Universidad Zaragoza
1542

Trabajo Fin de Grado

Sintetizador musical digital con controles inalámbricos mediante Arduino y Raspberry Pi.

Digital musical syntethizer with wireless controlers using Arduino and Raspberry Pi.

Autor:

Álvaro Hernando Carnicer

Director:

José Ramón Beltrán Blazquez

ESCUELA DE INGENIERÍA Y ARQUITECTRA

Departamento de Ingeniería Electrónica y Comunicaciones

2017

Sintetizador musical digital con controles inalámbricos mediante Arduino y Raspberry Pi.

RESUMEN

El trabajo está destinado a la creación de un sistema práctico y barato que permita la síntesis de audio digital en tiempo real. Para ello, nos servimos de las plataformas de Arduino y Raspberry Pi para realizar la monitorización de datos y la exposición de la información para el usuario gracias a un programa generado en Pure Data.

Dado que el objetivo principal es que sea un sistema modular, se prioriza la necesidad de que tenga elementos de comunicación inalámbrica a través de módulos de radio frecuencia.

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

D./Dª. Álvaro Hernando Carnicer,

con nº de DNI 73007901-J en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

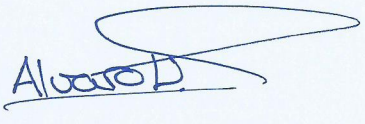
Declaro que el presente Trabajo de Fin de (Grado/Máster) Grado, (Título del Trabajo)

Sintetizador musical digital con controles inalámbricos mediante Arduino y Raspberry Pi.

Digital musical syntethizer with wireless controlers using Arduino and Raspberry Pi.

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, a 2 de febrero de 2017



Fdo: Álvaro Hernando Carnicer

Tabla de contenidos

1. Introducción

Un ordenador convenientemente programado es, virtualmente, capaz de producir cualquier sonido (in)imaginable, familiar o inaudito (aunque no sea capaz de hacerlo en tiempo real, podrá hacerlo en diferido). La única dificultad consiste en decirle cómo hacerlo o en describir, de alguna forma, el sonido que buscamos. Los nuevos métodos de síntesis sonora tienden por ello a ser más sofisticados en sus posibilidades, pero buscan interfaces de control más intuitivos.

En este capítulo se van a explicar tanto las razones por las que se ha realizado este proyecto como la estructura del mismo.

1.1. Motivación y objetivos.

La síntesis de audio y la mezcla a través de herramientas software han hecho que la producción de música sea más barata y accesible que nunca. Sin embargo, las interfaces de computadora típicas de ratón, teclado y monitor son poco adecuadas para el trabajo de controlar un mezclador o un sintetizador en tiempo real.

La principal motivación del proyecto es proveer al usuario de un entorno de síntesis de audio digital barato y de fácil uso, que sea configurable y que pueda correr en todas las plataformas. En este sentido, se pretende desarrollar una serie de módulos que se comuniquen con una interfaz gráfica de usuario. Adicionalmente, se ha desarrollado el software necesario para la síntesis correcta del audio a partir de las notas introducidas con un teclado MIDI.

Los objetivos de este proyecto son:

- Diseñar e implementar un sintetizador de audio digital con controladores inalámbricos modulares.
- Conseguir un sistema simple, que cualquier usuario pueda utilizar sin un entrenamiento previo.
- Diseñar un sistema de síntesis de audio digital que sea multiplataforma.

Una vez estudiadas las posibilidades de los tres elementos anteriores se diseñará un sistema modular basado en Arduino que se comunique inalámbricamente a través de radio frecuencia a una Raspberry Pi, que será la base de nuestro sintetizador. Para tener un control visual y generar la síntesis de audio se optará por el entorno de desarrollo Pure Data.

1.2. Estructura del documento.

En esta sección se detalla el contenido de cada uno de los capítulos que forman este documento, facilitando de esa manera la lectura del mismo.

En el segundo capítulo, se explicará en detalle el estado actual de las tecnologías utilizadas en el desarrollo del proyecto, así como una breve reseña histórica y la situación actual de los sintetizadores de audio digital.

El tercer capítulo describe la arquitectura del sistema, indicando los requisitos que tiene que satisfacer y el esquema general del sistema. Aquí se habla de la descripción básica de nuestro proyecto, explicando los requisitos del mismo y las decisiones del diseño a la hora de elegir cada componente.

El cuarto capítulo de este documento explica el desarrollo de la solución, así como los entornos de desarrollo utilizados. En el mismo, se ilustran los diseños tanto a nivel hardware como a nivel software del proyecto, explicando la funcionalidad de cada uno de los bloques que integran el sistema.

