Escuela Universitaria de Artes

Licenciatura en Música y Tecnología

Tesis de Grado

“Diseño y desarrollo de controlador MIDI inalámbrico de viento”

Tesista

Juan Mariano Ramos

Director de Tesis

Martín Matus Lerner

Fecha de presentación

21 de Noviembre de 2016

"Estamos hechos de estrellas. Somos un medio para que el cosmos se conozca a sí mismo" – Carl Sagan

Tabla de contenidos

Agradecimientos

Quiero agradecer en primer lugar a mi familia: a mi hermana, Lucía, por ser mi compañera eterna en las buenas y en las malas desde que puedo recordar, y hoy por regalarme tantas horas de su sueño para darme una mano con mis constantes diseños y rediseños de la boquilla. Cuando todo va, viene y se transforma, ella es un faro que siempre puedo ver. A mi viejo, Mariano, por enseñarme lo que es perseverar en una meta, y porque sin todo su afecto y apoyo constante e incondicional no habría podido tomar las riendas y reconstruir mi vida para llegar a donde estoy hoy. A mi vieja, Mariela, por criarnos rebeldes, guerreros, curiosos, por hacer que persigamos nuestras pasiones, y sobre todo, por romperse el lomo durante años para que pudiéramos tener la vida que quería para nosotros. A Viviana por su apoyo, y su afecto de siempre, además de esas charlas enriquecedoras que ocasionalmente solíamos tener. A mis hermanos Fede y Esteban, además de su afecto, por enseñarme cómo es ser personas buenas y totalmente desinteresadas.

A mi tía, Claudia, por su cariño eterno, las charlas y su apoyo incondicional. A mi tío, Guillermo, por ser una inspiración toda mi infancia, ser el tipo de quien siempre provenían novedades apasionantes, y muy especialmente por tirarme una soga en uno de los tiempos más duros que me tocó vivir. A mi abuelo, Jorge, por su dedicación, por su apoyo total con mi música, por estar siempre ahí. A mi abuela Graciela y a Domingo, ojalá pudiesen ver donde estoy parado hoy.

A Antonella por ser mi compañera todo este tiempo. De pronto apareciste en mi vida y me trajiste tiempos maravillosos. Gracias además por tus consejos y asesoramiento “oboístico” y técnico para realizar mi Tesis.

A Sofi, el ser más noble que pisó la Tierra, por acompañarme la mitad de mi vida, toda su vida. Algún día formaré parte de la tierra una vez más y te acompañaré para siempre.

A mis amigos, que son el orgullo y el combustible de mi vida. Todos ustedes me han dado lo mejor siempre, y han soportado incontables “noo, no puedo porque tengo que terminar una cosa para la

facu" de mi parte. Gracias Maxi por bancarme esos meses en tu casa, no lo voy a olvidar nunca. Gracias Carlos por tu amistad, por insistirme en que retome la carrera, en que viaje a Rosario, y por tu ayuda en incontables ocasiones. A todos mis amigos del colegio, de la facu, de la vida, por su apoyo y afecto constantes.

A Gabriel Colautti, por tu carácter jovial y tu locura divertida. Porque te copaste hace casi dos años cuando te hice la propuesta "indecente" de ser tu ayudante, tras lo cual se me vino una catarata de aprendizajes, experiencias, desafíos y por sobre todo, de risas y más risas hasta el día de hoy.

A Esteban Calcagno, por las charlas, los consejos, tu insistencia para que participe en cosas nuevas, tu apoyo y entusiasmo para mis propuestas y, sobre todo, por esa buena onda a prueba de todo.

Quiero agradecer muy especialmente a Martín Matus, mi Director, por confiar en mí y abrirme las puertas desde el principio, desde aquellas charlas en el seminario de Arduino. Por la buena onda, por la paciencia que me tuviste en todo este camino y por bancarte a un rebelde como yo de becario y tesista. Sin tu apoyo, no sólo no habría tenido la oportunidad maravillosa de iniciarme en la investigación, sino que todavía estaría debatiéndome acerca de qué camino elegir para mi Tesis. También por invitarme a participar de FASE y de la JAMTec. Gracias.

Por último, a la Educación Pública, a la Universidad Nacional de Quilmes, y a la Licenciatura en Música y Tecnología, mi lugar en el mundo.

Resumen

El proyecto comprende el diseño y desarrollo de un dispositivo electrónico musical de viento, de tipo controlador MIDI e inalámbrico. Uno de sus objetivos es el de presentar una curva de aprendizaje que permita producir música a cualquier individuo independientemente de sus conocimientos o pericia físico-técnica. Se buscará que, dada la respuesta expresiva, variedad sonora e independencia del uso de cables, posibilite su integración en ensambles de música de todo tipo, sin que su naturaleza electrónica lo condicione a una sonoridad o ámbito específicos.

El desarrollo está basado en un microcontrolador Arduino y la alimentación eléctrica es provista por baterías con un sistema de recarga interno. Se ha escogido el protocolo MIDI para comunicarse con dispositivos sintetizadores de audio diversos, ampliando su versatilidad en cuanto a posibilidades tímbricas y artísticas del instrumento. Mediante el uso de módulos Xbee y la confección de un receptor independiente se ha logrado una comunicación inalámbrica robusta y de baja latencia de datos seriales MIDI.

Se realizaron también diversos estudios previos como un análisis de aerófonos existentes (acústicos y electrónicos) en colaboración con instrumentistas, a fin de determinar la mayor ergonomía y comodidad para el intérprete. También se ha diseñado un sistema de digitación basado en electrodos táctiles para producir las diferentes notas musicales de una escala mayor de base y sus alteraciones cromáticas. Especial rigurosidad se ha puesto en el diseño del sensor de velocidad/presión de aire, la embocadura y la respuesta devuelta al usuario por ésta, requisito fundamental para acortar la distancia perceptual existente en el intérprete entre la acción realizada y el resultado sonoro obtenido. Atento a ello se ha diseñado y confeccionado una boquilla en impresión 3D, la cual incluye además un sensor de presión mandibular para añadir una capa de sensibilidad expresiva.

1. Introducción y justificación

1a. Marco histórico-teórico

El campo de los instrumentos musicales electrónicos lleva más de un siglo de desarrollo y avances, abarcando producciones muy diversas. Se pueden encontrar desde gigantescas instalaciones con dispositivos electromecánicos controlados por teclado como el Telarmonio (Cahill, 1897), pasando por dispositivos como el Theremin (Theremin, 1920) cuya interfaz de ejecución difiere sustancialmente de la de los instrumentos tradicionales, hasta aplicaciones sintetizadoras en pequeños dispositivos móviles hoy en día. Este grupo comprende entonces todo instrumento musical cuyos sonidos sean producidos por medios electrónicos, y excluye habitualmente a aquellos diseñados como instrumentos tradicionales acústicos pero que utilizan amplificación electrónica, como el caso de la guitarra eléctrica y otros. Desde la aparición del protocolo y especificación MIDI[1] en 1983, es cada vez más frecuente encontrar una separación funcional de la parte "controladora" (interfaz física de control por parte del usuario) y la "sintetizadora" (electrónica que produce los sonidos) en estos instrumentos, encontrando dispositivos que presentan una, otra, o ambas, además de medios de interconexión entre ellos. Esto trae nuevas posibilidades a este grupo: un instrumento ya no se encuentra limitado a su conjunto de sonidos incorporados, sino que puede utilizarse como controlador para una infinidad de sintetizadores disponibles, y, a la inversa, un sintetizador puede proveer sonidos a todo tipo y formato de controladores.

Telarmonio (1897)

Si bien existen muchas formas de categorizarlos -muchas veces análogas a las de los instrumentos acústicos- hay una tendencia generalizada a dividirlos en dos grandes grupos: los que presentan similitudes de diseño e interfaz con los instrumentos acústicos tradicionales, y aquellos que evitan

1 "Musical Instrument Digital Interface". Ver http://www.midi.org/techspecs/

en mayor o menor medida esta relación.

Los primeros son generalmente comparados con aquellos instrumentos acústicos a los que semejan[23]. El uso de cables y alimentación eléctrica suelen ser factores de exclusión a la hora de considerar su uso en disposiciones orquestales. Como excepción pueden destacarse algunos instrumentos de teclado utilizados para ensayos y en reemplazo de originales acústicos muy costosos o de difícil transporte. Aun así, a igualdad de condiciones suele preferirse la ejecución con instrumentos acústicos. Sin embargo esto no ha impedido su inserción exitosa en otros ámbitos, un ejemplo de ello es el "arpa láser", la cual musicalmente posee un muy limitado rango de notas y -a priori- dinámica fija encendido-apagado, pero sus características visuales, en conjunto con los timbres sonoros "de fantasía" con que se la suele utilizar la hacen extremadamente popular en los espectáculos.

El segundo grupo corre una suerte diferente ya que se los suele asociar con un tipo de música de índole modernista, en ocasiones "experimental". Las sonoridades de éstos recaen, por lo general, en el campo de los sonidos no afinados, o presentan limitadas capacidades para interpretar música dentro de un lenguaje tradicional. Así es que, en la mayoría de los casos, se utilizan como complementos de otros instrumentos en lugar de ser portadores de información primaria (en un sentido semiótico), o como instrumentos solistas por sus particulares disposiciones visuales, técnicas o estéticas.

---

2 http://the-piano-studio.com/digital-vs-acoustic-piano/

3 https://www.native-instruments.com/forum/threads/why-can-no-synth-rival-a-real-instrument.224683/

1b. Proyecto

Este proyecto contempla el diseño y desarrollo de un dispositivo electrónico musical de viento, inalámbrico y de tipo controlador MIDI.

Se ha puesto gran atención al sensado de la expresividad que el intérprete imprime en un instrumento. El desarrollo fue realizado con herramientas y componentes libres[4] y de fácil acceso toda vez que resultó viable. Se buscaron soluciones para mantener el costo de producción lo más bajo posible, sin que esto vaya en detrimento de la funcionalidad y fiabilidad, atendiendo a que el proyecto pudiera ser eventualmente publicado de manera abierta. Dentro de este espíritu, se sugiere y alienta el estudio y replicación del proyecto por parte de los usuarios y entusiastas. De esta manera el instrumento puede ser producido sin costo -más que el de los componentes físicos- por cualquier individuo en cualquier parte del mundo y aprovechar sus funcionalidades, proponer y aportar mejoras, y aplicar sus propias modificaciones.

Electrónicamente, el instrumento está basado en un microcontrolador Arduino[5]. El mismo es una plataforma electrónica programable de código abierto, basada en un hardware y entorno de desarrollo libres, que facilita el uso de la electrónica en proyectos multidisciplinares. Su facilidad de uso, bajo costo y amplia disponibilidad, permite a los usuarios experimentar y crear desarrollos novedosos no existentes en el mercado, o reemplazar las funcionalidades de productos cerrados, licenciados o de difícil obtención, contribuyendo a la expansión del acceso a la tecnología. La alimentación eléctrica del instrumento es provista por baterías de Ión de Litio con un sistema de recarga incorporado.

Por su sencillez de implementación y robustez, se ha escogido el protocolo MIDI para comunicarse con dispositivos sintetizadores de audio, como computadoras, módulos, teclados y otros, ampliando el abanico de posibilidades tímbricas y artísticas del dispositivo.

---

4 https://es.wikipedia.org/wiki/Cultura_libre

5 http://www.arduino.org/

Se ha diseñado e implementado un sistema de transmisión inalámbrica de información, a fin de evitar el tradicional uso de cables, frecuente en los controladores MIDI existentes, para darle mayor libertad y comodidad al intérprete. Para lograr esto se ha recurrido a dos módulos Xbee (emisor-receptor). En este sentido, ha sido necesaria la creación de un módulo receptor de la señal inalámbrica producida por el instrumento, a fin de transferirla de manera directa a un dispositivo sintetizador MIDI de cualquier tipo. De este modo el instrumento es totalmente independiente del tipo de dispositivo sintetizador con el que opere, liberando a este último de la necesidad de contar con soporte específico para el instrumento. Sin embargo, se ha provisto un conector DIN[6] para el uso de un cable MIDI si el usuario así lo prefiere o requiere por alguna situación en particular.

Respecto al diseño físico del instrumento, se han realizado análisis de algunos aerófonos existentes, acústicos y electrónicos, en colaboración con instrumentistas, a fin de determinar la mayor ergonomía y comodidad para el intérprete. La prioridad ha sido siempre la facilidad de uso, por lo que se ha estudiado también el sistema más conveniente de digitación para producir las diferentes notas musicales de la manera más intuitiva posible y versátil a la vez. Especial rigurosidad se ha puesto en el diseño del sensor de velocidad de aire y la embocadura.

El desarrollo ha girado principalmente en torno a estos puntos:

- La embocadura y el sensor de viento.
- La interfaz digital de elección de notas.
- El sistema inalámbrico de transmisión y recepción de datos.
- La articulación de estos aspectos con el diseño y construcción del cuerpo físico del controlador.

6 https://es.wikipedia.org/wiki/Conector_DIN

2. Estado actual del conocimiento sobre el tema

Existe actualmente una enorme cantidad de aerófonos electrónicos, en su mayoría controladores MIDI, con unas pocas excepciones -como el EWI4000s de Akai- que incorporan un sintetizador interno. A fin de analizarlos brevemente, nos enfocaremos en aquellos que utilizan el protocolo MIDI y los dividiremos en dos categorías: comerciales de producción masiva y de tipo DIY[7].

2a. Instrumentos comerciales

Dentro de los comerciales podemos destacar algunos modelos representativos:

Yamaha WX5[8]: es un controlador de viento con un diseño similar al de un saxofón soprano. Posee teclas en forma de llaves y dos boquillas intercambiables que simulan una de lengüeta simple y un pico de flauta dulce. Tiene un sensor de presión labial para controlar algunos parámetros en la emisión del sonido. Brinda la opción de elegir el tipo de digitación. Diseñado principalmente para músicos experimentados.

Yamaha WX5

Akai EWI4000s[9]: posee una apariencia similar a un clarinete, pero diseñado para facilitar su uso por parte de instrumentistas de vientos de metal. Es uno de los pocos que poseen módulo sintetizador interno. Hace uso de una interfaz de llaves táctiles con distintas opciones de digitación.

Akai EWI4000s

7 "Do It Yourself", "Hágalo Usted Mismo".

8 http://usa.yamaha.com/products/music-production/midi-controllers/wx5/

9 http://www.akaipro.com/product/ewi4000s

Eigenlabs Eigenharp[10]: no es específicamente un controlador de viento, sino un instrumento multientrada, con una matriz de elementos táctiles además de una boquilla al estilo de un fagot. Está diseñado para expandir las posibilidades instrumentales de los músicos avanzados y expertos.

Eigenlabs Eigenharp

Casio DH-100[11]: este instrumento fue producido desde mediados de la década de 1980, e introducido inicialmente en el mercado como un juguete. Pese a ello, sus características particulares lo convirtieron en un dispositivo muy utilizado por músicos entusiastas. Incluye un módulo sintetizador de 6 instrumentos (saxofón, trompeta, caña-sintetizada, oboe, clarinete y flauta) y conexión MIDI OUT. Una de sus cualidades más notables es la inclusión de un amplificador con altavoz (situado en su campana), de manera que, a diferencia del resto de los controladores mencionados, puede producir sonidos directamente.

Casio DH-100

Ninguno de estos dispositivos presenta algún tipo de comunicación inalámbrica.

10 http://www.eigenlabs.com/

11 https://en.wikipedia.org/wiki/Zanzithophone

2b. Proyectos DIY

Dentro de los DIY nos encontramos con todo tipo de desarrollos y formatos. Pocos de ellos superan la fase experimental y por lo general ni siquiera son montados en carcasas utilizables, limitándose a demostraciones con circuitos electrónicos totalmente expuestos, aunque de alto valor teórico.

Con la irrupción del Arduino en el campo DIY, la vasta mayoría de proyectos en la materia se ha volcado hacia esta tecnología, pues brinda a los usuarios una muy rápida y eficiente solución para los problemas de control electrónico y digital, permitiéndoles enfocar sus esfuerzos en el diseño y exploración de las funcionalidades. Se pueden encontrar en comunidades web como Instructables[12] o Make[13] una gran cantidad de proyectos en desarrollo. Ocarduina fue uno de ellos, que consistía en una ocarina controlada por Arduino, accionada por medio de sensores capacitivos en el lugar de los agujeros de una ocarina tradicional. Lamentablemente este proyecto fue cerrado, pero permanecen registros de algunas de sus implementaciones en internet[14].

Algunos de los proyectos más notables en esta categoría son el OHMs “Open Horn MIDI system”, el Gordophone y el MiniWI, estos dos últimos basados en el mencionado Akai EWI. Presentan algunas diferencias de implementación y sofisticación de sus interfaces.

12 http://www.instructables.com/

13 http://www.makezine.com/

14 Demostración de proyecto basado en Ocarduina https://www.youtube.com/watch?v=5MT-nALUGkM

Open Horn MIDI System[15]: creado por Kontinuum, este es uno de los proyectos más completos y sofisticados que se han estudiado. Posee una carcasa totalmente impresa en 3D[16], una pantalla LCD de 16x2 caracteres y un sistema de llaves basado en el saxofón. Utiliza un sensor basado en un globo de látex que al hincharse mediante el soplo, excita un FSR[17]. Esta solución es de bajo costo, sin embargo no ofrece una salida de aire y el usuario no puede vaciar sus pulmones, además al tratarse de un sistema mecánico está sujeto a numerosas posibilidades de interferencia climática y fallos por rotura o desgaste. Gracias a la mencionada inclusión de una pantalla LCD, ofrece numerosas posibilidades de configuración y personalización.

OHMs

Gordophone [18]: este proyecto creado por Gordon Good es el más exhaustivamente documentado y probablemente con mayor sofisticación en cuanto a la programación orientada a sensores de presión de aire. En su blog, el autor describe paso a paso todas las etapas de su desarrollo, incluyendo ideas que ha descartado en favor de otras más eficientes. Su instrumento final utiliza una interfaz táctil para la digitación. Sin embargo, no se ha hecho un énfasis en la ergonomía ni estética, no posee opciones de configuración, sus "llaves" constan de sencillas arandelas metálicas, y, al igual que el OHMs, no utiliza una salida de aire.

Gordophone

[15] http://kontinuumlab.blogspot.com.ar/2015/01/introducing-ohms-open-horn-midi-system.html

[16] La impresión 3D es una tecnología de fabricación por adición donde un objeto tridimensional es creado mediante la superposición de capas sucesivas de material, generalmente por extrusión en caliente.

[17] Sensor de Fuerza Resistivo.

[18] http://gordophone.blogspot.com.ar/2013/01/building-breath-controller.html

MiniWI[19]: este proyecto está basado íntegramente en un Akai EWI4000s y fue creado por Johan Berglund. El autor basó su sistema de digitación en el mencionado dispositivo de Akai, ya que posee uno y, según argumenta, no deseaba aprender un nuevo sistema. Para ello utiliza una interfaz táctil con electrodos realizados en cinta conductiva de cobre. Este instrumento posee una mayor sofisticación en cuanto a su estética, presenta una carcasa realizada en madera, lo cual también facilita el montaje interno de los componentes. Tampoco presenta una salida de aire.

MiniWI

Sylphyo[20]: Un proyecto destacable es el Sylphyo de Aodyo, que propone una de las pocas interfaces inalámbricas de conexionado MIDI en existencia. Sin embargo Sylphyo tiene proyección comercial, por lo que su diseño es cerrado y posee una patente sobre el sensor de viento. Se encontró hasta fines de 2015 en fase de desarrollo y buscando financiación de tipo crowdfunding[21], la cual finalizó exitosamente y permitió a los desarrolladores distribuir los primeros instrumentos al público, pero a un costo incluso mayor que las alternativas comerciales preexistentes.

