyals PWM generades per la pròpia Raspberry per *software*. Això fa que el senyal que reben aquests motors sigui inestable, i en el cas dels petits (que tenen un realimentació pel posicionament bastant deficient) ocasiona que perdin la seva posició. En l'intent de recuperar-la, es perd el control d'aquests ja que es tornen bojos reposicionant-se, per culpa de la inèrcia que guanyen deguda al pes de la baqueta.

Per a aquest motiu (que no es va descobrir fins ben avançat el prototip i amb els motors ja comprats), es decideix **sobredimensionar** expressament i utilitzar els mateixos servos grans, que no presenten aquest fet de descontrol, per a totes les funcionalitats del prototip.

Un cop realitzat tot el disseny mecànic, s'ha implementat aquest fent ús de llistons de fusta (per la base, les potes d'aquesta, i els suports elevadors), cargoleria, cola blanca i cola calenta, per tal de construir plenament el prototip de bateria musical.

A més, s'ha utilitzat la tecnologia FDM per tal d'imprimir suports pels motors, dissenyats amb l'eina CAD-CAM PTC Creo Parametrics.

Malauradament, per falta de temps, no s'ha pogut implementar la interacció del peu esquerre amb la maqueta, tot i que es va fer un primer disseny[27] del mecanisme que es podria haver arribat a construir per tal de dur-ho a terme, mostrat a la següent **Figura 34**:

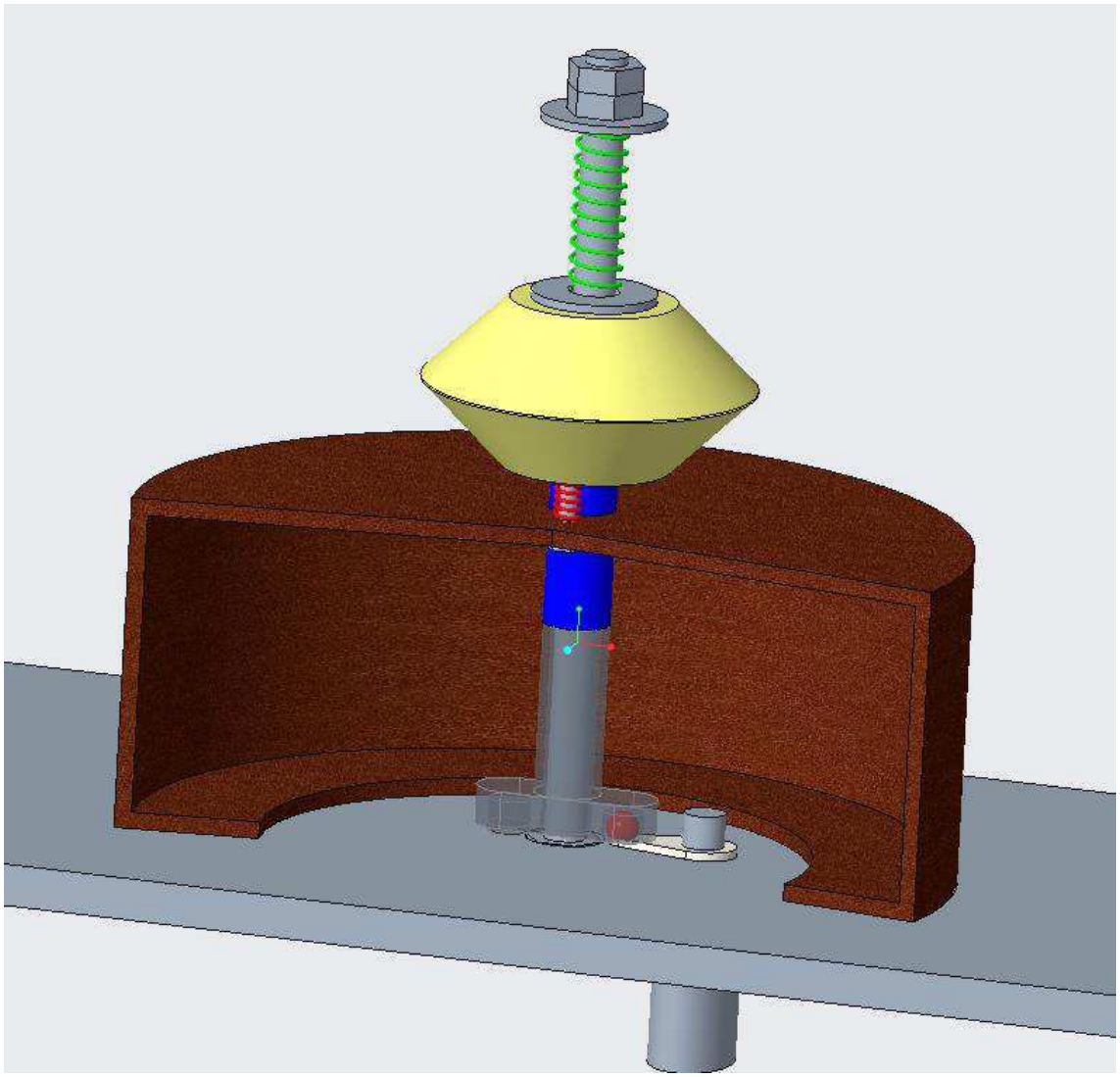

Figura 34.- Vallverdú, G. (2019). *1r disseny del mecanisme per fer picar els plats entre elles (xarles). Consta de 3 molles, un suport per aquestes. i un perfil tubular que, amb l'ajuda d'una bola, fa pujar i baixar un dels plats* [Captura de Pantalla Pròpia].

[27] Amb l'ajuda i col·laboració directa d'un company de classe i amic, que hi entén més en l'àmbit de la mecànica.

Aquest ha estat el resultat final de la construcció de la maqueta:

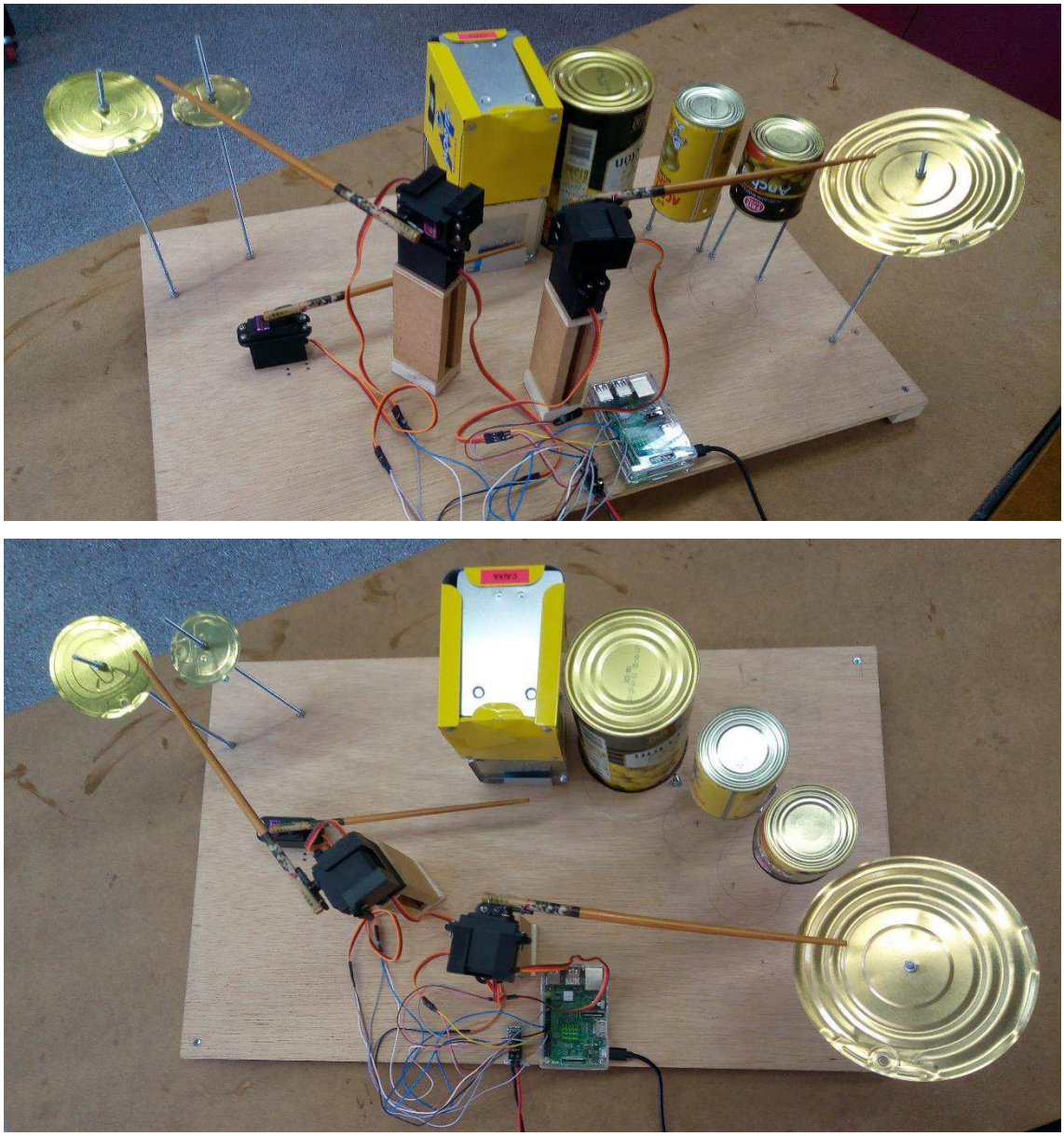

Figura 35.- Solà, M. (2019). *Maqueta final construïda* [Imatges].

3.5. - Electrònica – Circuiteria

Un cop feta la programació de l'aplicació i escollits el motors adequats per fer sonar la maqueta, el següent pas va ser triar el microcontrolador idoni per connectar aquestes dues parts. I aquest va ser la Raspberry Pi 3, la qual, a més de comptar amb accés a Internet, incorpora suficients pins per comandar els motors, en el quals es poden generar senyals PWM (dos amb *hardware* i quatre per *software*, a partir d'una llibreria).

Per temes de fiabilitat, es va escollir comandar els 2 motors amb baquetes que simulen les mans des dels dos pins amb senyals PWM per *hardware*, i els altres quatre, comandats per *software*:

- Pins *hardware*: 12 (GPIO18) i 33 (GPIO13)
- Pins *software*: 15 (GPIO22), 16 (GPIO23), 18 (GPIO24) i 22 (GPIO25)

Un cop connectats els servo motors amb la microcontroladora, es va decidir alimentar-los a +6V[28] des d'una font d'alimentació externa, per tal d'assegurar-se que reben el corrent necessari per ser moguts, i no sobrecarreguen la Raspberry amb realimentacions no desitjades.

En cas de voler veure el circuit electrònic final, o el pinout de la Raspberry, es poden consultar les informacions pertinents als **Apèndixs VII** i **VIII**, respectivament, del volum d'**Apèndixs & Annexes**.

[28] Alimentació necessària segons els *datasheets* dels motors.

Capítol 4. - RESULTATS i CONCLUSIONS

En aquest quart capítol es resumeixen els resultats obtinguts en aplicar la metodologia explicada en el capítol 2, i s'extreuen les conclusions finals del treball fet.

4.1. - Resultats

Com a resultats de la realització del projecte, s'han obtingut:

1. Una aplicació d'escriptori utilitzable en qualsevol ordinador (amb SO[29] Windows) prou senzilla i entenedora per l'usuari estàndard, capaç de llegir partitures digitals de bateria (en format .jpg[30]).
2. Una maqueta construïda 100% amb materials reciclats i, per tant, a cost 0, a la qual no se li ha implementat l'opció de poder tocar notes que es realitzarien amb el peu esquerre. Fa 666 x 40 x 30 $[mm]$ de dimensions.

4.2. - Conclusions

Tot i que el prototip funciona més lentament de l'esperat a l'hora de reproduir la melodia enviada des de l'app, i no s'ha implementat un dels tipus de sons característics de l'instrument comercial, es considera que s'ha seguit una bona metodologia de treball i s'ha desenvolupat un bon projecte perquè l'objectiu principal d'aquest ha estat complert, ja que es basava en crear l'aplicació funcional per a l'usuari, i la construcció d'una maqueta de bateria era sols per demostrar l'eficàcia del *software* desenvolupat.