El quinto capítulo explica la planificación de todo el proyecto y el presupuesto para conseguir llevarlo a cabo.

En el capítulo final, el sexto, se explican las conclusiones obtenidas después de la presentación del proyecto, así como la definición de posibles futuras modificaciones de manera que mejoren el proyecto.

Posteriormente, y como contenido adicional en forma de anexo, se va a mostrar tanto el código utilizado como las imágenes del programa y de los diseños de las placas. De esta manera el lector podrá consultar las fuentes de las imágenes que se detallan durante la memoria.

2. Estado del arte

En este capítulo se va a explicar brevemente dónde está situado a nivel tecnológico este proyecto. Para ello, se va a mostrar una breve reseña histórica de la síntesis de audio digital, así como el "entorno" en el que se encuentra nuestro proyecto.

Por otro lado, se explicará brevemente el estado actual de las tecnologías utilizadas en este proyecto.

2.1. Historia de la síntesis de audio digital.

Para encontrar los primeros sintetizadores de sonido debemos remontarnos a 1906, año de invención del Telharmonium, mientras que en los años veinte surgen el Theremin y las Ondas Martenot. Estos primeros aparatos eran evidentemente analógicos, y utilizaban osciladores eléctricos como fuente sonora. Max V. Mathews, padre indiscutible de toda la síntesis digital, generó los primeros sonidos, en los laboratorios de IBM, en 1957.

En aquella época, la escasa potencia de aquellos ordenadores hacía totalmente inviable la comercialización de sintetizadores digitales, por lo que las dos décadas siguientes vieron la eclosión de los sintetizadores analógicos.

Hasta principios de los ochenta, la tecnología no estuvo preparada para dar el gran salto. En 1983 surge el DX7 de Yamaha, el primer sintetizador digital comercial, y en pocos años los sintetizadores digitales desbancarían a los analógicos.

Actualmente, existen ya en el mercado muchos programas para PC que emulan por software (y por una décima parte de su precio en hardware) las prestaciones de sintetizadores profesionales y que, lo único que requieren es el conversor D/A que se encuentra en cualquier tarjeta de sonido.

Un ordenador convenientemente programado es virtualmente capaz de producir cualquier sonido (in)imaginable, familiar o inaudito. La única dificultad consiste en decirle cómo hacerlo o en describir, de alguna forma, el sonido que buscamos.

2.2. ¿Qué es un sintetizador?

Como su nombre sugiere, este dispositivo sintetiza el sonido, es decir, lo genera a partir de la combinación de elementos simples (normalmente señales periódicas y funciones matemáticas) que no tienen por qué existir fuera de sus circuitos.

En realidad, los términos síntesis y, por extensión, sintetizador no son, hoy en día, plenamente acertados, pues gran parte de los instrumentos digitales actuales no sintetizan totalmente el sonido (no parten de cero) sino que utilizan, recombinan y modifican fragmentos almacenados en su memoria. A pesar de ello, y para simplificar, seremos "flexibles" y, de momento, pasaremos por alto este matiz.

Sea o no un "genuino" sintetizador, conviene recordar, por último, que todo generador digital de sonido oculta un ordenador (con su CPU, su memoria, su sistema operativo, etc.) en su interior. Los chips de los sintetizadores actuales son, de hecho, potentísimos DSPs (chips para procesado digital de señal) capaces de realizar decenas de millones de instrucciones por segundo.

2.3. Conceptos de síntesis digital.

Existen diferentes sistemas de síntesis digital de sonido, ciertas técnicas y procesos que permiten la modificación y el enriquecimiento del sonido generado inicialmente, se han ido extendiendo tanto que hoy son, en mayor o menor medida, aplicables a casi todos los sistemas. Pasamos a detallar algunos conceptos importantes que incorporan la mayoría de sintetizadores actuales [1], independientemente del método de síntesis que implementen, y que se han incluido en el proyecto:

- Generador de envolvente. La envolvente de un sonido hace referencia al cambio tanto del pitch como de la ganancia durante la duración del sonido. Un generador de envolvente tiene cuatro características principales:
  - Attack. El tiempo que toma la nota para alcanzar su punto más fuerte.
  - Decay. El tiempo que tarda después del ataque en alcanzar el sonido de mantenimiento.
  - Sustain. El volumen de la nota que se mantiene hasta que se libera la misma.
  - Release. El tiempo que tarda el sonido en desaparecer desde que dejamos de pulsar la nota.
- Oscilador de baja frecuencia. El término low frequency oscillation o LFO (traducido como "oscilación de baja frecuencia") se refiere a una señal de audio normalmente por debajo de 20 Hz que crea un ritmo palpitante en vez de un tono audible. LFO se suele referir al efecto de sonido específicamente utilizada en la producción de música electrónica.
- Oscilador. Un oscilador es el generador de señales básico en la música electrónica. Mediante la combinación, filtrado o modulación de ellos, se puede generar casi cualquier sonido imaginable. Existen osciladores de diferentes tipos: onda sinusoidal, onda cuadrada, onda triangular, etc. El nombre de cada oscilador hace referencia a su forma de onda. Diferentes formas de onda hacen diferentes sonidos.
- Frecuencia. Para crear sonido, cada oscilador toma una entrada numérica que representa una frecuencia en Hertz. Este número determina el número de veces que el oscilador hará su forma de onda durante un segundo.
- Síntesis aditiva. La síntesis aditiva consiste en combinar dos o más señales en una única señal. Si se combinan dos formas de onda cuyas frecuencias están muy próximas entre sí, los valores combinados de las dos ondas interfieren entre sí, provocando una modulación periódica del sonido. La frecuencia de esta modulación es igual a la diferencia de las dos frecuencias originales, en Hz. Esto se conoce como "frecuencia de golpeo" o "interferencia de fase". El sonido de dos osciladores ligeramente desajustados entre sí se utiliza a menudo para diferentes tipos de sonidos de música electrónica.

2.4. Proyectos similares.

Un proyecto similar a este, puede ser el desarrollado por “The House of Synthesis” [2]. En él se utiliza WhySynth para la síntesis del audio.

Cuando se trata de la mezcla, hay una variedad de controladores de mezcla dedicados disponibles. Controlar sintetizadores es un asunto diferente. Aunque existen muchos controladores MIDI genéricos disponibles, surge un problema cuando se intenta utilizar estos con sintetizadores de software. La GUI presentada por el sintetizador a menudo no tiene semejanza alguna con el diseño de rejilla típicamente empleado en estos controladores. Hay, por supuesto, excepciones a esta regla, como por ejemplo la colección de legados de Korg, con su controlador de hardware MS20 y las interfaces Masticine y

Kore de Native Instrument, al igual que una variedad de interfaces orientadas a DJs y Turntable. Sin embargo, la gran mayoría de los sintetizadores de software todavía necesitan ser controlados con un ratón o un conjunto genérico de controles. Este controlador busca resolver este problema presentando una intuitiva experiencia práctica al usar un sintetizador de software.

Figura 1. Sintetizador Digital de Audio

2.5. Raspberry PI.

Raspberry Pi [3] es un ordenador de placa única (es decir, que tiene todos los componentes integrados en una sola placa) y de bajo coste desarrollado en Reino Unido por la Fundación Raspberry Pi. Con el tiempo, la Raspberry Pi, junto con la placa microcontroladora Arduino, se han convertido en un referente en el mundo maker y en el desarrollo del internet de las cosas (IoT).

La Raspberry Pi destaca por su gran flexibilidad y su reducido precio (unos 35€), y nos permitirá desarrollar tanto proyectos de software, como, por ejemplo, un centro multimedia o un servidor web, como también proyectos hardware, como un sistema de videovigilacia o la construcción de una estación meteorológica.

2.6. Arduino.

El lenguaje del Arduino [4] está basado en el mítico lenguaje C. Si no has trabajado nunca con este lenguaje de programación, te basta con saber que C es el lenguaje en el que se ha desarrollado los sistemas operativos UNIX, Linux, y cientos de sistemas, programas y aplicaciones de ordenador. El lenguaje del Arduino es una versión reducida y mucho más sencilla de manejar que el lenguaje C. El objetivo de este lenguaje es que puedas programar de una manera intuitiva concentrándote en lo que quieres hacer más que en la manera de hacerlo.

Arduino fue desarrollado originalmente en el Interactive Design Institute en Ivrea (Italia) donde los estudiantes estaban familiarizados con un lenguaje llamado Processing. Este lenguaje estaba orientado a estudiantes de arte y diseño y utilizaba un entorno de desarrollo visual e intuitivo en el cual se basó el entorno de desarrollo del Arduino y su lenguaje de programación.

Trabajar con un Arduino consiste fundamentalmente en interactuar con los diferentes puertos de entrada y salida del Arduino. A fin de evitar al programador el engorro y la complejidad de programar estos puertos (ya sean analógicos, digitales o de cualquier otro tipo) el lenguaje de Arduino usa una serie de librerías (de las que no te tienes que preocupar ya que forman parte del lenguaje, ya las iremos viendo con detenimiento más adelante). Estas librerías te permiten programar los pins digitales como puertos de entrada o salida, leer entradas analógicas, controlar servos o encender y apagar motores de continua. La mayor parte de estas librerías de base ("core libraries") forman parte de una macro librería llamada Wiring desarrollada por Hernando Barragán.