Sylphyo

Existen numerosas dificultades técnicas a la hora de diseñar y realizar un instrumento electrónico de viento. Principalmente en el sensado de la velocidad del aire, del tipo de ataque ejecutado, de las variaciones impuestas al sonido luego de que este se ha puesto en marcha, de la respuesta al cambio de notas ligadas y no ligadas, del tipo de interfaz que accionarán las manos del usuario, del método escogido para la digitación de las diferentes notas (los aerófonos no tienen un sistema uniforme, sino que varía según la familia y el tipo de instrumento), y otros.

---

19 https://hackaday.io/project/11843-miniwi-woodwind-midi-controller

20 https://www.aodyo.com

21 Cooperación colectiva llevada a cabo por personas que realizan una red para conseguir dinero u otros recursos.

3. Desarrollo del proyecto

Tras un análisis inicial de la problemática planteada, de otras iniciativas relacionadas y estudiar los medios técnicos a disposición, se resolvió subdividir la investigación y desarrollo en áreas discretas a trabajar de forma modular. Se trabajó en base a diversas hipótesis para resolver las problemáticas -principalmente técnicas y de montaje- presentadas, documentando cada paso realizado mediante notas, diagramas manuscritos y registros fotográficos. Asimismo se han realizado diversos estudios complementarios para asistir al diseño de algunos puntos del proyecto. Anexos al final de la presente Tesis pueden encontrarse fotografías descriptivas de la manufactura, una tabla de implementación MIDI y el código fuente en su totalidad. Se exponen a continuación los procesos y sus resultados organizados por área.

Instrumento y receptor inalámbrico

3a. Sensado de velocidad/presión de aire

Dado que el proyecto plantea la creación de un aerófono, es necesario contar con un método fiable de medición de la velocidad del flujo de aire, a fin de proveer al microcontrolador Arduino con valores proporcionales a la acción del intérprete y que estos puedan ser procesados para generar la información (notas) MIDI consecuente. Se han establecido ciertas condiciones básicas que dicho método debe satisfacer respecto de las expresiones aplicadas por el usuario:

- Proporcionalidad directa entre la velocidad del flujo de aire y el valor medido
- Ausencia o minimización de componentes inerciales en el sistema (es decir, que al cesar el estímulo exista el menor retraso posible hasta que el sistema se estabilice)
- Demora de respuesta estímulo-lectura mínima (idealmente inferior a 1ms)
- Rango de medición acorde al rango de presiones producibles por el aparato respiratorio de un ser humano en condiciones normales (aproximadamente de 0 a 15 o hasta 20 kPa)
- Resolución de al menos 128 pasos entre las lecturas mínima y máxima (ya que el protocolo MIDI trabaja con dicho rango en sus mensajes CC[22] de expresión dinámica)
- Bajo nivel de ruido o sensibilidad a interferencias externas a la acción del usuario
- Permitir en su implementación la evacuación del aire pulmonar para otorgar una sensación de soplo al usuario

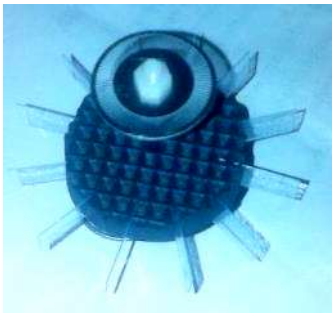
Algunas de estas condiciones surgen como conclusiones tras los experimentos realizados, otras (como la resolución y el rango de presiones) responden a las propiedades del sistema planteado.
Los métodos que se exploraron son:

- Sistemas rotatorios de hélice con dínamo
- Sistemas rotatorios de hélice con codificadores ranurados
- Obturadores ópticos con solapa móvil tipo bisagra
- Obturadores magnéticos (sensor de efecto hall) con solapa móvil tipo bisagra

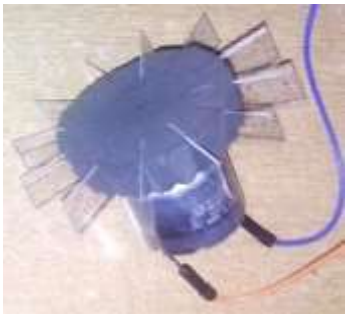
[22] *Continuous controller* (controlador continuo).

- Sensores piezorresistivos (Freescale MPX5050 y MPX5010)

Obturador óptico



Codificador ranurado



Dínamo con hélice

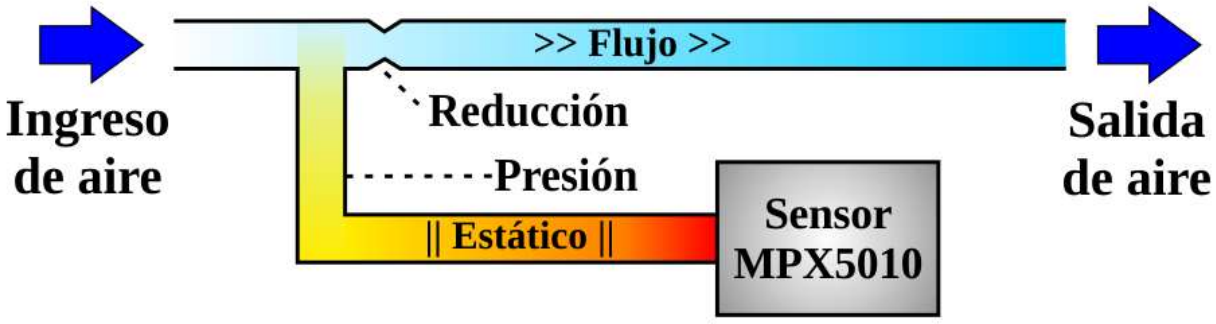
Todo método que implique una pieza mecánicamente móvil -ya sea una hélice, solapa u otros- presenta una inercia intrínseca proporcional a la masa de dicha pieza, produciendo diversos grados de retrasos perceptibles en la lectura. En el caso de las solapas obturadoras existe el problema adicional de la interferencia de la gravedad según la inclinación del sensor. Ésto intentó compensarse mediante el uso de muelles estabilizadores que cerraran la solapa en ausencia de soplo, pero la tensión mínima requerida para vencer el efecto de la gravedad por completo fue excesiva en relación a la fuerza producible por el flujo de aire, tornando muy inestable al sistema. Otro factor de incidencia crítico fue la condensación de la humedad natural contenida en la exhalación sobre las diversas piezas móviles, afectando su movimiento de manera impredecible. Ésto produce inestabilidad en todos los sistemas mecánicos y altera la proporcionalidad esperada en la lectura.

Tras estos experimentos se determinó que solo los sensores de presión piezorresistivos reúnen las condiciones necesarias. Éstos poseen un componente de cristal piezoeléctrico que varía su conductividad en proporción a la presión aplicada sobre él, de este modo no utilizan piezas móviles y devuelven un valor muy preciso y estable. Se diseñó entonces un sistema similar al “tubo de Pitot” utilizado en aeronáutica para medir la velocidad de los vehículos aéreos, que se basa en el efecto Venturi. Éste predice que un fluido -el aire- en movimiento dentro de un conducto *disminuye su presión cuando aumenta la velocidad al pasar por una zona de sección menor*. Así, midiendo la presión en puntos específicos del conducto puede calcularse la velocidad del flujo. A efectos de su

MPX5010

utilización en el control de un aerófono electrónico es irrelevante conocer la velocidad o presión absolutas aplicadas, sino que es preciso obtener una magnitud relativa a la fuerza de la exhalación que permita el procesamiento y calibraciones en el procesamiento de manera proporcional.

El sistema diseñado además permite la evacuación del flujo por una vía separada de la del sensor: según indica el principio de Bernoulli para fluidos, *la suma de energías potencial y cinética en todos los puntos del mismo es constante*. Por esta razón, la energía aplicada por el usuario es efectivamente dividida -por la morfología de los conductos- y manifestada en dos formas: por un lado *presión* (potencial), que es registrada y medida por el sensor, y por otro *flujo* (cinética), el cual otorga la sensación de soplo y además expulsa el contenido de humedad pulmonar hacia el exterior del instrumento sin comprometer su funcionamiento y manteniendo al sensor libre de posibles obstrucciones y malfuncionamientos causados por la condensación.



Esquema del sistema utilizado para el sensado del flujo de aire

Teniendo en cuenta que un humano es capaz de producir una presión al exhalar de alrededor de 15-20 kPa como máximo, se ha escogido el sensor MPX5010 de Freescale, el cual trabaja en el rango de 0-10 kPa y se alimenta con 5 v, lo que lo convierte en idóneo para la plataforma Arduino. Este sensor devuelve un valor analógico (continuo) de entre aproximadamente 0,3 y 4,7 v proporcional a todo el rango de medición, el cual es recogido por una de las entradas analógicas del microcontrolador y convertido a un valor digital con resolución de 10 bits (0 a 1023). Debe considerarse que, como se ha mencionado, en un tubo de Pitot la presión aplicada sobre el conducto de medición no corresponde a la energía total contenida en el flujo, ya que al ser un conducto abierto una parte es expresada como presión y el resto como movimiento. Esto no afecta la

proporcionalidad de la medición, pero su consecuencia es que el sensor no será excitado por la totalidad de los 15-20 kPa presumiblemente aplicables por el usuario. Por ello el rango de trabajo del mismo debe tener en cuenta esta discrepancia, y se ha escogido un límite máximo de 10 kPa para optimizar su respuesta.

Pieza "T"

De esta manera, en primera instancia se previó la introducción de aire por medio de una embocadura conectada a una doble tubería de PVC flexible (estándar industrial de 6 mm) mediante una pieza de unión en "T". Sin embargo tras los progresos en el diseño de la boquilla (ver sección "Boquilla"), se determinó que es posible prescindir de la pieza "T" y realizar la bifurcación de los conductos directamente en aquella. El conducto de evacuación se dispuso de manera que tenga salida en la parte inferior del instrumento y, en condiciones normales, el agua condensada sea expulsada hacia el suelo.

Tubo de PVC 6 mm

3b. Boquilla

Este elemento ha sido considerado de vital importancia desde el comienzo de la investigación, y debido a ello ha sufrido sucesivas modificaciones en su diseño y se han producido algunos prototipos funcionales. Tras la experiencia adquirida en las correspondientes puestas a prueba, se ha llegado a un resultado satisfactorio y funcional adecuado a los objetivos planteados. Sin embargo considerando que podían realizarse mejoras sustanciales en cuanto a ergonomía, montaje y estética recurriendo a la tecnología de impresión 3D, se procedió a realizar un modelo de la boquilla que optimizara sustancialmente las posibilidades existentes. Para esto último se ha contado con el asesoramiento de la Diseñadora Industrial Lucía Ramos (UBA), en aspectos técnicos respecto de la elección de materiales y modelado en 3D.

Inicialmente se realizó un análisis de la morfología de las embocaduras de diversos aerófonos convencionales, sus técnicas de utilización y tiempos de aprendizaje, contrastando las primeras observaciones con las experiencias de algunos instrumentistas consultados a tal fin. Para esto se

Señales de tensión en músculos faciales

Menores señales de tensión labial

realizó también una encuesta consistente en 4 preguntas abiertas orientadas a conocer las problemáticas de las diversas embocaduras según los usuarios (ver anexo II). Tras estudiar los datos recopilados se concluyó que, en la mayoría de los casos, el diseño de tales embocaduras está condicionado por el método de producción física del sonido. Como es natural, debe garantizarse primero el funcionamiento correcto del medio productor de sonido, y solo entonces existe una cierta libertad de diseño para favorecer la potencial comodidad del intérprete. Posteriormente se realizó un estudio de las posiciones y formas naturales de apertura labial durante el soplo utilizando el menor esfuerzo posible en los músculos faciales. Éste muestra que la morfología general es similar a una elipse, cuyas dimensiones varían según la persona, pero pueden enmarcarse en un promedio de 18 x 8 mm. Al confrontar esta información con el análisis previo realizado, se puede inferir una relación directa entre el estrés muscular facial y el desviamiento respecto de esta forma cuasi elíptica hallada según imponen las diversas morfologías de embocaduras.

Bocas en posición de soplo

Debe mencionarse algunas observaciones adicionales que surgen de las consultas a instrumentistas (anexo II). En las familias de cañas (clarinete, oboe, fagot, el saxofón en menor medida, y otros), ya sean dobles o simples, el ejecutante debe adquirir una gran pericia inicial, y utilizar posiciones muy específicas de la boca y labios. En los metales (como trompeta, trombón, corno francés, tuba y otros) el caso es similar, añadiendo a ello el estrés epidérmico producido por la permanente tensión y vibración de los labios. Se requiere destreza y una presión mandibular precisa y estable para producir un sonido claro y afinado. La suma de estas presiones y tensiones musculares de diversa índole demanda un mínimo de estado y resistencia físicos, y aún en músicos experimentados esto lleva ineludiblemente a la aparición de cansancio transcurrido cierto tiempo de ejecución. Incluso existe el riesgo tangible durante los primeros estadios del aprendizaje de generar vicios o lesiones

físicas por la incorrecta aplicación de las técnicas.

Según la encuesta y estudios realizados, la apreciación general es que el saxofón y la flauta dulce, entre otros, presentan las boquillas más cómodas y fáciles de utilizar, al menos para un primer acercamiento a la cuestión. Sin embargo, los instrumentos que presentan condiciones aún más favorables en cuanto a los factores mencionados son aquellos en que el medio productor de sonido no se encuentra alojado en la propia boquilla, sino que ésta actúa como mero acoplador entre la boca del ejecutante y un conducto de aire dirigido hacia la parte activa del sistema. De esta manera su diseño queda liberado de la función de soporte o alojamiento, y suele presentar morfologías más cómodas y sencillas de utilizar. El caso arquetípico es de la flauta melódica de teclado.

Otro aspecto encontrado en casi la totalidad de los aerófonos es la imposibilidad de mover de forma independiente la embocadura y el cuerpo principal del instrumento (la mencionada melódica con manguera es una notable excepción). Dado que el intérprete precisa realizar numerosos movimientos corporales tanto para acceder a las distintas posiciones de digitación como para asistir mediante gestos a su cualidad expresiva, contar con dicha posibilidad le brindaría un grado mayor de comodidad y libertad.

Atento a estas circunstancias, y tras el análisis de las posiciones de los labios y la boca en posiciones de soplo antes mencionado, se produjo el diseño tentativo de la primera boquilla rudimentaria (Prototipo 1), y de inmediato surgieron dos nuevas conclusiones vinculadas. La forma en bisel utilizada en sus extremos resultaba cómoda para el soplo, pero presentaba una tendencia a deslizarse “escapando” de los labios cuando se pretendía utilizar estos como punto de anclaje para regular la posición e inclinación del instrumento en general. Este problema está presente también en muchas flautas dulces, motivo por el cual ejecutantes inexpertos tienden a tomar con los dientes su boquilla para sostenerla en posición, algo que debe ser evitado.

Prototipo 1

Se determinó así que el diseño no debía presentar un bisel en la zona de apoyo de los labios. Por otro lado, este nuevo criterio permitiría eliminar a tal punto la necesidad de utilizar la fuerza mandibular que, además de lograr la comodidad del intérprete, abrió el camino para añadir otra dimensión al sensado de la expresión. La fuerza mandibular puede ser ahora aprovechada para excitar *a voluntad* -y ya no de manera obligada- un nuevo sensor colocado en la superficie de la boquilla. Este punto es tratado en la sección "Otros sensores de expresión". Tras ello se diseñó y construyó otra boquilla (Prototipo 2), de sección oval sin biseles según lo mencionado. Esta probó ser muy efectiva para los fines esperados, y además proporcionó un espacio adecuado para alojar el sensor de fuerza resistivo capaz de medir la presión mandibular ejercida y generar así un nuevo parámetro de control expresivo.

Prototipo 2

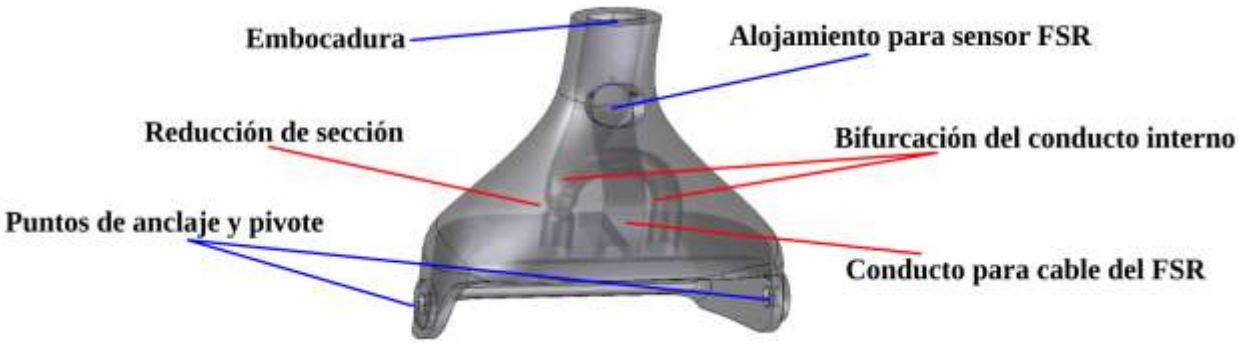
Probado el correcto funcionamiento y aptitud de la boquilla Prototipo 2, se determinó que a efectos de mejorar su montaje al cuerpo del instrumento y de optimizar sus cualidades materiales, ergonómicas y estéticas, resultaría conveniente que la misma fuera modelada específicamente y manufacturada por medio de la impresión en 3D. Este es actualmente uno de los métodos más utilizados en la industria para la generación de prototipos funcionales y posibilitar su evaluación previa a la instancia de fabricación en serie, e incluso para la creación de numerosas piezas mecánicas y artísticas únicas. Se considera que de esta manera se puede ofrecer la mejor experiencia al usuario del instrumento, en concordancia con los objetivos planteados para el proyecto. En este sentido, y como se ha mencionado antes, surgió la posibilidad de incorporar la función de bifurcación previamente asignada al conector en T a la propia boquilla, pudiendo mejorarse notablemente el control de este aspecto.

Para el diseño conceptual de esta pieza se buscó entonces contar con las siguientes características:

- Un extremo de introducción de aire acorde al diseño determinado con anterioridad
- Un medio de anclaje al cuerpo del instrumento que al mismo tiempo permita cierto grado de

independencia de movimientos

- Un sitio de alojamiento para el sensor de fuerza resistivo, considerando que este debe encontrarse protegido (aislado) de la saliva y humedad de la boca, pero manteniendo su susceptibilidad a la presión ejercida sobre él
- Una bifurcación interna del flujo de aire en dos conductos, de manera que pueda omitirse esta función en el cuerpo del instrumento
- Un diseño de tales conductos que favorezca el aprovechamiento del efecto Venturi-Bernoulli y un correcto estímulo del sensor MPX5010
- Una orientación de los mismos que ayude -en algún grado- a evitar la introducción de partículas de agua y otras en el conducto dirigido al sensor MPX5010
- Terminales de conexión para los tubos de 6 mm (sensor y evacuación), teniendo en cuenta que estos no deben limitar el movimiento de la pieza
- Una terminal de conexión eléctrica para el sensor de fuerza resistivo
- Ergonomía y continuidad morfológica-estética



Modelo 3D de la boquilla (con vista interna)

La nueva boquilla pudo ser producida exitosamente en PLA[23]. Este material es un termoplástico biodegradable obtenido a partir de la caña de azúcar (o almidón de maíz según la zona). Debido a que el proceso de impresión 3D utiliza filamentos, fue necesario el lijado y emparejamiento de la superficie para eliminar escalones y defectos. Algunos sectores debieron ser rellenados o sellados

[23] Ácido poliláctico.

con cianoacrilato.

Boquilla impresa en PLA (proceso de emparejamiento y suavizado)

El sensor FSR 400 fue montado en el alojamiento diseñado a tal fin, y posteriormente recubierto con sellador siliconado negro del tipo utilizado en plomería. Si bien este material es poco resistente al desgarramiento, debido a la cavidad mencionada resulta suficientemente protegido contra daños accidentales. En la parte trasera de la pieza se ha incorporado un terminal de dos pines macho para conectar el cable hacia la placa base y el Arduino.