Finalment, comentar que al llarg de tot el projecte han anat apareixent alguns conceptes i característiques que haguessin aportat valor afegit a la programació realitzada, però que no han estat implementades en no ser part de l'abast d'aquest treball, com ara l'automatització del processat d'imatge (sense el requeriment de la configuració dels valors dels paràmetres), o l'addició de la lectura de silencis, i que ara es plantegen com a possibles *next steps* a realitzar en un futur proper per tal de millorar la primera versió de l'aplicació i poder-la penjar a *File Exchange*[31].

[29] *Sistema Operatiu*. Més informació a: https://ca.wikipedia.org/wiki/Sistema_operatiu
[30] *Joint Photographic Experts Group*. Més informació a: https://ca.wikipedia.org/wiki/JPEG
[31] Comunitat de Matlab on la gent penja els seus propis projectes i codis desenvolupats: https://es.mathworks.com/matlabcentral/fileexchange/

Capítol 5. - WEBGRAFIA

En aquest cinquè, i últim, capítol es llisten les fonts utilitzades per realitzar el projecte i citades dins del treball (ordenades alfabèticament) en format APA, de manera que els lectors puguin aprofundir més en matèria si es desitja.

[1] Burchill, J. (2012). *4 Armed Mohawked Robot playing the drums* [Vídeo]. Recuperat de https://www.youtube.com/watch?v=X1jsU1lBZMc&t=1s

[2] Creación de aplicaciones multiventana en el diseñador de aplicaciones. (2017). Recuperat de https://es.mathworks.com/help/matlab/creating_guis/creating-multiwindow-apps-in-app-designer.html

[3] Designing Apps in App Designer. (2016). Recuperat de https://es.mathworks.com/help/matlab/components-in-app-designer.html

[4] CloudMQTT. (2016). *Documentation. Getting started*. Recuperat de https://www.cloudmqtt.com/docs/index.html

[5] CloudMQTT. (2016). *Documentation. Python*. Recuperat de https://www.cloudmqtt.com/docs/python.html

[6] Longruner Servo Motor Micro SG90 9 g RC robot Control para Helicóptero Avión Barco LKY66-UK-10. (2019). Recuperat de https://www.amazon.es/Micro-Helicopter-Airplane-Controls-LKY66-UK-5/dp/B072J59PKZ/ref=sr_1_1_sspa?ie=UTF8&qid=1553175133&sr=8-1-spons&keywords=servo&th=1

[7] MathWorks Internet of Things Team. (2018). *MQTT in MATLAB*. Recuperat de https://es.mathworks.com/matlabcentral/fileexchange/64303-mqtt-in-matlab

[8] MG996R Servo digital de par motor de engranaje con brazo de brazo, SENHAI Servo de robot de 4 paquetes para Futaba Hitec Sanwa GWS JR Helicóptero de RC. (2019). Recuperat de https://www.amazon.es/MG996R-digital-engranaje-paquetes-Helic%C3%B3ptero/dp/B0716V3WNH/ref=sr_1_2_sspa?__mk_es_ES=%C3%85M%C3%85%C5%BD%C3%95%C3%91&keywords=servo+motor&qid=1554031861&s=gateway&sr=8-2-spons&psc=1

[9] Moondra. (2017). *Python Multi-Threading Tutorial for beginners : part-1* [Vídeo]. Recuperat de https://www.youtube.com/watch?v=2ZwuKeL0aHs

[10] Oz Phunkeenstein. (2012). *Animusic - Gyro Drums [HD]* [Vídeo]. Recuperat de https://www.youtube.com/watch?v=W-kOpucwA-Y

[11] Payne, T. (2014). *Let's Learn Python #22 - Multithreading* [Vídeo]. Recuperat de https://www.youtube.com/watch?v=i1SW4q9yUEs

[12] Raspberry Pi Forum. (2017). *Cómo arrancar un programa al iniciar la raspberry* [Fórum]. Recuperat de https://www.raspberrypi.org/forums/viewtopic.php?t=182097

[13] Sentdex. (2014). *Python 3 Programming Tutorial - Threading module* [Vídeo]. Recuperat de https://www.youtube.com/watch?v=NwH0HvMI4EA