Se han colocado dos tornillos laterales en los orificios de montaje y de diámetro ligeramente inferior a estos últimos, de manera que, al fijarse en la carcasa del instrumento, actúen como eje de pivote. Esto proporciona un cierto grado de movimiento independiente de la boquilla respecto al cuerpo del instrumento. Por esta razón, los orificios de conexión para los tubos de PVC fueron alineados de manera paralela al eje de rotación de la pieza, reduciendo de esta manera la resistencia a la flexión opuesta por dichos tubos.

3c. Interfaz de digitación

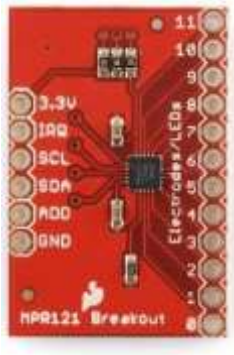
Si bien en un principio se contempló la utilización de diversos interruptores electromecánicos para hacer las veces de "llaves" como poseen muchos aerófonos acústicos, esto presentaba algunos problemas como diseñar un sistema mecánico de acción compatible con la posición de las manos y dedos del usuario, además de requerir puertos en exceso del Arduino.

Durante las primeras etapas de la investigación se ensayaron distintas hipótesis:

- Una botonera electrónica de 8 interruptores sin retención, la cual fue utilizada como medio elemental de digitación a la hora de diseñar otros aspectos de la programación hasta tanto se llegase a una solución más definitiva. Tal sistema tiene la desventaja de demandar un puerto del microcontrolador por cada interruptor que se incluya, lo cual imposibilitaba utilizar un Arduino UNO o Nano (basados en Atmega 328), por lo que se recurrió temporalmente a un Arduino Mega, el cual dispone de una mayor cantidad de puertos, para cubrir esta eventualidad. Sin embargo las dimensiones de este último son relativamente excesivas, lo que habría dificultado su incorporación en un formato reducido de instrumento musical.
- Un sistema basado en un divisor resistivo múltiple, implementando una serie de resistencias cuyos puntos medios fueran contactados por los interruptores de la botonera. De esta manera, al aplicar una tensión de 5 v entre los extremos de la serie, cada interruptor interceptaría un punto específico de entre 0 y 5 v, el cual sería leído por un único puerto analógico del Arduino. Sin embargo tras los experimentos correspondientes se descartó este método debido a las inestabilidades eléctricas que se producían hacia los niveles más bajos de tensión, lo cual no garantizaba una lectura firme ante determinadas digitaciones.
- Alternativamente se utilizó una serie de diodos 1N4007 del mismo modo que las resistencias, a fin de aprovechar su caída de tensión en directa de 0,7 v aproximadamente, pero esta propiedad resultó ser altamente inestable ante mínimos cambios de temperatura.

Atendiendo a estas problemáticas se ha exploró la alternativa de una interfaz táctil capacitiva, lo cual resultó ser el medio más adecuado para los objetivos del proyecto. De entre las opciones disponibles se ha escogido y utilizado la interfaz MPR121 desarrollada por Freescale. Esta provee

una solución integrada para la utilización de hasta 12 electrodos táctiles, y demanda un mínimo de puertos ya que hace uso del protocolo serie I^2C[24] para comunicarse con microcontroladores. Este tipo de sistema requiere de una superficie conductiva que, al ser tocada por el usuario, produce una drástica variación de capacitancia que es medida e interpretada por el sensor como un evento de "toque". Tras las primeras pruebas con electrodos provisorios el sistema probó ser suficientemente robusto y responsivo como para satisfacer las demandas del proyecto. Sin embargo se requirieron numerosos ajustes de programación en el módulo hasta lograr una respuesta libre de interferencias, estos son discutidos en la sección "Programación". Del mismo modo, debido a que el módulo se alimenta de 3,3 v, se hizo imprescindible incluir en el instrumento una fuente estable de dicha tensión, lo cual es expuesto en la sección "Alimentación eléctrica".



MPR121

Una vez superada la etapa de pruebas al MPR121, se procedió a diseñar una serie de electrodos (en lo sucesivo y para facilitar la exposición serán llamados "llaves") de aluminio pulido "a espejo". Esto último además de aportar una mejor estética, optimiza el contacto con la piel del usuario. En un principio se contempló disponer de las llaves en dos grupos de 4 siguiendo el contorno de ambas manos, pero finalmente se optó por hacerlo de manera simétrica, es decir centradas a lo largo del instrumento y con anclajes por medio de tornillos en el centro de cada una. Esto brinda la opción al usuario de utilizar las manos como le resulte más conveniente, en lugar de estar forzado a colocar la mano izquierda en la sección superior y la derecha en la inferior, como es el caso de muchos instrumentos tradicionales. Otros aspectos del instrumento también fueron diseñados bajo este criterio de simetría en la utilización de las manos.

Llaves

[24] I^2C es un bus serie de datos utilizado para la comunicación entre diferentes partes de un circuito, por ejemplo, entre un controlador y circuitos periféricos integrados (como sensores y otros).

En lo respectivo a la elección de notas, se compararon los sistemas de digitación de algunos aerófonos, especialmente el saxofón, el oboe, la flauta y la trompeta. A partir de ello y dado que la amplia mayoría de la música está basada en escalas diatónicas, se resolvió utilizar un diseño con 8 notas de base, dispuestas en una escala mayor más la octava de la primera. Para esto se utilizaron 7 llaves (la ausencia de toque en todas ellas es considerada una 8va posición) y determinadas combinaciones para producir las restantes notas de la octava. Éstas pueden ser digitadas de manera aditiva (a la manera los aerófonos tradicionales) o bien tocando solo la llave que representa la nota elegida.

Posiciones naturales de los dedos sobre la carcasa

A partir de allí, existe una 8va llave (que en adelante se llamará "#/b") que puede configurarse desde el panel de control (ver sección "Interfaz de configuración") para subir o bajar un semitono a cualquier nota, de modo que facilite la adecuación a distintas tonalidades o alteraciones armónicas. Esta es una posibilidad que no existe en los aerófonos de madera, y tiene su base en un aspecto del funcionamiento de los de metal. Estos instrumentos fundamentan su digitación en la generación de un parcial de la serie armónica mediante la vibración de los labios, y las digitaciones (o movimiento de la vara en el caso del trombón) permiten alargar o acortar efectivamente el largo del tubo resonante, alterando en dicha proporción la nota producida por el intérprete. Este es un modo relativamente complejo de utilización para un ejecutante sin tales conocimientos teóricos. Sin embargo se ha tomado la idea aislada de "*alterar la nota que esté sonando*" como un modo sencillo de acceder a las notas restantes de la octava, además de resultar convenientemente análogo al sistema de notación musical occidental basado en el pentagrama moderno.

Portavoces

Sumado a esto se han dispuesto dos llaves en la zona de acción del pulgar de la mano superior para hacer las veces de portavoces, permitiendo en sus diferentes combinaciones utilizar hasta 4 octavas con digitación constante en todas ellas. Este último punto es particularmente deseado en vistas de la facilidad de ejecución, ya que, por lo general, los aerófonos tradicionales presentan numerosas diferencias de digitación en las distintas octavas de su rango debido a distintos condicionantes físicos y morfológicos. La forma de los portavoces es ligeramente triangular, ha sido determinado así tras estudiar el movimiento rotatorio del pulgar en esa zona, y evita toques involuntarios de los portavoces, además de ser simétricos lateralmente. Por último y para facilitar de manera integral la interpretación, se ha incorporado la opción de transposición en el panel de control, que permite desplazar en una cantidad a elección de semitonos tanto hacia arriba como hacia abajo la escala diatónica de base.

Para el montaje y el conexionado eléctrico de las llaves se diseñó un sistema basado en tornillos pasantes a través de la carcasa, que anclan en tuercas fijadas a una pequeña pieza de madera en forma de plancha. A estas tuercas se les han soldado cables conductores que son guiados por calados en la madera hasta finalizar en un terminal de 8 pines apto para la interconexión con el MPR121 (un sistema similar pero de solo 2 conductores es utilizado para los portavoces). Este método de tornillos y soldadura en las tuercas establece un contacto eléctrico robusto y de baja resistencia entre los electrodos y el MPR121, permite su desmontaje sin riesgo de dañar las conexiones, y además posibilita rotar fácilmente en diversos ángulos cada una de las llaves para favorecer el alcance de los dedos del usuario. Por otra parte, como se explica en la sección “Carcasa y montaje interno”, esta pieza de madera sirve internamente de anclaje para otros componentes del instrumento.

3d. Otros sensores de expresión

FSR 402

Además de la expresión dinámica controlada por el flujo de aire se ha dispuesto añadir dos opciones adicionales para el control de parámetros expresivos por parte del usuario. Ambas constan de sensores de fuerza resistivos (FSR 400 y 402), los cuales son capaces de disminuir su resistencia eléctrica según la presión mecánica ejercida sobre ellos. En estado de reposo, presentan una resistencia de varios MΩ, considerada "infinita" a efectos prácticos (ver tabla y gráfico). En este proyecto son utilizados como divisores resistivos en conjunto con una resistencia "pull down" de 10 KΩ conectada a 0 v y un capacitor electrolítico de 1 uF en paralelo con ella como filtro para estabilizar su valor y suavizar el recorrido. Los experimentos con capacitores de mayor valor (10 uF) mostraron un mejor filtrado de ruido, sin embargo introducían un retardo considerable en la señal (en el rango de 1 segundo durante la descarga), al punto de tornar impráctica su utilización. Por esta razón se ha definido 1 uF como valor final. La diferencia entre ambos sensores radica en su implementación y método de excitación.

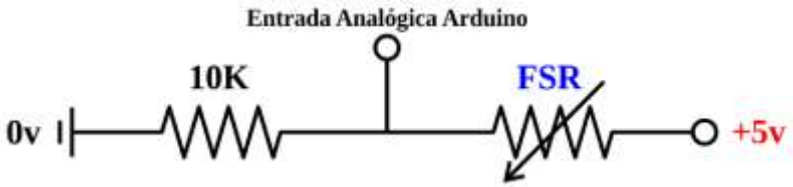
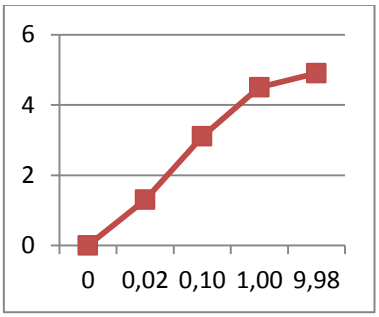


Diagrama de conexión genérico de un FSR

Tensión utilizando una resistencia R "pull down" de 10 KΩ

Fuerza (Kg)	Fuerza (N)	FSR (KΩ)	FSR + R (KΩ)	Corriente FSR+R (mA)	Tensión en R (v)
0	0	Infinita	Infinita	0	0
0,02	0,2	30	40	0,13	1,3
0,10	1	6	16	0,31	3,1
1,00	10	1	11	0,45	4,5
9,98	100	0,25	10,25	0,49	4,9

Tensión (v) vs. Fuerza (Kg)



El sensor FSR 402 llamado "A", posee 18 mm de diámetro y se ha dispuesto en la zona de acción del pulgar de la mano inferior, bajo la pieza de soporte (ver sección "Carcasa y montaje interno") de manera que el usuario pueda desplazarse sobre aquél y presionarlo para producir diferentes intensidades de variación a parámetros MIDI de control expresivo. Luego de las primeras pruebas se hizo necesario recurrir a un material intermediario que permitiera distribuir uniformemente la

fuerza aplicada sobre el sensor y brindar una respuesta sensorial más adecuada para el usuario. Se utilizaron diversos tipos de plásticos blandos hasta dar con una goma densa que cumplía con los requisitos esperados, y se moldeó la misma para que pudiera ser colocada en la pieza de soporte. Adicionalmente, se colocó un plástico espaciador de 0,3 mm de espesor, el cual garantiza tal distancia entre la goma y el sensor de modo que este no pueda ser accionado involuntariamente, por roces o vibraciones.

Soporte y goma de "A"

El segundo sensor, "B" (FSR 400), que posee 7,6 mm de diámetro (y ha sido introducido en la sección "Boquilla"), se ha colocado en la zona de acción del labio inferior del usuario para ser accionado por la presión mecánica ejercida por la boca. Inicialmente fue colocado para ser accionado por el labio superior, pero resultó más cómodo a los usuarios consultados al situarlo en la cara inferior. Se encuentra 12 mm desplazado hacia el instrumento de modo que no se vea afectado durante el uso normal de la boquilla. En la boquilla prototipo 2 se le había añadido un pequeño disco de goma para favorecer la distribución de la presión, del mismo modo que en "A". Al trasladarlo a la boquilla impresa en 3D se ha recubierto con sellador siliconado negro para realizar esta función y otorgar una terminación suave, además de proteger al sensor de la posible humedad proveniente de la boca del usuario y otros agentes que puedan dañarlo o perjudicar su funcionamiento a nivel eléctrico. La introducción de este sensor se ha contemplado para proveer una vía de expresión análoga a la que presentan los instrumentos de lengüeta principalmente, en los cuales pueden realizarse cambios de altura y otros efectos según como se tome la boquilla/lengüeta y la presión ejercida sobre ella.

3e. Interfaz de configuración

Dado que es necesaria la visualización del estado de algunos parámetros y su personalización por parte del usuario, se ha incorporado una pantalla de visualización LCD monocromática tipo Nokia 5110. Esta posee una resolución de 84x48 píxeles y una retroiluminación LED susceptible de controlarse por diversos medios. Se ha escogido por su adecuado tamaño, bajo costo y la facilidad con que pueden producirse todo tipo de aplicaciones gráficas en ella. Inicialmente se adquirió y utilizó una pantalla LCD de 16x2 caracteres, pero una vez definidas las dimensiones y carcasa del instrumento resultó ser excesivamente grande, por lo que se recurrió a la mencionada 5110. Se han incorporado resistencias (ver sección "Arduino") en serie a sus líneas de conexión para protegerlas de la tensión de 5 v de los puertos del Arduino dado que estas utilizan una lógica de 3,3 v.

Luego de algunas pruebas se decidió acoplar un transistor NPN (337) a fin de controlar la intensidad de la retroiluminación por PWM (modulación de ancho de pulso) provisto por un puerto del Arduino. Esto fue necesario por dos razones: en primer lugar el Arduino solo puede proveer 5 v en sus puertos digitales, y la iluminación de la pantalla 5110 trabaja con 3,3 v, lo cual impide su directa conexión. En segundo lugar pero más determinante, sus LEDs de retroiluminación ya se encuentran conectados a 3,3 v, y requieren una conexión a 0 v (tierra) para cerrar su circuito y encenderse, por lo cual es necesario contar con un anclaje de corriente (en lugar de una fuente) controlable por el puerto del Arduino. Los transistores de tipo NPN son especialmente adecuados para esta tarea. La posibilidad de controlar la retroiluminación no responde solo a una función estética, sino que contribuye al ahorro de batería al desactivarse cuando no es necesaria la utilización de la pantalla.

Pantalla y botones

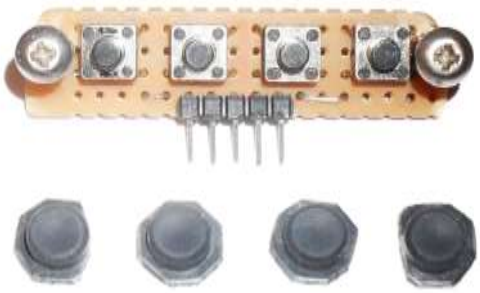
Mediante la programación del Arduino detallada en "Programación" se provee control y visualización de los siguientes elementos al usuario:

- **Número de instrumento sintetizado**: se visualiza una lista para elegir entre los 127 instrumentos GM estándar

- **Parámetro MIDI de expresión dinámica**: la fuerza del soplo aplicada post encendido de notas puede controlar los parámetros MIDI Volume, Breath, Expression o ninguno.
- **Parámetro MIDI controlado por los sensores "A" y "B"**: de manera independiente, ambos sensores pueden controlar Pitch Bend +, Pitch Bend -, Modulation, Portamento o ninguno.
- **Sensibilidad al soplo**: valor continuo de 0 al 9, donde 0 implica que toda nota presentará un velocity de 127 (sin sensibilidad), y 9 implica una sensibilidad lineal para todo el rango.
- **Transposición cromática**: transpone entre -24 y +24 semitonos la escala base.
- **"#" o "b"**: determina si la 8va llave subirá o bajará un semitono a la nota actual.
- **Encendido y estado del módulo inalámbrico**: éste puede ser apagado si se utiliza un cable MIDI estándar en el puerto DIN provisto para ahorrar batería. A su vez parpadea "OK!" si el módulo Xbee se encuentra activo.
- **Batería**: se muestra una barra que indica de manera aproximada el estado de la batería interna, y muestra "CARGAR!" si ésta se encuentra muy baja.

Anexos a esta pantalla se han dispuesto 4 pulsadores sin retención, conectados directamente a 4 entradas digitales del Arduino (haciendo uso de las resistencias "pull up" internas en estos puertos) para controlar el software del instrumento:

- Disminuir valor del parámetro (-)
- Aumentar valor del parámetro (+)
- Alternar parámetro escogido (rota de manera circular)
- Botón de pánico (apaga todos los sonidos y restablece los parámetros MIDI CC)



Pulsadores y botones plásticos

3f. Transmisión inalámbrica

Se exploraron cuatro opciones para el tratamiento de este objetivo: protocolo WIFI, Bluetooth, transmisión directa por RF y protocolo ZigBee. El protocolo WIFI es muy robusto en cuanto al envío correcto de información y tiene gran alcance, pero esto posee un coste significativo de tiempo de proceso interno, lo cual produce una latencia en ocasiones superior a los 1000 ms que resulta prohibitiva ante los requisitos del proyecto. Por su parte el Bluetooth estándar presenta una latencia de entre 150 y 200 ms, pudiendo reducirse hasta los 40 ms en el caso del Low Latency Bluetooth. Estos valores aún están muy por encima del límite de 10 a 15 ms necesarios para obtener una experiencia psicoacústica cercana al tiempo real. Además el Bluetooth estándar está diseñado para la comunicación entre dispositivos a corta distancia, por lo que su alcance es muy limitado (frecuentemente entre 2 y 10 m dependiendo del módulo).

Respecto a la transmisión por RF puro deben hacerse algunas observaciones. Se han estudiado casos de su utilización en aplicaciones similares. Su alcance es potencialmente alto (algunos usuarios de módulos NRF24L01 estándar han reportado entre 30 y 100 metros). Sin embargo se requiere la implementación de algún tipo de protocolo para su uso, el cual debe ser específicamente diseñado. Esto brinda la posibilidad de adaptarlo a las necesidades de cada proyecto, pero demandaría un trabajo excesivo a los efectos y plazos de esta Tesis.

Finalmente, se han realizado pruebas satisfactorias de transmisión MIDI por radiofrecuencia utilizando el "modo transparente" de los módulos Xbee, los cuales están basados en el protocolo ZigBee. En dicho modo, éstos funcionan como "reemplazo de cable serie", y se limitan a replicar sin cambios la información que les es provista de un módulo a otro. Esto resulta ser adecuado para el proyecto ya que MIDI es un protocolo serie, y fácilmente puede adaptarse para su transmisión directa por dicho medio. Para que esto sea posible, se utilizaron dos módulos, oficiando como emisor y receptor. Ambos fueron configurados para trabajar a la velocidad de datos de MIDI, 31.250 bps.

Xbee Serie 1

Es importante destacar la necesidad de utilizar un módulo adaptador de niveles lógicos para su conexión con una computadora personal a fin de configurarlos para el uso propuesto y evitar errores de lectura. Por otra parte, en los sistemas operativos Windows, 31.250 bps no es una velocidad de datos soportada de manera nativa. Como consecuencia, fue necesario indicar a los módulos Xbee que debían operar a tal velocidad sólo cuando se hubo realizado toda la configuración general. Una vez hecho esto no pudo volver a establecerse una conexión Xbee-Windows por medios convencionales, puesto que ya no coinciden las velocidades de transmisión de datos. Sin embargo, cabe aclarar que existen medios para salvar esta situación como el uso de software alternativo o de otro sistema operativo.

La latencia obtenida hasta el momento ronda los 3 a 6 ms dependiendo de las condiciones de uso (distancia y obstáculos, volumen de datos), lo cual cabe con suficiencia dentro de las expectativas. Por otro lado, el alcance de los módulos Xbee (serie 1) está especificado en 300 pies (90 m) en condiciones óptimas, y se ha comprobado un rango de al menos 10 m en interiores.

Eléctricamente fue necesario realizar ciertas adaptaciones a la señal dado que, como se ha mencionado, estos módulos operan con lógica de 3,3 v, mientras que Arduino y MIDI utilizan 5 v. En el instrumento (emisor), la adaptación de tensión de señal la provee un divisor resistivo de relación 2/3. De esta manera, la señal MIDI provista por el puerto digital Tx del Arduino es dirigida en paralelo tanto al conector MIDI-DIN como al mencionado divisor[25]. El módulo inalámbrico puede ser encendido o apagado por el usuario mediante la interfaz de configuración para ahorrar batería. Esto es posible gracias a la utilización de un transistor NPN (337) que funciona como interruptor de corriente para la alimentación 3,3 v del módulo Xbee, controlado por otro puerto digital del Arduino. Cabe aclarar que este último no puede utilizarse de manera directa para alimentar al módulo ya que provee 5 v en lugar de 3.3 v y posee un límite absoluto de 40 mA, y el Xbee requiere de un mínimo de 45 mA. Es por esto que se utiliza el mencionado transistor, cuya base es controlada por el puerto del Arduino a través de una resistencia de 1 KΩ, de modo que

[25] El receptor requiere otro tipo de adaptación, que es discutida en la sección “Módulo receptor”.

permita o bloquee el flujo de corriente hacia el módulo.

Por otro lado, aprovechando que el Xbee al entrar en funcionamiento inicia un ciclo alto-bajo de alrededor de 2 Hz en uno de sus pines (pin 15 “Associate”) para indicar su estado, se ha realizado una conexión entre éste y una entrada digital del Arduino para posibilitar la visualización en pantalla del texto “OK!”. De este modo el texto parpadea según la señal producida por el Xbee. Este tipo de conexión no requiere adaptación alguna, ya que el Arduino es capaz de interpretar como estado “alto” un nivel de tensión mínima de hasta 3 v, abarcando entonces los 3,3 v que provee el Xbee. Debido a que la carcasa elegida es de un material no conductor (PVC), no ofrece una barrera significativa para la señal RF, de modo que el módulo transmisor puede ser montado en el interior del instrumento sobre la pieza de madera de anclaje sin afectar su funcionamiento.

3g. Módulo receptor

Dada la naturaleza del proyecto, se ha diseñado y confeccionado un dispositivo receptor para retransmitir la señal MIDI a los diversos sintetizadores. Al igual que el instrumento, utiliza una batería recargable, un conector MIDI-DIN estándar y presenta un sistema de visualización de estado mediante 7 LEDs.

Una vez el Xbee alojado en su interior recibe datos serie del emisor, la señal debe ser nuevamente adecuada para cumplimentar las especificaciones de MIDI. Para ello se utiliza un transistor NPN de alta velocidad 13003. Si bien es posible utilizar los 3.3 v directamente en una conexión MIDI (ya que el protocolo trabaja por corriente más que por tensión), esta requiere alrededor de 22 mA, lo cual excede las capacidades del módulo Xbee, y hace necesario el método mencionado. Como en las situaciones anteriormente mencionadas con transistores 337, una resistencia limita la corriente que ingresa a la base del transistor para volverlo conductivo o no. Sin embargo, MIDI requiere que la línea de datos se mantenga normalmente a 5 v (alta), para obtener un 0 lógico, ya que el LED del optoacoplador[26] de entrada verá el mismo potencial (5v) en sus terminales y la corriente no circulará por él, impidiendo que se ilumine y produciendo en consecuencia el estado lógico bajo. Esto implica que la línea de datos debe poseer la capacidad para anclar corriente cuando se encuentra baja.

Por esta razón y para evitar problemas de potenciales (5 v conectado a 3,3 v) fue necesario invertir el funcionamiento habitual de la línea MIDI y del interruptor basado en NPN por medio de una compuerta NOT (inversora). En esta configuración, se conecta de forma permanente la línea de datos a 0 v, y es la línea –originalmente- de 5 v (con su correspondiente resistencia de 220 Ω en serie) la que resulta alta o baja según la salida del Xbee. El transistor 13003 es controlado en su base por el módulo, su emisor es conectado a 0 v de manera directa, y su colector es puesto en paralelo a la línea MIDI de 5 v luego de la resistencia de 220 Ω. De esta manera, al recibir un nivel

[26] Dispositivo que funciona como un interruptor activado mediante la luz emitida por un diodo LED que satura un componente optoelectrónico, normalmente en forma de fototransistor o fototriac.

lógico bajo, el transistor no conduce, y la línea permanece normalmente alta, encendiendo el LED del optoacoplador, lo cual es interpretado como un estado bajo por el dispositivo MIDI. Al recibir un nivel alto, el transistor se vuelve conductivo y efectivamente ofrece una vía de baja resistencia hacia 0 v luego de la resistencia de 220 Ω, de manera que el LED del optoacoplador conectado en paralelo al transistor ahora presenta 0 v en ambos terminales, y no enciende, produciendo un estado alto en el dispositivo de destino.

Módulo receptor (frente, interior, reverso)

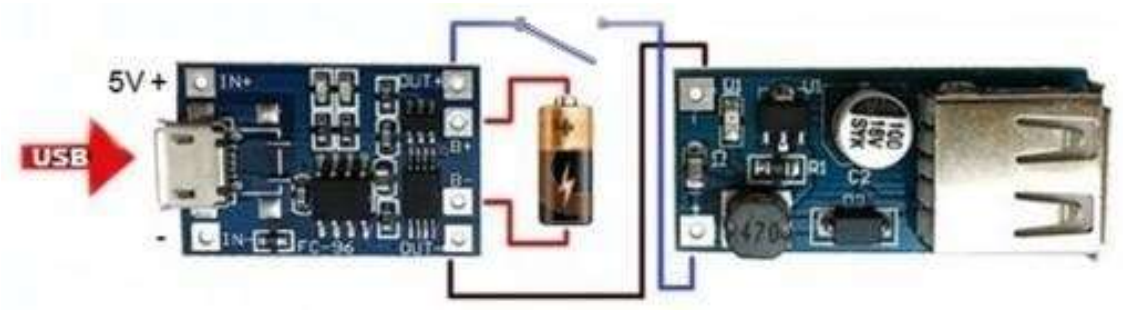
En paralelo a ello, se ha dispuesto un transistor 337 para controlar un LED indicador de transmisión (Tx) en el módulo receptor. Este LED enciende cuando el sistema recibe un 1 binario, de manera que el usuario pueda ver un parpadeo “testigo”. Del mismo modo que en el instrumento, se ha utilizado otro LED en conexión directa al pin 15 del Xbee (Associate), el cual parpadea cuando el módulo se encuentra en funcionamiento normal, además de otro conectado al pin 6 (RSSI), que posee una salida PWM proporcional a la intensidad de la señal recibida. Por último, existen tres LED adicionales para indicar el estado de la batería: el primero es explicado en la sección “Alimentación eléctrica” y da aviso al usuario que la batería debe ser cargada prontamente. Los restantes son conectados directamente en reemplazo de los LED de estado del módulo cargador TP4056, pero dispuestos de manera que iluminen el mismo orificio en la carcasa del receptor. Uno de ellos es un LED “baliza”, que alterna rojo-azul al ser encendido, e indica que la batería está siendo cargada, mientras que el otro es azul estático y enciende cuando se ha completado la carga.

3h. Alimentación eléctrica

La alimentación es provista por baterías de iones de litio recargables, la misma tecnología que se utiliza en teléfonos celulares. Para el instrumento se ha utilizado una batería modelo 18350 con capacidad de 900 mAh, debido a sus dimensiones adecuadas para el montaje. En el receptor se utilizó un modelo EP500 de 1160 mAh que pudo recuperarse de un teléfono en desuso, y que posee una morfología plana adecuada para situarla en la caja plástica empleada. Éstas poseen un rango de tensión que va desde 4.2 v con la carga completa, a alrededor de 2,5 volt totalmente descargadas. Es necesario entonces elevar esta tensión -que además no es constante- a los 5 v estabilizados que requieren el Arduino y el sensor MPX5010, entre otros. Para esta tarea se utiliza un elevador de conmutación (step-up) cuya eficiencia de conversión es superior al 90% (algo importante a tener en cuenta al trabajar con baterías) el cual provee una salida muy precisa y estable de 5 v, siempre que la batería posea carga suficiente. Para los elementos que utilizan 3.3 v (la pantalla 5110, el módulo Xbee y el MPR121) se ha incorporado un regulador de tensión LM1117, el cual es de baja pérdida y provee una salida estable de 3.3 v a partir de los 5 v antes mencionados. Nuevamente, por limitaciones de corriente, no es posible utilizar la salida de 3.3 v del Arduino para alimentar este conjunto de elementos.

EP500

18350



Sistema de alimentación eléctrica (cargador – batería – elevador)

El estado de carga estas baterías es proporcional a la tensión de las mismas. Utilizando esta propiedad, en el instrumento es registrado a todo momento el nivel por una de las entradas analógicas a través de una resistencia de 4,7 KΩ y reportado en la pantalla. Esta resistencia es necesaria para limitar la corriente y evitar que el Arduino sea incorrectamente alimentado y encendido a través del puerto analógico cuando el instrumento se encuentre apagado. Dado que el

receptor no posee pantalla, esta tarea es realizada por un sistema diseñado específicamente para tal fin, el cual consta de una compuerta lógica NOT basada en un transistor NPN. Aprovechando su tensión de saturación fija es posible ajustar un divisor resistivo para que el transistor cambie de estado en un punto calibrado aproximadamente a 3 v. De esta manera, un LED enciende gradualmente una vez el nivel de la batería desciende a dicho punto, a fin de notificar al usuario de que debe ser recargada.

Por otro lado, se ha recurrido a la incorporación de un módulo basado en TP4056. Este gestiona de forma integral la carga y protección de la batería, de modo que se puede conectar una fuente estándar de 5 v (como un cargador de celular) para la recarga sin necesidad de abrir o desmontar los dispositivos. El módulo cuenta con un transistor MOSFET en su salida, calibrado de modo tal que si la batería desciende su nivel por debajo de los 2,5 v, es automáticamente desconectada de la carga (es decir, de la electrónica del instrumento). Todo este sistema de alimentación es replicado en el receptor inalámbrico.

3i. Arduino

Tras pruebas iniciales con Arduino UNO y MEGA, se determinó que el módulo microcontrolador más adecuado a utilizar sería el Arduino Nano (328). Éste presenta las mismas características funcionales que el modelo UNO, pero sus dimensiones son sensiblemente menores. Las especificaciones principales de este son:

- Procesador: Atmel ATmega328 16 mhz
- Tensión de alimentación: 5 V
- Puertos de entrada/salida digitales: 14
- Puertos de entrada analógicos: 8
- Memoria: FLASH 32 KB – RAM 2 KB – EEPROM 1 KB
- Dimensiones: 45 x 18 mm

Arduino Nano

La plataforma Arduino utiliza un lenguaje de programación de alto nivel basado en Java/C++. Su sintaxis es sencilla, pero capaz de realizar operaciones de alta complejidad. Cuenta además con un entorno de desarrollo integrado (*IDE*) basado en Processing.

El mismo es alimentado por el módulo elevador de tensión directamente a través de su entrada de 5 v no regulada (pin 5 v). Esto solo es recomendable si se dispone de una fuente regulada y fiable, de otro modo deben tomarse medidas alternativas para garantizar un funcionamiento estable del microcontrolador. A través de las conexiones existentes en la placa base (ver sección "Circuitos electrónicos"), los puertos resultan utilizados según la siguiente tabla (se señalan los casos en que existe una resistencia intermediaria):

Puertos digitales	**Puertos analógicos**
0 - Monitor de estado Xbee	A0 - MPX5010
1 - Tx / Salida MIDI	A1 - Sensor A (10 KΩ)
2 – [no utilizado]	A2 - Sensor B (10 KΩ)

3 - Control PWM de retroiluminación LCD	A3 - Nivel de batería (4.7 KΩ)
4 - LCD (SCK) (10 KΩ)	A4 - MPR121 (SDA)
5 - LCD (MOSI/DIN) (10 KΩ)	A5 - MPR121 (SCL)
6 - LCD (DC) (10 KΩ)	A6 - [no utilizado]
7 - LCD (RESET) (10 KΩ)	A7 - [no utilizado]
8 - LCD (CS/CE) (1 KΩ)	
9 - Botón - Pánico	
10 - Botón - Desplazamiento	
11 - Botón - “-”	
12 - Botón - “+”	
13 - Control encendido Xbee	

3j. Carcasa y montaje interno

Como carcasa para el instrumento se ha utilizado, por su bajo costo y amplia disponibilidad, un tubo de PVC de 5 cm de diámetro y 30 cm de largo, que posteriormente fue colapsado mediante el uso de calor de manera que su sección adquiera la forma aproximada de óvalo de 6 x 3 cm con dos caras planas paralelas. Esto provee una sensación continua al tacto, sin ángulos que pudieran interferir el agarre o la comodidad de las manos. Al mismo tiempo, sus caras planas brindan superficies adecuadas para el montaje de los diferentes componentes tanto internos como externos. Sobre una de ellas se ha efectuado un corte rectangular de 26 x 4 cm que sirve de tapa de acceso. Posteriormente todo fue revestido utilizando PVC adhesivo negro del tipo utilizado para ploteos y señalética, entre otros.

Carcasa revestida en negro (con boquilla montada)

Este proceso de creación de la carcasa y diseño de los puntos de anclaje y soporte de los componentes ha resultado ser uno de los puntos más complejos (y demandantes en cuanto a tiempo) de todo el desarrollo. La dificultad principal residió en el hecho de que debían diseñarse y manufacturarse todo tipo de soportes internos inmediatamente a medida que se definían los elementos específicos a utilizar, imposibilitando un proceso por separado y predeterminado de estos factores. En algunos casos fue necesario efectuar modificaciones mecánicas sobre los módulos utilizados, como perforaciones o soldaduras fuera de las especificaciones previstas por los fabricantes solo a efectos de permitir los diversos montajes. También fueron necesarias reiteradas

modificaciones a la estructura prevista en la carcasa. Tal fue el caso de la pantalla 5110, que inicialmente se colocó atravesando la superficie del instrumento, lo cual perjudicaba notablemente el aspecto estético, pero finalmente se rediseñó su montaje interno de manera que pudiera abrirse una pequeña ventana de las dimensiones exactas de visualización.

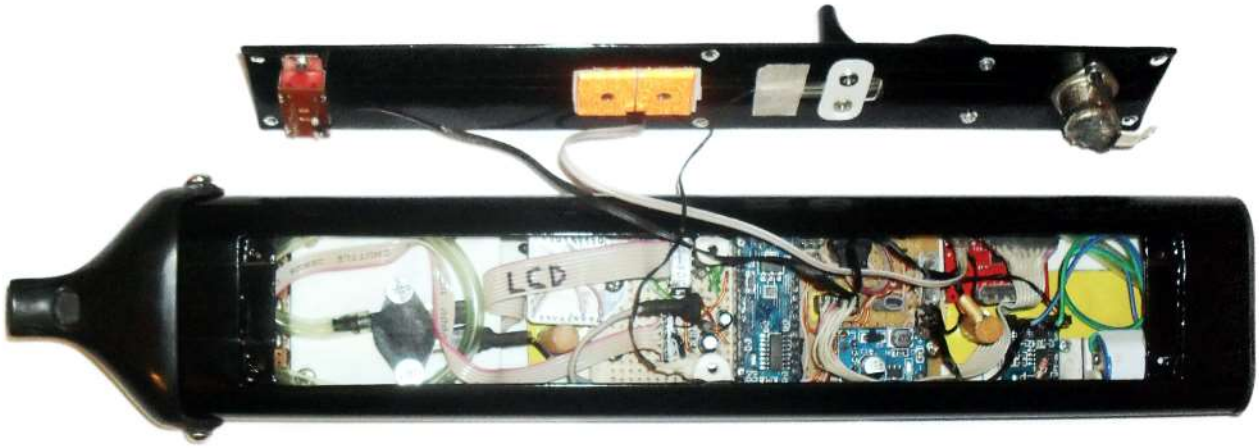
Tapa con un portavoz y plástico separador para el FSR 402 (cara interna y externa)

Como se ha mencionado anteriormente, se ha creado una base de madera terciada que sirve de anclaje para la mayor parte de los componentes internos y al mismo tiempo, mediante tuercas y tornillos, provee el agarre y los conductores eléctricos para las 8 llaves de la parte superior. Sobre ella también se han colocado dos torrecillas de madera que hacen contacto con la tapa en las zonas de acción de los pulgares, de manera que no pueda deformarse por la presión dicha tapa ni dañarse la carcasa.

Base de madera con soportes y conector para los electrodos

Sobre la tapa removible mencionada se han colocado el conector MIDI-DIN, las dos llaves portavoces, un soporte para el pulgar derecho (construido a partir del pico silbador de una pava en

desuso) desde donde puede accionarse el sensor “A” y el interruptor de encendido general. Sobre la cara superior externa de la carcasa han sido montadas las 8 restantes llaves de digitación, la pantalla 5110 y los 4 pulsadores de control. La boquilla actual está colocada en la pieza que actúa de “tapa” superior y es fijada por medio de dos tornillos. En la cara inferior del instrumento se ha colocado una pieza de goma que actúa de anclaje para el tubo de evacuación de aire y humedad. Sobre uno de los laterales se exhibe el conector Micro USB para cargar la batería.



Vista interna completa con boquilla y tapa.

El módulo receptor (ilustrado en la sección correspondiente) ha sido montado en una caja de proyectos plástica de propósitos generales, con el conector MIDI-DIN, Micro USB, el interruptor de encendido y los LEDs de estado correspondientes. Su interior ha sido modificado para alojar la batería EP500, pero pudieron aprovecharse las torrecillas incorporadas para la sujeción de la placa principal mediante tornillos. Las perforaciones de visualización de los LEDs fueron cubiertas con silicona traslúcida del tipo utilizado en caliente como adhesivo.

3k. Circuitos electrónicos

Si bien ha sido descrito en las secciones correspondientes gran parte del diseño y funcionamiento electrónico del instrumento y receptor, se exponen aquí los diagramas esquemáticos y algunas aclaraciones pertinentes. Por razones de claridad ha sido excluido de los mismos el sistema de alimentación, cuyo diseño se encuentra expuesto en "Alimentación eléctrica". En sentido general ambos circuitos se componen de módulos y sensores con interconexiones y componentes de apoyo. Los transistores requieren de una corriente limitada en su base, para lo cual se han utilizado resistencias de 4,7 KΩ, lo que además ayuda a mantener bajos los niveles de corriente demandados a los microcontroladores. Los LEDs utilizan a su vez resistencias limitadoras cuyo valor varía según el color empleado y el tipo de función. En el caso de Tx, presenta un parpadeo tan veloz que debió optimizarse su brillo por medio de una resistencia de 1KΩ.

Para el instrumento ha sido necesario el diseño y manufactura de una placa base principal, sobre pertinax[27] perforado estándar, que sirve de alojamiento para el Arduino Nano escogido y provee terminales de conexión para todos los componentes del instrumento. También se alojan en ella todos los componentes electrónicos adicionales necesarios para las diversas funciones, como ser: resistencias, capacitores de filtrado, transistores, el regulador LM1117 y el elevador de conmutación para proveer la alimentación de 5v. Es fijada mediante 4 tornillos a la pieza de madera terciada que soporta las llaves y otros componentes del instrumento.

El receptor ha sido construido sobre una segunda placa de pertinax perforado, y consta del módulo Xbee, la sección de alimentación eléctrica, un transistor 13003 de alta velocidad para la conversión de niveles y corrientes al estándar MIDI, el sistema basado en una compuerta inversora (NOT) para dar aviso de batería baja, más una serie de LEDs indicadores de estado con sus correspondientes componentes anexos (resistencias y transistores).

---

[27] Las placas de pertinax se realizan con resina y fibra de vidrio reforzada, sobre la que se aplica una capa de cobre. Existen en numerosas presentaciones, incluyendo pistas y perforaciones prefabricadas.

Diagrama esquemático del instrumento

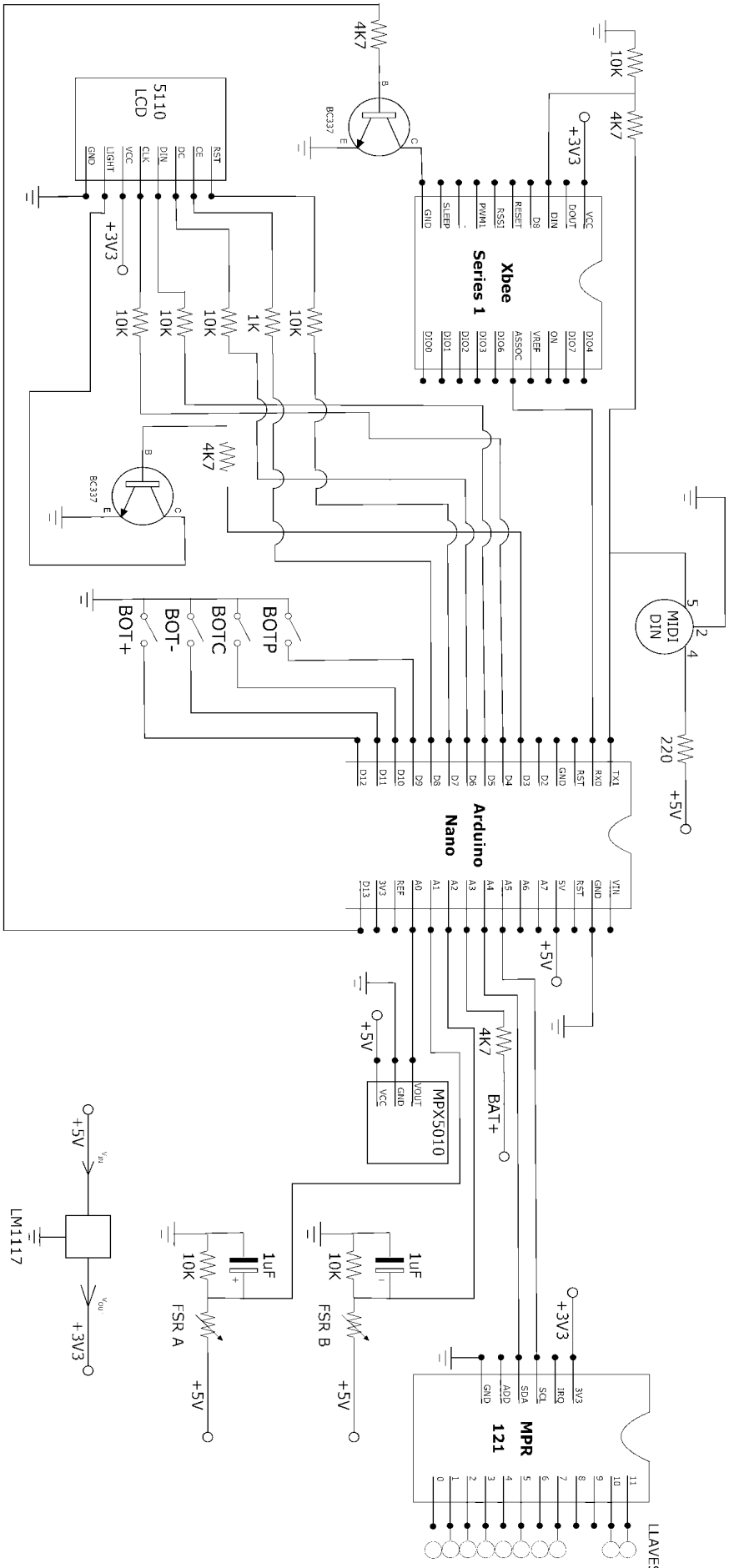
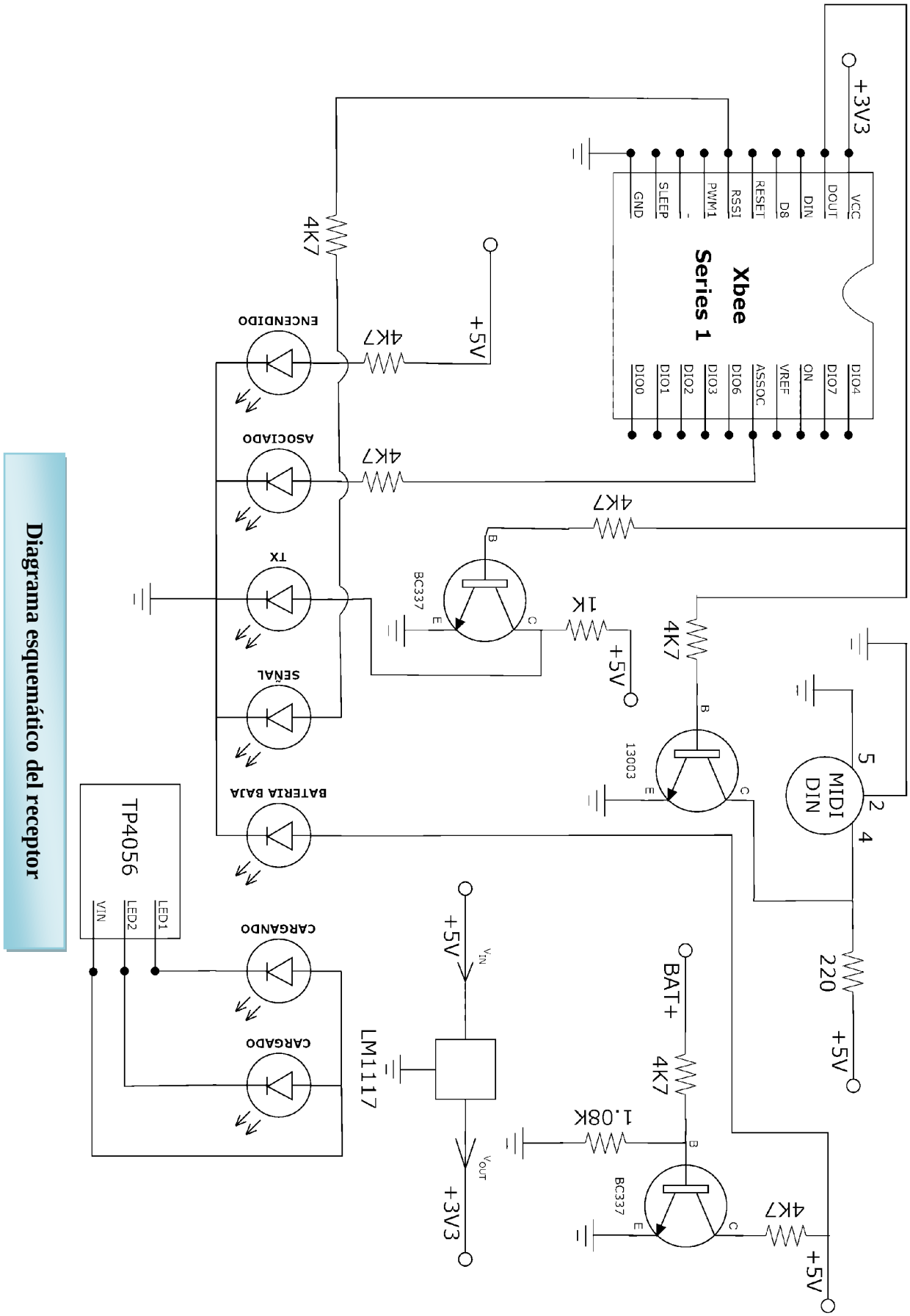


Diagrama esquemático del receptor



3l. Dinámica de funcionamiento

El instrumento ha sido concebido, al menos a priori, de manera monofónica, es decir está limitado a una sola nota en un mismo intervalo de tiempo. Hacerlo de otro modo implicaría un rediseño integral en algunos puntos centrales del proyecto, aunque no se descarta explorar la incorporación de capacidades polifónicas en el futuro.

La generación de notas responde a la siguiente dinámica de eventos y condiciones: al superar un determinado umbral en el valor de presión recogido por el sensor MPX5010, es decir, al iniciarse el soplo por parte del usuario, se aguarda un pequeño intervalo de tiempo. Una vez transcurrido, se mide nuevamente el valor del sensor. Este segundo valor alcanzado tras un tiempo fijo conocido permite estimar la intensidad inicial del soplo del usuario, y asignar así un valor de Velocity (intensidad) con el que se producirá la nota. Este procedimiento es análogo al utilizado en los controladores MIDI de teclado sensitivos (pianos eléctricos, órganos y otros) en los cuales la intensidad de cada nota es asignada al medir el diferencial de tiempo entre dos contactos físicamente desfasados en cada tecla. Existe una proporcionalidad entre la fuerza aplicada a la tecla y la velocidad en la que esta desciende y acciona ambos contactos, y es por ello que la especificación MIDI llama “Velocity” al parámetro que controla la intensidad inicial de las notas.

Una vez se ha determinado que una nota debe sonar tras el procedimiento anterior, se lee el estado de las 10 llaves de digitación. Primero se leen las 7 llaves diatónicas y se asigna una nota de referencia. Luego se le suma o resta un semitono si está siendo tocada la llave #/b. A continuación se añaden múltiplos de 12 semitonos (una octava) según el estado de los portavoces. Finalmente, se aplica la suma o resta correspondiente al valor de transposición general escogida en la configuración por parte del usuario. Tras estas operaciones, se dispone de un número de nota MIDI y de un valor de Velocity y se procede a producir un mensaje de Note-On. El sintetizador escogido deberá entonces comenzar a reproducir el sonido correspondiente a la acción del usuario.

A partir de aquí se aplican una serie de condiciones nuevas. A intervalos regulares es leído el valor del sensor MPX5010 y se envía un mensaje MIDI de expresión (CC) según lo escogido por el

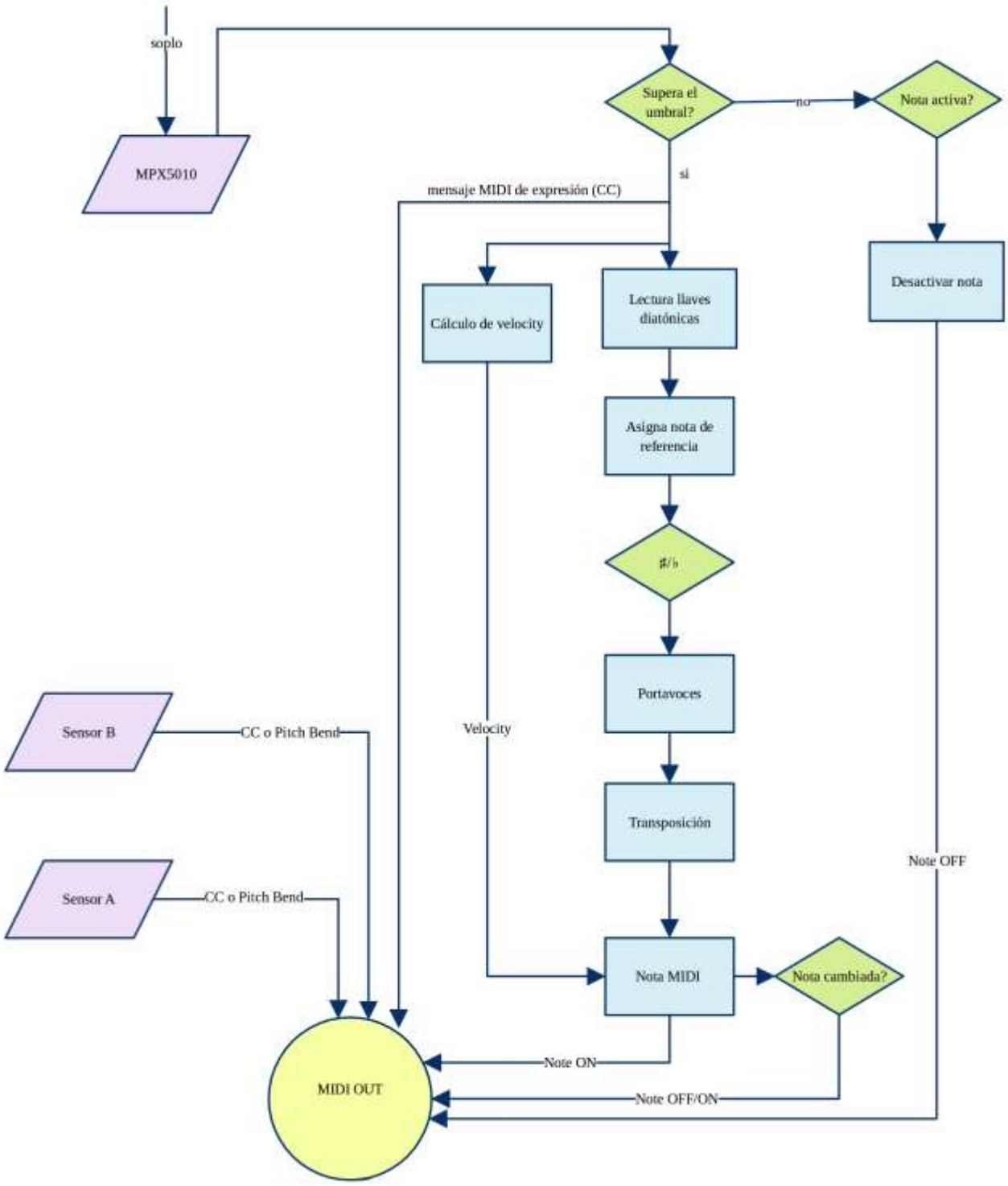
usuario, a fin de controlar la evolución dinámica de la nota que está sonando. De este mismo modo son leídos los sensores "A" y "B" y enviados los mensajes (CC o Pitch Bend) correspondientes.

Si existe una modificación en el estado de las 10 llaves, es decir, que se ha cambiado la digitación en algún modo, el sistema entonces produce un nuevo mensaje de Note-On con Velocity según el estado actual del sensor MPX5010 y asignando una nota tras reproducir el procedimiento original. Inmediatamente tras esto, se produce un mensaje de Note-Off para apagar la nota anterior. Este es un punto único en el que existen técnicamente dos notas sonando al mismo tiempo, hasta que se ejecute el Note-Off. Ha sido concebido así tras contemplar la posibilidad de utilizar la capacidad de *Portamento* existente en algunos sintetizadores, el cual requiere para su funcionamiento que una nota se solape ligeramente con la siguiente.

El último caso procesado por el sistema es, naturalmente, el descenso del valor del MPX5010 por debajo del umbral mencionado (cese del soplado). Este evento produce un mensaje de Note-Off. Así quedan contempladas todas las posibilidades de producción e interrupción de sonidos. Cabe recordar la eventualidad de que el usuario utilice el pulsador de "pánico", el cual enviará un mensaje de apagado de todas las notas y restablecerá los valores iniciales para cada parámetro expresivo. Este mismo recurso es utilizado por el software de manera automática cuando el usuario modifica cualquier parámetro en la configuración.

A continuación se grafica de manera general este proceso en un diagrama de flujo.

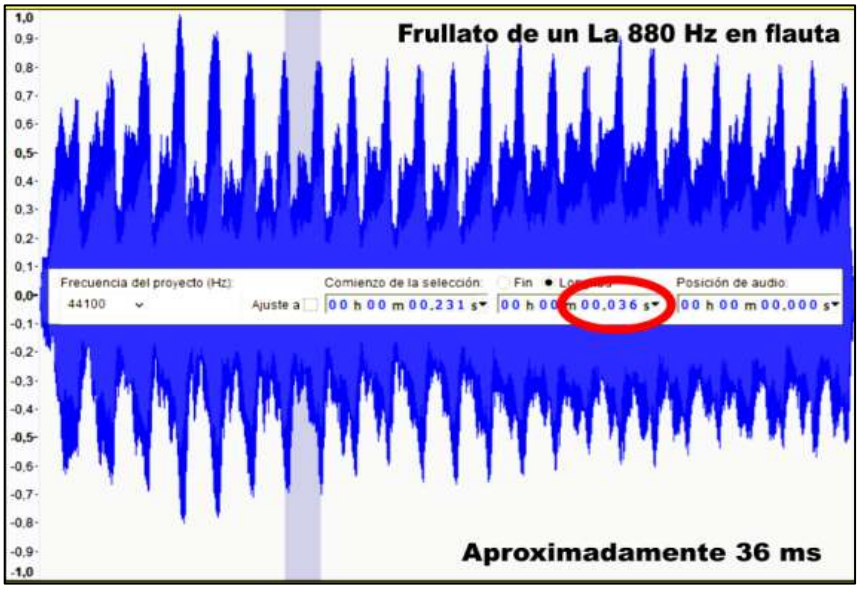
Esquema general de procesamiento de notas

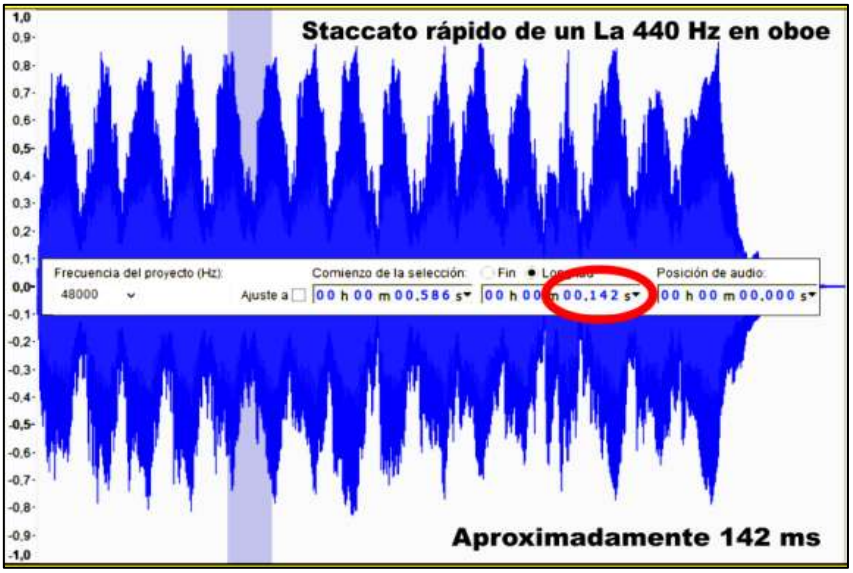


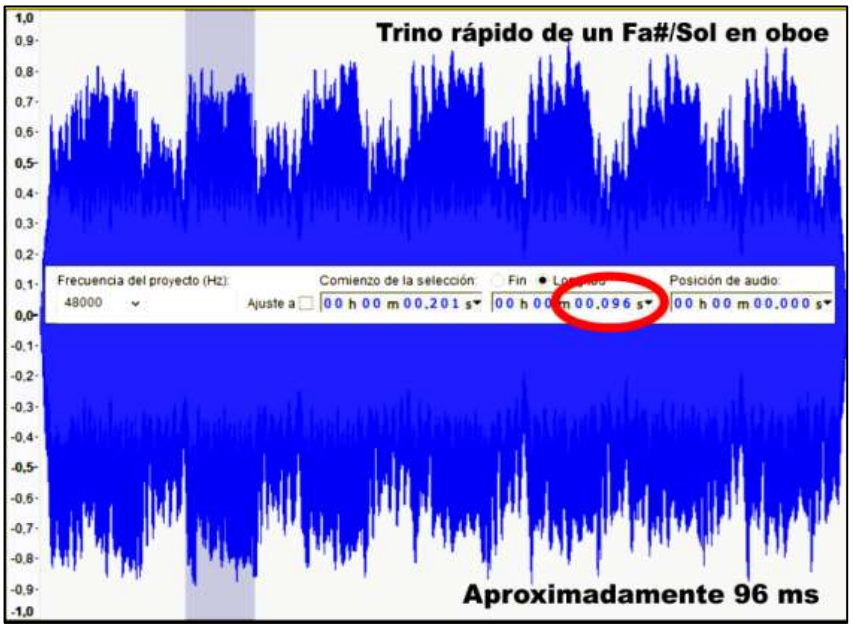
Paralelo a ello, con cada cambio de parámetros de configuración se indica internamente al sistema que debe guardar en memoria EEPROM el estado de aquella en caso de que se ejecute una nueva nota. Esta memoria es incluida en el Arduino a fin de registrar variables e información que deba

permanecer tras una interrupción de energía, de modo que pueda restaurarse al iniciar nuevamente. Esta metodología de guardar información en la EEPROM solo cuando se ejecuta una nota -y por única vez hasta que se produzca un nuevo cambio de parámetros-, responde a la necesidad de proteger a esta memoria de excesivos ciclos de reescritura que podrían reducir su vida útil.

Se ha realizado un análisis de los tiempos de cambio o reiteración de notas en algunas de las técnicas de ejecución más rápidas existentes para flauta y oboe (ver más abajo). Los resultados arrojan que, en general, la técnica más veloz es el *frullato*, con un período mínimo de 36 ms aproximadamente, seguido por el trino (ligado) de digitación con 96 ms. Atento a estos valores, se ha determinado -a priori- utilizar una grilla de lectura de 5 ms en el sistema mencionado, la cual ayuda a reducir el volumen de datos generados de manera considerable, a la vez que proporciona un tiempo prudente de respuesta al cambio de digitaciones. De esta manera se ignoran los numerosos cambios de estado electrónicos registrados por la enorme precisión y velocidad de respuesta del MPR121 para con sus electrodos. No utilizar este método o alguno equivalente implicaría que, por ejemplo, al cambiar de una nota que implique varios dedos a otra que implique pocos o una combinación muy diferente, el sistema registraría y haría sonar (muy breve pero perceptiblemente) una nota por cada cambio individual de las llaves.







Entre otras situaciones tratadas por el software se cuenta la activación momentánea de la retroiluminación LED en la pantalla 5110 tras utilizar un pulsador de control, y su desactivación tras un período de 10 segundos de inactividad, de manera que pueda ahorrarse energía cuando no es requerida. También es posible controlar el encendido y apagado del módulo Xbee, como se ha tratado en la sección correspondiente, por medio de un transistor NPN conectado a un puerto digital del Arduino cuyo estado es definido por el usuario a voluntad. La lectura del nivel de la batería es altamente inestable debido a las propiedades eléctricas y de consumo del instrumento. Por esta razón se ha implementado un sistema de promediado a intervalos de 10 segundos que otorga una visualización más adecuada para el control del usuario.

3m. Programación

El código fuente completo en lenguaje Arduino del proyecto puede encontrarse en el Anexo VII de la presente Tesis. En esta sección se expondrán las funciones existentes en el mismo y cuestiones relacionadas a la relación hardware-software del instrumento. El proyecto consta de un bloque principal, contenido en el archivo "proyecto_general.ino", y dos anexos tipo cabecera (header) contenidos en "instrumentos.h" y "digitaciones.h". A su vez se han utilizado las siguientes librerías externas para el tratamiento de algunos componentes y funciones:

- **LCD5110_Graph.h**: administra el control de la pantalla LCD 5110, y provee funciones automatizadas para el dibujado de texto y otros.
- **avr/pgmspace.h**: permite el almacenamiento y lectura de variables en memoria de programa (FLASH) para grandes volúmenes de datos que dificultarían su uso en RAM.
- **Metro.h**: provee un metrónomo de tiempo asignable para la ejecución de instrucciones a intervalos regulares sensiblemente mayores a la velocidad del Loop.
- **Wire.h**: permite la comunicación con dispositivos I^2C como el MPR121.
- **Adafruit_MPR121.h**: administra el funcionamiento y lectura del sensor táctil MPR121. Esta librería debió ser modificada ligeramente para incluir la función de Debounce (filtrado de ruidos eléctrico) y garantizar un funcionamiento robusto del sensor. Debounce es una función nativa del MPR121, pero no es implementada en la librería.
- **MIDI.h**: administra el envío y recepción de mensajes MIDI.
- **Bounce2.h**: provee funciones de Debounce para los puertos digitales del Arduino, de manera que puedan utilizarse pulsadores de manera estable.
- **EEPROM.h**: permite almacenar y recuperar datos en memoria EEPROM.

A continuación se exponen las funciones y procesos utilizados en el código.

- **Declaraciones iniciales**: aquí se incluyen las librerías y dependencias y se declaran todas las

variables, objetos y definiciones a utilizar en las distintas funciones.

- **Void setup**: aquí se realizan las operaciones de inicialización del sistema como asignar modos de funcionamiento de pines digitales, arranque del puerto serie en protocolo MIDI, de la pantalla 5110 y módulo MPR121. Además se realiza una lectura de la batería para mostrar inmediatamente, se envían mensajes de reinicio MIDI (pánico), se cargan los parámetros de configuración guardados en memoria EEPROM y se envía un mensaje de cambio de instrumento para establecer el último utilizado.

- **Void loop()**: este bloque es reiterado *ad infinitum* por el microcontrolador y es utilizado para controlar o producir todos los eventos de funcionamiento del sistema. Aquí se toma periódicamente una lectura del sensor MPX5010 y los FSR A y B, además de enviar -si corresponde- los mensajes MIDI apropiados para estos últimos. La función principal llamada en el loop es sonarNotas(), que administra toda la producción de notas MIDI. Adicionalmente se imprime el contenido correspondiente en la pantalla LCD, se controla el retroiluminador, se leen y administran las funciones de los pulsadores, se estima y promedia -cada 10 segundos- el nivel de batería. Finalmente se aplica un retardo específico que determinará la frecuencia del bloque loop, sirviendo como grilla espaciadora de eventos.

- **Void guardarParametros() y void cargarParametros()**: estas dos funciones realizan el guardado y recuperación de los datos de configuración de usuario del sistema en memoria EEPROM. Dado que ésta utiliza valores tipo *byte* sin signo (8 bits, positivos), es necesario realizar una operación de desfasaje (+100 en escritura, -100 en lectura) para almacenar el parámetro de transposición cuyo rango incluye números negativos (-24 a +24).

- **Void calcularAB()**: lee los valores de A y B, y determina qué mensajes MIDI de CC o Pitch Bend deben enviarse según lo escogido por el usuario, además de adaptar los valores al rango requerido por cada uno.

- **Void ejecutarAB()**: envía los mensajes correspondientes para A y B. También administra los umbrales de actividad/inactividad para que se envíen los mensajes únicamente si existe un estímulo (presión) mínimo, y sean totalmente desactivados al regresar a su estado de reposo. En el caso del Portamento, es necesario enviar un mensaje de activación y

desactivación del mismo además del propio valor contínuo.

- **Void imprimeLCD**: administra de manera integral el contenido a mostrar en pantalla. Para ello son leídos regularmente los valores de configuración, estado de los pulsadores, de la batería, del módulo Xbee y otros. Para mostrar un cursor de selección actual se recurre a la inversión de los píxeles durante la impresión del parámetro correspondiente.
- **String txtInstr()**: realiza una lectura de la matriz de cadenas de caracteres (texto) almacenada en la memoria FLASH que representa la lista completa de instrumentos GM, y devuelve el texto correspondiente al instrumento actual para su impresión en pantalla.
- **Void actualizarBotones()**: lee el estado de los 4 pulsadores y ejecuta las instrucciones correspondientes para cada uno en caso que sean presionados. También indica que la retroiluminación debe activarse por 10 segundos tras cada pulsación.
- **Void dibujarBateria()**: esta función realiza una conversión numérica del valor de estado de la batería y dibuja en pantalla el gráfico correspondiente. También determina si debe mostrarse la advertencia “Cargar!” en caso que el nivel sea bajo.
- **Void promediar()**: realiza una cantidad predeterminada de lecturas al nivel de la batería, las cuales presentan altas discrepancias (valor inestable). Tras ello determina el valor máximo y mínimo y devuelve el promedio de ambos.
- **Void sonarNotas()**: esta función es la encargada de producir las notas MIDI según el valor recogido del sensor MPX5010. Está basada en una serie de condicionales anidados a fin de administrar y aplicar el esquema explicado en “Dinámica de funcionamiento”.
- **Void obtenerNota()**: lee el estado de las 10 llaves y ejecuta todas las operaciones aritméticas necesarias para obtener un número de nota MIDI correspondiente a lo digitado por el usuario, teniendo en cuenta el valor de transposición general. Para ello compara el estado de las 7 llaves diatónicas con la lista de digitaciones posibles incluidas en “digitaciones.h” a fin de determinar la nota de partida.
- **Void panico()**: envía los mensajes MIDI *Reset all controllers*, *All sound off* y *All notes off*.
- **Void apagarLuz() y void encenderLuz()**: estas funciones administran el encendido y

apagado (tras 10 segundos) de la retroiluminación.

- **Void cambiarParametro(int8_t numero)**: cambia el valor de cada parámetro según la selección actual. La variable *numero* es utilizada para indicar al sistema si debe incrementarse o disminuirse el valor del mismo según se haya presionado el botón "+" o "-".
- **Void textoParametros(uint8_t numeroParametro)**: devuelve una cadena de caracteres (texto) que representa cada opción de los parámetros de expresión para su visualización en pantalla. La variable *numeroParametro* es utilizada para indicar el parámetro actual.

Contenido de "digitaciones.h":

- **Declaraciones**: se declaran dos constantes para administrar internamente las funciones: CANTIDAD_DIGITACIONES, que será utilizada para definir el tamaño de la matriz *mapa*, y transp_interna, que aplica una transposición auxiliar interna para facilitar el cambio de octava básica predeterminada.
- **Struct mapa_entradas**: declara un tipo de estructura de datos (bidimensional) que almacenará un número de nota para una combinación específica de llaves.
- **Struct mapa_entradas mapa[CANTIDAD_DIGITACIONES]**: esta estructura define una matriz de datos de tipo *mapa_entradas* con contenido predefinido, creando una matriz bidimensional digitación-nota. En ella se almacenan todas las opciones de digitación con sus correspondientes números de notas MIDI. Esto permite establecer y modificar con facilidad un sistema de digitación integral, a la vez que posibilita su expansión futura.

Contenido de "instrumentos.h":

- **Typedef struct {…} instrTipo**: define una matriz de 10 caracteres, que es el máximo que cabrá en pantalla posteriormente, más un caracter final de terminación. Esto será utilizado para almacenar la lista de instrumentos GM. Se define utilizando *typedef* por requerimiento de la librería *pgmspace.h* y permitir su almacenamiento en memoria de programa (FLASH).
- **Const instrTipo nombreInstrumento [11]**: define una matriz de 128 elementos de tipo

instrTipo que almacena las cadenas de caracteres que representan los nombres de instrumentos del estándar GM en el orden correspondiente. Esta matriz de 11 caracteres y 128 elementos tiene un tamaño de 1408 bytes (11 x 128), lo cual representa casi un 67% de la memoria RAM con que cuenta el Arduino Nano (2048 bytes). Almacenarla en RAM tornaría imposible el funcionamiento del resto del programa por falta de memoria, razón por la cual esta lista es colocada de manera íntegra en memoria de programa. Por otra parte, dada su naturaleza, dicha lista no es sensible a posibles bajas velocidades de lectura o cálculo causadas por los tiempos mayores de acceso de la memoria FLASH respecto de la RAM.

4. Conclusión y consideraciones finales

Transcurrida la etapa de investigación y desarrollo se ha logrado producir controlador MIDI que satisface en gran medida los requerimientos iniciales planteados. Se han realizado pruebas exitosas de interpretación musical, y el sistema inalámbrico -si bien presenta algunas limitaciones de alcance- brinda una libertad de movimiento presente en muy pocos desarrollos similares. Las dificultades principales que se han enfrentado radicaron en la solución de aspectos técnicos y de montaje, mayormente debido a las constantes modificaciones que fueron necesarias a medida que otros factores relacionados eran definidos. Otro aspecto que presentó inconvenientes fue la medición del nivel de carga de la batería, el cual presentaba serias inestabilidades de lectura. Aunque los indicios apuntan a una combinación de variaciones en el consumo eléctrico del sistema (lo que en baterías de calidad media conlleva inestabilidades en la tensión proporcionada) e interferencias provocadas por el elevador step-up, aún no se ha logrado determinar con precisión su causa. Para tratar esto se diseñó un sistema óptico de medición de tensión, pero tampoco fue capaz de proveer una lectura suficientemente estable. También se recurrió al uso de un capacitor de filtrado para estabilizar el valor de manera provisoria, y finalmente se diseñó un algoritmo de promediado por software.

Durante el desarrollo de la investigación han surgido numerosas posibilidades de ampliación y refinamiento para el proyecto. Por ejemplo, se disponía de un vasto reservorio de transistores al iniciar el diseño eléctrico del sistema, por lo cual fueron aprovechados en la construcción de los circuitos. Sin embargo, en caso de plantear una replicación del instrumento, sería aconsejable su reemplazo por transistores tipo MOSFET, los cuales no requieren de una corriente circulando por su base (*gate* en caso del MOSFET) para alcanzar un estado de saturación y conductividad, sino de una tensión determinada. Esto, sumado al hecho de que en general poseen una mayor eficiencia y menor resistencia interna en conductividad, se traduce en un consumo menor de corriente generalizado y la consecuente menor disipación térmica. Las mejoras y modificaciones que se sugieren a futuro o en caso de la replicación del proyecto por parte de terceros son las siguientes:

- Medición y registro de especificaciones técnicas (particularmente parámetros eléctricos).
- Confección de un manual de usuario.
- Introducción de más posibilidades de digitación basadas en otros instrumentos.
- Incorporación de un sistema de soporte con correa u otro medio.
- Implementación de soporte MIDI-USB por medio de la inclusión de un segundo Arduino en el módulo receptor u otros medios.
- Añadir un sistema de calibración *in situ* producible por el usuario de los umbrales inferior y superior de sensibilidad de los distintos sensores.
- Reemplazar transistores de unión bipolar NPN por transistores MOSFET tipo N de 3,3 y 5 v para reducir consumo de corriente y disipación térmica.

Finalmente, enmarcado en el espíritu del desarrollo libre y abierto a la comunidad, y de los valores fundacionales de la Universidad Nacional de Quilmes[28], se *sugiere* y *alienta* el estudio y replicación de este proyecto, así como el intercambio de ideas que de él surjan por parte de toda la comunidad universitaria, DIY y desarrolladora en general toda vez que tales actividades se inscriban en el espíritu y valores mencionados.

Juan Mariano Ramos

[28] "La Universidad Nacional de Quilmes tiene por misión *la producción, enseñanza y difusión de conocimientos del más alto nivel en un clima de igualdad y pluralidad*", "La ciencia, en su diversidad, es una constante del saber y su perfeccionamiento el objetivo de la investigación, que redunda en el *bienestar colectivo de la sociedad*", "La formación de profesionales, técnicos e investigadores involucra la *elaboración, promoción, desarrollo y difusión de la cultura, el arte y la ciencia*" (www.unq.edu.ar/secciones/41-la-universidad)

5. Bibliografía

Adler, Samuel (2006): "*El estudio de la orquestación*", Barcelona (España), Idea Books.

Anderson, Timothy (2012): "*AIRduino: An Inexpensive DIY MIDI Wind Controller*" (Artículo en ICMC2012), Missoula (EE.UU.).

(Aodyo SAS) (2016): "*Sylphyo User Guide*" (Manual de usuario), Villeneuve d'Ascq (Francia).

Apple Inc. (2014): "*Apple Bluetooth Low Energy MIDI Specification*" (Documento PDF), Cupertino (EE.UU.).

(Arduino Guide): "*Arduino Xbee, primeros pasos*", (documento PDF).

(Digi International): "*XBee/XBee-PRO RF Modules"* (hoja de datos).

Dubief, Fletcher y Meyers (2013): "*Mechanical Engineering Lab II – Fluids Laboratory 2: Venturi Effect"* Vermont, EE.UU. (Texto académico universitario).

Fletcher, N. H. y Tarnopolsky, A. (1998): "*Blowing pressure, power, and spectrum in trumpet playing*" (artículo en "*1999 Journals - Acoustical Society of America*"), Canberra (Australia).

Ganssle, Jack G. (2004): "*A guide to debouncing*", Baltimore (EE.UU.), The Ganssle Group.

Groover, Mikel P. (1997): "*Fundamentos de la manufactura moderna*", Naucalpan de Juárez (México), Prentice-Hall Hispanoamericana S.A.

(Interlink Electronics): "*FSR Integration Guide & Evaluation Parts Catalog With Suggested Electrical Interfaces*" (Documento PDF).

Kon, F. y Posse Lago N. (2004): "*The quest for low latency"* (en "Proceedings of the 30th International Computer Music Conference" pp. 33-36). Miami, EE.UU.

Lady Ada (Adafruit Industries) (2015): "*Xbee Radios*" (documento PDF), Adafruit Industries.

Lady Ada (Adafruit Industries) (2016): "*Adafruit MPR121 12-Key Capacitive Touch Sensor Breakout Tutorial*" (Documento PDF), Adafruit Industries.

Lefteri, Chris (2008): "*Así se hace: técnicas de fabricación para diseño de producto*", Barcelona (España), BLUME.

(Microchip Technology Inc.) (2008): "*3V Tips 'n Tricks*" (documento PDF).

(NASA) (2010): *"Principles of flight – Bernoulli's Principle"*, EE.UU. (Texto académico escolar).

(PAAB Tekno Trading AB): "*Introduction to Pitot Tube Flow Measurement*" (descripciones técnicas), Säffle (Suecia).

Panasonic Corporation (2007): "*Lithium Ion Batteries Technical Handbook*" (documento PDF).

Serway, Raymond A. (1996): *"Bernoulli's Principle"* (en "Physics for Scientists and Engineers, Fourth Edition, Vol.1" pp. 422-434), Philadelphia, EE.UU. Saunders College Publishing.

The MIDI Manufacturers Association (1995): "*MIDI 1.0 Detailed Specification*", Los Angeles (EE.UU.), The MIDI Manufacturers Association.

Anexos

Anexo I - Tabla de implementación MIDI

Función	Transmitido	Observaciones
Número de nota	**SI** (11 – 73)	Según transposición y portavoces
Velocity Note On	**SI** (1 – 127)	
Velocity Note Off	**NO**	
Pitch Bend	**SI**	Dividido en + y -
Control Change	**SI** (1, 2, 5, 7, 11)	Modulation, breath, portamento, volume, expression
Program Change	**SI** (0 – 127)	
CC Auxiliares	**SI** (120, 121, 123)	Función pánico (all sound off, reset controllers, all notes off)

Se utiliza el canal 1 exclusivamente.

El dispositivo no cuenta con MIDI-IN, por lo que no recibe datos.

Anexo II - Encuesta realizada a intérpretes de aerófonos sobre las embocaduras

1) Indicar tu instrumento principal (puede ser más de uno si se trata de otros aerófonos).

2) ¿En cuánto tiempo mínimo creés que alguien sin experiencia alguna lograría dominar la embocadura (y respiración) de tu instrumento lo suficiente como para producir de manera estable y afinada una escala mayor elemental?

3) ¿Qué dificultades te produjo la embocadura durante tus primeros acercamientos al instrumento?

4) ¿Qué instrumento te resultó más fácil o cómodo de tocar respecto a su embocadura?

Instrumento	Tiempo mínimo	Dificultades	Instrumento fácil
Clarinete	1 semana	"Es muy fácil hacerse malos hábitos muy rápido que después no los corregís más. Depende mucho de la boquilla y caña, y si elegís una caña cualquiera termina hasta siendo peor."	"Saxo, porque es la boquilla más cómoda y fácil de aprender."
Flauta traversa	6 a 12 meses	"La inclinación varía afinación, caudal de aire, posición correcta de labios, posición de manos y dedos."	"Saxo alto. La afinación no varía tanto como en la flauta, es más estable."
Trompeta	2 a 3 meses	"Lastimaduras en los labios e hiperventilación."	"Saxo."
Flauta dulce	1 mes	"Ninguna."	"Flauta dulce. Es un silbato, hasta un niño puede hacerla sonar."
Saxofón Alto	2 meses	"Cansancio muscular, lesiones labiales y afinación."	"el saxo alto me pareció más accesible."
Quena Shakuhachi Bansuri	1 mes	"Insuficiencia respiratoria e hiperventilacion, no siempre obtener sonido."	"bansuri, porque una vez que encontrás la embocadura, funciona. La quena y el shakuhachi son más difíciles porque el medio círculo que le falta a la embocadura lo tiene que hacer uno con los labios."
Saxofón Alto Flauta traversa Flauta dulce Armónica Sikus	2 meses	"Apoyo, lesiones labiales, posición, dirección del aire, hiperventilación, regulación del aire a la emisión."	"En orden de más fácil a menos fácil 1.- Flauta dulce 2.- Armónica 3.- Sikus 4.- Flauta traversa 5.- Saxofón"
Fagot Saxofón	1 mes	"Me trajo problemas de posición, dolores musculares y lastimaduras en los labios.	"Todos tienen su dificultad, con embocaduras

Clarinete Flauta Oboe Trompeta		Hasta poder tomar conciencia de la embocadura correcta.”	totalmente diferentes y con sus dificultades y facilidades. Personalmente me resultan más sencillos los de lengüeta simple.”
Saxofón alto	20 días	“Dolores en los labios principalmente, y después el ir descubriendo la influencia en el sonido del grosor de las cañas, la longitud de la boquilla misma, la altura del mentón y caudal del aire. Por la embocadura pasa el espíritu de lo que se va a transmitir. El carisma, las pausas, los énfasis y la convicción.“	“De los pocos vientos de distinta embocadura que tuve el lujo de tocar, el más sencillo para mi criterio es el saxo. Porque el caudal de aire entra sola y directamente por la hendija. En instrumentos como flauta traversa, siento que es poco es aire a utilizar y mucho que se desperdicia, y en instrumentos como trompeta, es muy difícil lograr la correcta afinación.”
Oboe	1 año	“Dolor en el labio, exceso de aire, afinación alta por apretar la caña con los labios.”	“Saxo.”
Flauta traversa	2 meses	“La inclusión y producción del sonido, la estabilidad del sonido, mareo por la forma en la que hay que soplar y la intensidad.”	“La de la flauta soprano, o las de pico, son fáciles, el aire va directo, alcanza mucho más y no se pierde, y no he intentado tiempo suficiente con otras embocaduras.”
Flauta traversa	6 a 12 meses	“Afinación.”	“Flauta traversa porque es lo único que toco.”
Quena Armónica Saxofón alto	2 meses	“Relativamente tres: mala afinación, dolor muscular por tensión en manos y posiciones incómodas.”	“La armónica y el saxo alto, dicen que el saxo es de los más complejos pero como toque de chico me fue natural y la armónica fue algo muy perspectivo al igual que las quenas.”
Trompeta	1 año	“La afinación”	“El saxo.”

Anexo III - Código fuente

A continuación se expone el código fuente en lenguaje Arduino. Su formato ha sido ligeramente modificado para adecuarlo a esta presentación. El proyecto se compone de 3 archivos: "*proyecto_general.ino*", "*instrumentos.h*" y "*digitaciones.h*".

proyecto_general.ino

```
#include <LCD5110_Graph.h>
#include "instrumentos.h"
#include "digitaciones.h"
#include <avr/pgmspace.h>
#include <Metro.h>
#include <Wire.h>
#include "Adafruit_MPR121.h"
#include <MIDI.h>
#include <Bounce2.h>
#include <EEPROM.h>

MIDI_CREATE_DEFAULT_INSTANCE();
Adafruit_MPR121 cap = Adafruit_MPR121();
LCD5110 myGLCD(4,8,5,6,7);  // SCK-MOSI-DC-RST-CS
extern uint8_t SmallFont[];
Metro temporizadorLuz = Metro(10000);
Metro temporizadorBat = Metro(5000);

//  variables de configuracion
uint8_t funcionSeleccionada = 0;  //  posicion del cursor en pantalla
uint8_t instrumentoActual = 65;    //  instrumento seleccionado (inicia Saxo
Alto)
uint8_t modoSensorA = 3;    //  mensaje de CC del sensor A (pulgar)
uint8_t modoSensorB = 2;    //  mensaje de CC del sensor B (boquilla)
uint8_t sensibilidad = 9;   //  sensibilidad del sensor de aire
uint8_t modoExpresion = 5;  //  mensaje de expresion para el sensor de aire
int8_t transposicion = 0;   //  este valor va de -24 a +24
boolean alteracion = false;   //   true = #    false = b
int8_t valorAlteracion = -1;   //   valor a añadir para la llave #/b (inicia en
b)
#define BEMOL -1
#define SOSTENIDO 1
boolean Xbee = true;   //  encendido del Xbee  true o false
boolean guardarCambios = false;   //  indica si deben guardarse cambios en
EEPROM

//  variables de medicion de bateria
#define PROMEDIAR_MUESTRAS 10
#define BATERIA_LLENA 850
#define BATERIA_VACIA 520
uint16_t muestras[10];
uint16_t promedio = 0;

#define CANAL_MIDI 1
const uint8_t DELAY_GRILLA = 5;
instrTipo listaInstrumentos;
```

```
uint8_t notaActual = 0;

//  definiciones para los botones
#define BOTON_MENOS 9
#define BOTON_MAS 10
#define BOTON_FUN 11
#define BOTON_PANICO 12
#define TIEMPO_REPETICION 150
unsigned long contadorPresionado;

//  definiciones de expresion de sensores
#define DESACTIVADO 0
#define PITCH_BAJA 1
#define PITCH_SUBE 2
#define MODULACION 3
#define PORTAMENTO 4
#define VOLUMEN 5
#define BREATH 6
#define EXPRESSION 7
#define SIN_DINAMICA 8

//  variables de sensor MPX5010
//  Minimo: 47  (piso)
//  Minimo: 60  (soplo minimo)
//  Maximo: 540 (esfuerzo mediano)
//  Maximo: 650 (muy esforzado)
#define UMBRAL_NOTE_ON 60
#define MAXIMO_SOPLO 400
#define DELAY_VELOCITY 2
boolean notaSonando = false;
uint16_t valorSensor;
uint8_t ccMidiExpresion = 7;
uint16_t velocityMuestra = 0;

//  variables de sensores A B
#define UMBRAL_A 50
#define MAXIMO_A 900
boolean activoA = false;
int16_t valorA = 0;
uint8_t ccMidiSensorA = 1;
#define UMBRAL_B 200
#define MAXIMO_B 850
boolean activoB = false;
int16_t valorB = 0;
uint8_t ccMidiSensorB = 1;

// Botones
Bounce Bmas = Bounce();
Bounce Bmenos = Bounce();
Bounce Bfun = Bounce();
Bounce Bpanico = Bounce();


//--------------------------------------------------------------------------
void setup() {

  delay(1000);      //  retardo inicial para estabilizar electronica

  pinMode(0, INPUT);     //  Monitor Xbee
  pinMode(3, OUTPUT);    //  Backlight
  pinMode(13, OUTPUT);   //  Encendido Xbee
  digitalWrite(13, HIGH);  // activa el modulo Xbee
```

```
  // inicia la botonera de configuracion
  pinMode(BOTON_MAS,INPUT_PULLUP);
  pinMode(BOTON_MENOS,INPUT_PULLUP);
  pinMode(BOTON_FUN,INPUT_PULLUP);
  pinMode(BOTON_PANICO,INPUT_PULLUP);
  Bmas.attach(BOTON_MAS);
  Bmas.interval(5);
  Bmenos.attach(BOTON_MENOS);
  Bmenos.interval(5);
  Bfun.attach(BOTON_FUN);
  Bfun.interval(5);
  Bpanico.attach(BOTON_PANICO);
  Bpanico.interval(5);

  promediar();    //  toma una primer lectura de la bateria

  myGLCD.InitLCD(65);   // inicia la pantalla con contraste 65
  myGLCD.setFont(SmallFont);

  cap.begin(0x5A);    //  valores de inicializacion del sensor mpr121
  cap.setThresholds(12, 6);
  cap.setDebounce(0x33);

  MIDI.begin();
  panico();
  cargarParametros();
  MIDI.sendProgramChange (instrumentoActual, CANAL_MIDI);   // inicia en ultimo
instrumento

}


//--------------------------------------------------------------------------
void loop() {

  valorSensor = analogRead(0);   //  toma el valor del sensor MPX5010
  calcularAB();   //  calcula y envia mensajes CC segun corresponda
  ejecutarAB();
  if(temporizadorBat.check() == 1) promediar();   //  promedia la lectura de
bateria cada X tiempo
  actualizarBotones();    //  lee y administra el estado de los botones de
configuracion
  imprimeLCD();           //  establece y muestra en pantalla toda la
informacion
  sonarNotas();           //  administra la produccion de notas MIDI
  apagarLuz();            //  administra la iluminacion de la pantalla
  delay(DELAY_GRILLA - DELAY_VELOCITY); // aplica un ligero retardo para
controlar el volumen de datos

}


//--------------------------------------------------------------------------
// Funciones para guardar y recuperar parametros de memoria EEPROM
void guardarParametros() {
    EEPROM.write(0, instrumentoActual);
    EEPROM.write(1, modoSensorA);
    EEPROM.write(2, modoSensorB);
    EEPROM.write(3, sensibilidad);
    EEPROM.write(4, modoExpresion);
    EEPROM.write(5, transposicion + 100);   //  +100 para adecuar numeros
negativos
    EEPROM.write(6, alteracion);
```

```
    EEPROM.write(7, Xbee);
}


void cargarParametros() {
    instrumentoActual = EEPROM.read(0);
    modoSensorA = EEPROM.read(1);
    modoSensorB = EEPROM.read(2);
    sensibilidad = EEPROM.read(3);
    modoExpresion = EEPROM.read(4);
    transposicion = EEPROM.read(5) - 100;   //  -100 para adecuar numeros
negativos
    alteracion = EEPROM.read(6);
    Xbee = EEPROM.read(7);
}



//--------------------------------------------------------------------------
void calcularAB() {

    switch (modoSensorA) {        // determina el valor a asignar cada CC segun
el modo escogido de A
        case DESACTIVADO:
            valorA = 0;
            ccMidiSensorA = 1;
            break;

        case PITCH_BAJA:
            valorA = constrain(analogRead(1),UMBRAL_A, MAXIMO_A);
            valorA = map(valorA, UMBRAL_A, MAXIMO_A, 0, -8192);
            ccMidiSensorA = 1;
            break;

        case PITCH_SUBE:
            valorA = constrain(analogRead(1),UMBRAL_A, MAXIMO_A);
            valorA = map(valorA, UMBRAL_A, MAXIMO_A, 0, 8191);
            ccMidiSensorA = 1;
            break;

        case MODULACION:
            valorA = constrain(analogRead(1),UMBRAL_A, MAXIMO_A);
            valorA = map(valorA, UMBRAL_A, MAXIMO_A, 0, 127);
            ccMidiSensorA = 1;
            break;

        case PORTAMENTO:
            valorA = constrain(analogRead(1),UMBRAL_A, MAXIMO_A);
            valorA = map(valorA, UMBRAL_A, MAXIMO_A, 0, 127);
            ccMidiSensorA = 5;
            break;
    }

    switch (modoSensorB) {        // determina el valor a asignar cada CC segun
el modo escogido de B
        case DESACTIVADO:
            valorB = 0;
            ccMidiSensorB = 1;
            break;

        case PITCH_BAJA:
            valorB = constrain(analogRead(2),UMBRAL_B, MAXIMO_B);
            valorB = map(valorB, UMBRAL_B, MAXIMO_B, 0, -8192);
```

```
            ccMidiSensorB = 1;
            break;

        case PITCH_SUBE:
            valorB = constrain(analogRead(2),UMBRAL_B, MAXIMO_B);
            valorB = map(valorB, UMBRAL_B, MAXIMO_B, 0, 8191);
            ccMidiSensorB = 1;
            break;

        case MODULACION:
            valorB = constrain(analogRead(2),UMBRAL_B, MAXIMO_B);
            valorB = map(valorB, UMBRAL_B, MAXIMO_B, 0, 127);
            ccMidiSensorB = 1;
            break;

        case PORTAMENTO:
            valorB = constrain(analogRead(2),UMBRAL_B, MAXIMO_B);
            valorB = map(valorB, UMBRAL_B, MAXIMO_B, 0, 127);
            ccMidiSensorB = 5;
            break;
    }

}


//--------------------------------------------------------------------------
void ejecutarAB() {


// envia los mensajes CC correspondientes para el sensor A  ------------------

      if (valorA != 0) {

          switch (modoSensorA) {
              case DESACTIVADO:
                  break;

              case PITCH_BAJA:
                  activoA = true;
                  MIDI.sendPitchBend(valorA, CANAL_MIDI);
                  break;

              case PITCH_SUBE:
                  activoA = true;
                  MIDI.sendPitchBend(valorA, CANAL_MIDI);
                  break;

              case MODULACION:
                  activoA = true;
                  MIDI.sendControlChange(ccMidiSensorA, valorA, CANAL_MIDI);
                  break;

              case PORTAMENTO:
                  if ((!activoA) && (modoSensorA == PORTAMENTO)) {    // activa
el portamento una vez
                      MIDI.sendControlChange(65, 127, CANAL_MIDI);
                  }
                  activoA = true;
                  MIDI.sendControlChange(ccMidiSensorA, valorA, CANAL_MIDI);
                  break;
          }
```

```
      } else {                      // deja en cero los CC, y se asegura que
solo ocurra una vez
          if ((activoA) && ( (modoSensorA == PITCH_BAJA) || (modoSensorA ==
PITCH_SUBE) )  ) {
              MIDI.sendPitchBend(0, CANAL_MIDI);
              activoA = false;
          }
          if ((activoA) && (modoSensorA == MODULACION)) {
              MIDI.sendControlChange(ccMidiSensorA, 0, CANAL_MIDI);
              activoA = false;
          }
          if ((activoA) && (modoSensorA == PORTAMENTO)) {
              MIDI.sendControlChange(ccMidiSensorA, 0, CANAL_MIDI);
              MIDI.sendControlChange(65, 0, CANAL_MIDI);
              activoA = false;
          }
      }


// envia los mensajes CC correspondientes para el sensor B  -------------------

      if (valorB != 0) {

          switch (modoSensorB) {
              case DESACTIVADO:
                  break;

              case PITCH_BAJA:
                  activoB = true;
                  MIDI.sendPitchBend(valorB, CANAL_MIDI);
                  break;

              case PITCH_SUBE:
                  activoB = true;
                  MIDI.sendPitchBend(valorB, CANAL_MIDI);
                  break;

              case MODULACION:
                  activoB = true;
                  MIDI.sendControlChange(ccMidiSensorB, valorB, CANAL_MIDI);
                  break;

              case PORTAMENTO:
                  if ((!activoB) && (modoSensorB == PORTAMENTO)) {    // activa
el portamento una vez
                      MIDI.sendControlChange(65, 127, CANAL_MIDI);
                  }
                  activoB = true;
                  MIDI.sendControlChange(ccMidiSensorB, valorB, CANAL_MIDI);
                  break;
          }

      } else {                      // deja en cero los CC, y se asegura que
solo ocurra una vez
          if ((activoB) && ( (modoSensorB == PITCH_BAJA) || (modoSensorB ==
PITCH_SUBE) )  ) {
              MIDI.sendPitchBend(0, CANAL_MIDI);
              activoB = false;
          }
          if ((activoB) && (modoSensorB == MODULACION)) {
              MIDI.sendControlChange(ccMidiSensorB, 0, CANAL_MIDI);
              activoB = false;
          }
```

```
            if ((activoB) && (modoSensorB == PORTAMENTO)) {
                MIDI.sendControlChange(ccMidiSensorB, 0, CANAL_MIDI);
                MIDI.sendControlChange(65, 0, CANAL_MIDI);
                activoB = false;
            }
        }

}


//--------------------------------------------------------------------------
void imprimeLCD() {
  myGLCD.clrScr();

  //  Instrumento   ------------------------------------------------------
  myGLCD.print(":", 18 , 0);
  myGLCD.printNumI(instrumentoActual + 1, 0, 0, 3, '0');
  if (funcionSeleccionada == 0) {myGLCD.invertText(true);}
  myGLCD.print(txtInstr(), 24, 0);
  myGLCD.invertText(false);

  //  Sensores A y B   ---------------------------------------------------
  myGLCD.print(":", 6, 8);
  myGLCD.print(":", 48, 8);
  myGLCD.print("A", 0, 8);
  if (funcionSeleccionada == 1) {myGLCD.invertText(true);}
  //  myGLCD.printNumI(analogRead(2) / 8, 18, 8);
  myGLCD.print(textoParametros(modoSensorA), 12, 8);
  myGLCD.invertText(false);
  myGLCD.print("B", 42, 8);
  if (funcionSeleccionada == 2) {myGLCD.invertText(true);}
  //  myGLCD.printNumI(nroB, 60, 8);
  myGLCD.print(textoParametros(modoSensorB), 54, 8);
  myGLCD.invertText(false);

  //  Sensibilidad y Expresion   -----------------------------------------
  myGLCD.print("Sn:", LEFT, 16);
  if (funcionSeleccionada == 3) {myGLCD.invertText(true);}
  myGLCD.printNumI(sensibilidad, 18, 16);
  myGLCD.invertText(false);

  myGLCD.print("Ex:", 30, 16);
  if (funcionSeleccionada == 4) {myGLCD.invertText(true);}
  myGLCD.print(textoParametros(modoExpresion), 48, 16);
  myGLCD.invertText(false);

  //  Transposicion y #/b   ----------------------------------------------
  myGLCD.print("Trsp:", LEFT, 24);
  if (funcionSeleccionada == 5) {myGLCD.invertText(true);}
  myGLCD.printNumI(abs(transposicion), 36, 24);
  if (transposicion > 0) {myGLCD.print("+", 30, 24);}   //  esto es para forzar
que aparezca el "+" cuando el valor es positivo
  if (transposicion < 0) {myGLCD.print("-", 30, 24);}
  myGLCD.invertText(false);

  myGLCD.print("ST:", 54, 24);
  if (funcionSeleccionada == 6) {myGLCD.invertText(true);}
  if (alteracion) {myGLCD.print("#", 72, 24);}
  if (!alteracion) {myGLCD.print("b", 72, 24);}
  myGLCD.invertText(false);

  //  myGLCD.print(String(4096 + cap.touched(), BIN), LEFT, 24); // Esto
imprimia el mapa de llaves
```

```
  //  myGLCD.printNumI(obtenerNota(), LEFT, 24);
  //  myGLCD.printNumI(digitalRead(0), CENTER, 24);

  //  Xbee Estado y Encendido   ------------------------------------------
  myGLCD.print("Xbee:", LEFT, 32);
  if (funcionSeleccionada == 7) {myGLCD.invertText(true);}
  if (Xbee) {myGLCD.print(" SI ", 30, 32);}
  if (!Xbee) {myGLCD.print(" NO ", 30, 32);}
  myGLCD.invertText(false);
  if ((digitalRead(0) == HIGH) & Xbee) {myGLCD.print("Ok!", 66, 32);} //
Imprimir el parpadeo de encendido

  //  Bateria   ----------------------------------------------------------
    dibujarBateria();
  myGLCD.print("Bat:", LEFT, 40);

  //  Opciones de "debug"
  //  myGLCD.printNumI(promedio, 30, 40);
  //  myGLCD.printNumI(valorB, RIGHT, 40);
  //  myGLCD.printNumI(valorA, 30, 40);


  myGLCD.update();  // dibuja todo lo establecido

}

//--------------------------------------------------------------------------
String txtInstr() {
  memcpy_P (&listaInstrumentos, &nombreInstrumento [min(instrumentoActual,
127)], sizeof listaInstrumentos);
  return listaInstrumentos.nombreInstrumento;
}


//--------------------------------------------------------------------------
void actualizarBotones() {

  //  Boton Funcion   ----------------------------------------------------
    Bfun.update();
    if ( Bfun.fell() ) {
        funcionSeleccionada++;
        encenderLuz();
        contadorPresionado = millis();
    }
    if  ( Bfun.read() == 0 ) {      //  Repetir si sigue presionado
      if ( millis() - contadorPresionado >= TIEMPO_REPETICION * 2) {
          contadorPresionado = millis();
          funcionSeleccionada++;
          encenderLuz();
      }
    }
    if (funcionSeleccionada >= 8) {funcionSeleccionada = 0;}

  //  Boton Panico   -----------------------------------------------------
    Bpanico.update();
    if ( Bpanico.fell() ) {
        panico();
        encenderLuz();
    }

  //  Boton +   ----------------------------------------------------------
    Bmas.update();
    if ( Bmas.fell() ) {
```

```
        cambiarParametro(1);
        encenderLuz();
        contadorPresionado = millis();
    }
    if  ( Bmas.read() == 0 ) {      //  Repetir si sigue presionado
      if ( millis() - contadorPresionado >= TIEMPO_REPETICION ) {
          contadorPresionado = millis();
          cambiarParametro(1);
          encenderLuz();
      }
    }

  //  Boton -   ----------------------------------------------------------------
    Bmenos.update();
    if ( Bmenos.fell() ) {
        cambiarParametro(-1);
        encenderLuz();
        contadorPresionado = millis();
    }
    if  ( Bmenos.read() == 0 ) {      //  Repetir si sigue presionado
      if ( millis() - contadorPresionado >= TIEMPO_REPETICION ) {
          contadorPresionado = millis();
          cambiarParametro(-1);
          encenderLuz();
      }
    }

}


//--------------------------------------------------------------------------------
void dibujarBateria() {

    const uint8_t posY = 41;
    uint16_t proporcionBateria = 0;      //  maximo 54
    proporcionBateria = constrain(promedio, BATERIA_VACIA, BATERIA_LLENA);

    proporcionBateria = map(proporcionBateria, BATERIA_VACIA, BATERIA_LLENA, 0,
54);

    myGLCD.drawLine(26, posY + 1, 26, posY + 5);    //  izquierda
    myGLCD.drawLine(81, posY + 1, 81, posY + 5);    //  derecha
    myGLCD.drawLine(27, posY, 81, posY);            //  arriba
    myGLCD.drawLine(27, posY + 5, 81, posY + 5);    //  abajo

    for (int i=0; i <= 3; i++){    //  dibuja el nivel de la bateria
        myGLCD.drawLine(27, posY + 1 + i, 27 + proporcionBateria, posY + 1 + i);
    }

    if (proporcionBateria <= 6) myGLCD.print("Cargar!", RIGHT, 40);   //  da
aviso de bateria baja

}


//--------------------------------------------------------------------------------
void promediar(){

    int maximo = 0;
    int minimo = 1023;

    for (int i=0; i <= (PROMEDIAR_MUESTRAS - 1); i++){    //  llena la matriz
con X muestras
```

```
        muestras[i] = analogRead(3);
    }

    for (int i=0; i < PROMEDIAR_MUESTRAS; i++) {    //  encuentra el valor maximo
        if (muestras[i]>maximo) {
          maximo = muestras[i];
        }
    }

    for (int i=0; i < PROMEDIAR_MUESTRAS; i++) {    //  encuentra el valor minimo
        if (muestras[i] < minimo) {
        minimo = muestras[i];
        }
    }

    promedio = (minimo + maximo) / 2;

}


//--------------------------------------------------------------------------------
//  Esta funcion es la encargada de producir las notas MIDI segun el soplo y
digitacion

void sonarNotas() {

    uint8_t notaBoton = obtenerNota();
    uint16_t nuevoValorSensor = 0;

    switch (modoExpresion) {    // determina el nro de CC segun el modo escogido
de expresion
        case VOLUMEN:
          ccMidiExpresion = 7;
          break;
        case BREATH:
          ccMidiExpresion = 2;
          break;
        case EXPRESSION:
          ccMidiExpresion = 11;
          break;
    }

    if (valorSensor > UMBRAL_NOTE_ON) {
        if (notaSonando) {                                      // Si ya SONABA
solo usar el sensor para aplicar un CC
            if (modoExpresion != SIN_DINAMICA)
MIDI.sendControlChange(ccMidiExpresion, min((valorSensor - UMBRAL_NOTE_ON),
127), 1); // envia el mensaje CC

        } else {                                                // Si NO SONABA y
el valor del sensor supera el umbral, enciende la nota
            delay(DELAY_VELOCITY);  //  aplica un leve retardo para mitigar el
efecto rebote y ruido en la señal y determinar velocity

            nuevoValorSensor = analogRead(0);
            if (nuevoValorSensor < UMBRAL_NOTE_ON) return;  //  si
transcurrido el tiempo X, el valor es bajo, se ignora la señal.

            velocityMuestra = int((float(nuevoValorSensor) /
float(MAXIMO_SOPLO * 0.5)) * 127);   //  calcula el velocity
            velocityMuestra = velocityMuestra * (1.0 + ((8.0 - sensibilidad -
1.0) / 8.0));   //  aplica la sensibilidad seleccionada
```

```
        //  (sens9 = 1.0   sens1 = 2.0)
              velocityMuestra = min(velocityMuestra,
127);                              //  limita el valor a 127
              if (sensibilidad == 0) velocityMuestra =
127;                              //  fija velocity a 127 si sens0

              notaActual = notaBoton;
              MIDI.sendNoteOn(notaActual, velocityMuestra, CANAL_MIDI);
              notaSonando = true;

              if (guardarCambios) {
                  guardarParametros();
                  guardarCambios = false;
              }

        }

  } else {                                                          // Si SONABA y
el valor del sensor cae bajo el umbral, apaga la nota
        if (notaSonando) {
          MIDI.sendNoteOff(notaActual, 0, CANAL_MIDI);
          notaSonando = false;
        } else {
          // no hacer nada... nota apagada
        }
  }

  // Cambio de notas segun cambio de llaves (legato)
  if ((notaSonando)&&(notaActual != notaBoton)) {

    velocityMuestra = int((float(valorSensor) / float(MAXIMO_SOPLO * 0.5)) *
127);   //  calcula el velocity
    velocityMuestra = velocityMuestra * (1.0 + ((8.0 - sensibilidad - 1.0) /
8.0));   //  aplica la sensibilidad seleccionada
                                                                          //
  (sens9 = 1.0   sens1 = 2.0)
    velocityMuestra = min(velocityMuestra,
127);                              //  limita el valor a 127
    if (sensibilidad == 0) velocityMuestra =
127;                              //  fija velocity a 127 si sens0

    MIDI.sendNoteOn(notaBoton, velocityMuestra, CANAL_MIDI);
    MIDI.sendNoteOff(notaActual, 0, CANAL_MIDI);
    notaActual = notaBoton;

  }

}


//--------------------------------------------------------------------------------
uint8_t obtenerNota() {                     // LEE las llaves y devuelve una
nota midi segun la tabla de digitaciones
    uint16_t tactil = cap.touched();
    tactil = tactil & 0b0000000001111111;   // se enmascaran las llaves que no
son diatonicas -- AND (&) deja lo que esta SOLO si hay un 1
    uint8_t devuelve = mapa[0].nota_midi;   // valor por defecto = llave mas
grave.. (ninguno presionado)

    for (uint8_t i=0; i < CANTIDAD_DIGITACIONES; i++) {
        if (tactil == mapa[i].llaves) {
          devuelve = mapa[i].nota_midi;
```

```
          break;
        }
    }

    switch (cap.touched() & 0b0000110000000000) {   // verificar portavoces
        case 0b0:
          break;
        case 0b100000000000:
          devuelve = devuelve + 12;
          break;
        case 0b110000000000:
          devuelve = devuelve + 24;
          break;
        case 0b010000000000:
          devuelve = devuelve + 36;
          break;
    }


     if (cap.touched() & (1 << 7)) {     // aplica la alteracion #/b si esta
tocada la llave
      devuelve = devuelve + valorAlteracion;
     }

     return devuelve + transposicion;    // aplica la transposicion elegida
}


//--------------------------------------------------------------------------------
void panico(){                  // Reinicia parametros MIDI
  MIDI.sendControlChange(120, 0, CANAL_MIDI);
  MIDI.sendControlChange(121, 0, CANAL_MIDI);
  MIDI.sendControlChange(123, 0, CANAL_MIDI);
}


//--------------------------------------------------------------------------------
void apagarLuz() {
    if (temporizadorLuz.check() == 1) {
      digitalWrite(3, LOW);
      temporizadorLuz.reset();
    }
}

void encenderLuz() {
      digitalWrite(3, HIGH);
      temporizadorLuz.reset();
}


//--------------------------------------------------------------------------------
void cambiarParametro(int8_t numero) {

    switch (funcionSeleccionada) {
      case 0:       //  Instrumento
          instrumentoActual = instrumentoActual + numero;
          panico();
          if (instrumentoActual == 128) {instrumentoActual = 0;}
          if (instrumentoActual == 255) {instrumentoActual = 127;}
          MIDI.sendProgramChange(instrumentoActual, CANAL_MIDI);
          break;

      case 1:       //  Sensor A
```

```
            modoSensorA = modoSensorA + numero;
            panico();
            if (modoSensorA == 5) {modoSensorA = 0;}
            if (modoSensorA == 255) {modoSensorA = 4;}

            valorA = 0;
            break;

        case 2:       //  Sensor B
            modoSensorB = modoSensorB + numero;
            panico();
            if (modoSensorB == 5) {modoSensorB = 0;}
            if (modoSensorB == 255) {modoSensorB = 4;}
            valorB = 0;
            break;

        case 3:       //  Sensibilidad
            sensibilidad = sensibilidad + numero;
            panico();
            if (sensibilidad == 10) {sensibilidad = 0;}
            if (sensibilidad == 255) {sensibilidad = 9;}
            break;

        case 4:       //  Expresion
            modoExpresion = modoExpresion + numero;
            panico();
            if (modoExpresion == 9) {modoExpresion = 5;}
            if (modoExpresion == 4) {modoExpresion = 8;}
            if (modoExpresion == SIN_DINAMICA) {MIDI.sendControlChange(7, 100,
CANAL_MIDI);}
            break;

        case 5:       //  Transposicion
            transposicion = transposicion + numero;
            panico();
            if (transposicion == 25) {transposicion = 24;}
            if (transposicion == -25) {transposicion = -24;}
            break;

        case 6:       //  #/b
            alteracion = !alteracion;
            if (alteracion) {valorAlteracion = SOSTENIDO;}
            if (!alteracion) {valorAlteracion = BEMOL;}
            panico();
            break;

        case 7:       //  Xbee
            panico();
            Xbee = !Xbee;
            digitalWrite(13, Xbee);
            panico();
            break;
        }

        guardarCambios = true;
}


//--------------------------------------------------------------------------
String textoParametros(uint8_t numeroParametro) {

    switch (numeroParametro) {
      case 0:       //  Desactivado
```

```
            return " -- ";
            break;

        case 1:       //  Baja Pitch
            return "PB-";
            break;

        case 2:       //  Sube Pitch
            return "PB+";
            break;

        case 3:       //  Modulacion
            return "Mod";
            break;

        case 4:       //  Portamento
            return "Port";
            break;

        case 5:       //  Volumen
            return "Volume";
            break;

        case 6:       //  Breath
            return "Breath";
            break;

        case 7:       //  Expression
            return "Expres";
            break;

        case 8:       //  Sin dinamica
            return " -- ";
            break;
    }
}
```

digitaciones.h

```
//-------------------------------------------------------------------------
// DIGITACIONES

const byte CANTIDAD_DIGITACIONES = 14;   // cantidad de digitaciones
const int transp_interna = -24;


struct mapa_entradas {
  uint8_t llaves;
  uint8_t nota_midi;
};

struct mapa_entradas mapa[CANTIDAD_DIGITACIONES] = {   // poner nro de
digitaciones

  // llave ABAJO > 0b1000001 < llave ARRIBA

  // DO GRAVE
  {0b1111111, 60 + transp_interna},
  {0b1000000, 60 + transp_interna},
  // RE
```

```
  {0b0111111, 62 + transp_interna},
  {0b0100000, 62 + transp_interna},
  // MI
  {0b0011111, 64 + transp_interna},
  {0b0010000, 64 + transp_interna},
  // FA
  {0b0001111, 65 + transp_interna},
  {0b0001000, 65 + transp_interna},
  // SOL
  {0b0000111, 67 + transp_interna},
  {0b0000100, 67 + transp_interna},
  // LA
  {0b0000011, 69 + transp_interna},
  {0b0000010, 69 + transp_interna},
  // SI
  {0b0000001, 71 + transp_interna},
  // DO AGUDO
  {0b0000000, 72 + transp_interna},

};
```

instrumentos.h

```
//--------------------------------------------------------------------------------
//  Lista de instrumentos GM para mostrar en pantalla
//  Se almacena en PROGMEM (memoria flash), para que no sature la RAM

typedef struct {
   char nombreInstrumento [11];  // maximo caracteres + 1
} instrTipo;

const instrTipo nombreInstrumento [128] PROGMEM = {

  "G. Piano",
  "B. Piano",
  "E. Piano",
  "H.T. Piano",
  "E. Piano 1",
  "E. Piano 2",
  "Harpsich.",
  "Clavi",
  "Celesta",
  "Glockensp.",
  "Music Box",
  "Vibraphone",
  "Marimba",
  "Xylophone",
  "Tub. Bells",
  "Dulcimer",
  "Db. Organ",
  "P. Organ",
  "Rock Organ",
  "Ch. Organ",
  "Reed Organ",
  "Accordion",
  "Harmonica",
  "Tango Ac.",
  "Gt. nylon",
  "Gt. steel",
  "Gt. jazz",
```

```
"Gt. clean",
"Gt. muted",
"Gt. Overd.",
"Gt. Dist.",
"Gt. Harm.",
"Ac. Bass",
"E. Bass F",
"E. Bass P",
"F.L. Bass",
"Slap B. 1",
"Slap B. 2",
"Synth B. 1",
"Synth B. 2",
"Violin",
"Viola",
"Cello",
"Contrabass",
"Trem. Str.",
"Pizz. Str.",
"Harp",
"Timpani",
"Str. En. 1",
"Str. En. 2",
"Synth S. 1",
"Synth S. 2",
"Choir Aahs",
"Voice Oohs",
"Synth Voc.",
"Orch. Hit",
"Trumpet",
"Trombone",
"Tuba",
"M. Trumpet",
"Fr. Horn",
"Brass Sec.",
"S. Brass 1",
"S. Brass 2",
"Sop. Sax",
"Alto Sax",
"Tenor Sax",
"Bar. Sax",
"Oboe",
"E. Horn",
"Bassoon",
"Clarinet",
"Piccolo",
"Flute",
"Recorder",
"Pan Flute",
"Blown Bot.",
"Shakuhachi",
"Whistle",
"Ocarina",
"Square",
"Sawtooth",
"Calliope",
"Chiff",
"Ccharang",
"Voice",
"Fifths",
"Bass + Ld.",
"New age",
"Warm",
```

```
    "Polysynth",
    "Choir",
    "Bowed",
    "Metallic",
    "Halo",
    "Sweep",
    "Rain",
    "Soundtrack",
    "Crystal",
    "Atmosphere",
    "Brightness",
    "Goblins",
    "Echoes",
    "Sci-Fi",
    "Sitar",
    "Banjo",
    "Shamisen",
    "Koto",
    "Kalimba",
    "Bag pipe",
    "Fiddle",
    "Shanai",
    "Tnk. Bell",
    "Agogo",
    "Stl. Drums",
    "Woodblock",
    "Taiko Dr.",
    "Melo. Tom",
    "Synth Drum",
    "Rev. Cymb.",
    "Gt. Fret",
    "Breath",
    "Seashore",
    "Bird Tweet",
    "Telephone",
    "Helicopter",
    "Applause",
    "Gunshot"

};
```

Anexo IV - Algunas imágenes del proceso de manufactura y otros

Boquilla prototipo 2

Pruebas I

Pruebas II

Portavoces y soporte

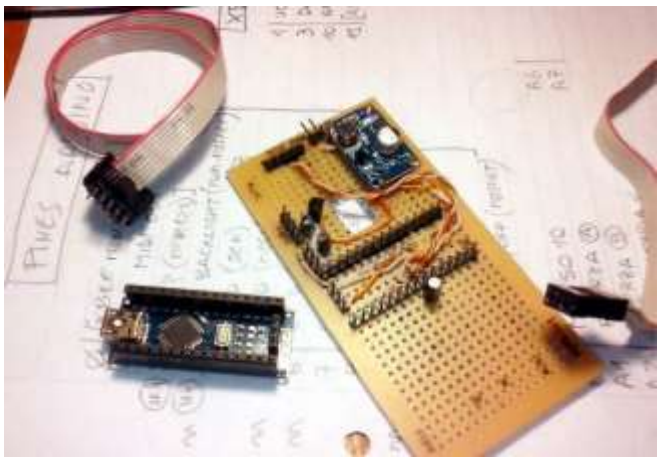
Botonera para pruebas

Etiquetado de módulos Xbee

Concepto de llaves laterales

Proceso paso a paso de construcción de llaves

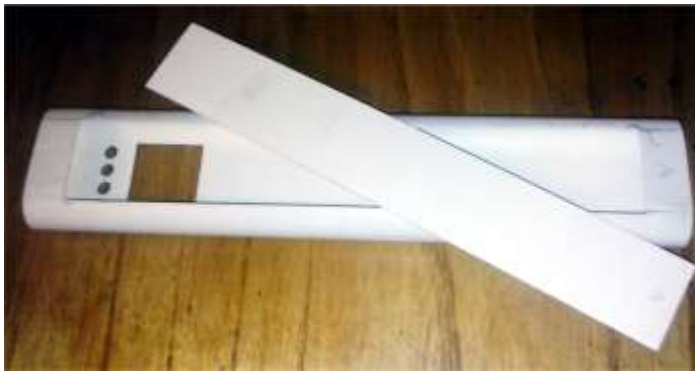
Confección de cables de interconexión



Construcción de placa base

Moldeado de carcasa I

Moldeado de carcasa II



Apertura de tapa y orificios

Anclaje portavoces

Anclaje llaves I

Anclaje llaves II


Universidad
Nacional
de Quilmes