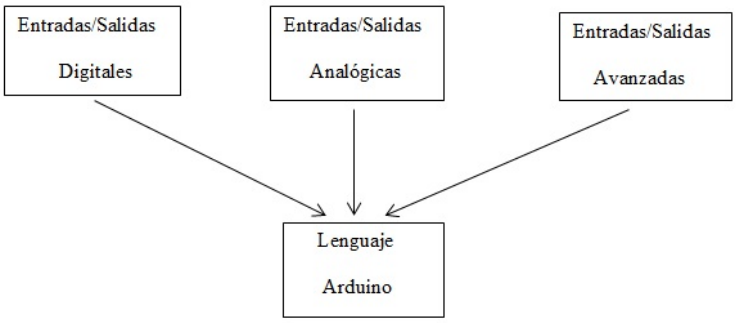


Figura 2. Diagrama de trabajo de Arduino.

Cada vez que un nuevo puerto de entrada o salida es añadido al Arduino, un nuevo conjunto de librerías especializadas en ese puerto es suministrada y mantenida por los desarrolladores del nuevo puerto.

Las placas se pueden ensamblar a mano o encargarlas pre ensambladas; el software se puede descargar gratuitamente. Los diseños de referencia hardware archivos CAD) están disponibles bajo licencia open-source, por lo que eres libre de adaptarla a tus necesidades.

2.7. MIDI.

MIDI (abreviatura de Musical Instrument Digital Interface) [5] es un estándar tecnológico que describe un protocolo, una interfaz digital y conectores que permiten que varios instrumentos musicales electrónicos, computadoras y otros dispositivos relacionados se conecten y comuniquen entre sí. Una simple conexión MIDI puede transmitir hasta dieciséis canales de información que pueden ser conectados a diferentes dispositivos cada uno.

El sistema MIDI lleva mensajes de eventos que especifican notación musical, tono y velocidad. Estos mensajes son enviados mediante un cable MIDI a otros dispositivos que controlan la generación de sonidos u otras características. Estos datos también pueden ser grabados en un hardware o software llamado secuenciador, el cual permite editar la información y reproducirla posteriormente

2.8. Altium Designer IDE.

Altium [10] desarrolla herramientas tanto a nivel hardware como software para el diseño electrónico. Su único entorno de diseño permite a los desarrolladores electrónicos conectar a la gente con los dispositivos.

El desarrollo de productos electrónicos implica la combinación de componentes disponibles para formar una plataforma hardware, y añadir la "inteligencia" a esta plataforma con el software de aplicación o mediante circuitos softwired en dispositivos programables como FPGAs.

Altium Designer proporciona una aplicación única y unificada que incorpora todas las tecnologías y capacidades necesarias para el desarrollo completo de productos electrónicos.

Altium Designer integra:

- Diseño de tarjetas y sistemas basados en FPGAs
- Sofware de desarrollo para procesadores basados en FPGA
- Diseño, edición y fabricacíon de PCB
- Dentro de un entorno de diseño único.

Altium Designer incluye la tecnología de productos anteriores como Protel y P-CAD. Es un entorno de desarrollo electrónico unificado que incorpora las herramientas necesarias para desarrollar los productos electrónicos inteligentes más avanzados. Basado en una aplicación simple suministra la tecnología de diseño más avanzada, la gestión de la integridad de los datos de diseño y un nuevo enfoque innovador para el diseño.

Figura 3. Logo Altium Designer.

2.9. Arduino IDE.

Arduino es una plataforma de prototipos electrónica de código abierto basada en hardware y software flexibles y fáciles de usar. Arduino puede "sentir" el entorno mediante la recepción de entradas desde una variedad de sensores que puede interactuar con su entorno mediante el control de luces, motores y otros artefactos.

El microcontrolador de la placa se programa usando el "Arduino Programming Language" (basado en wiring) y el "Arduino Development Environment" (basado en Processing). Los proyectos de Arduino pueden ser autónomos o se pueden comunicar con software en ejecución en un ordenador (por ejemplo, con Flash, Processing, MaxMSP, etc.).

Figura 4. Logo Arduino.

2.10. Pure Data.

Pure Data (o Pd) [9] es un entorno de programación gráfico en tiempo real y procesamiento gráfico. Pure Data es comúnmente usado para la ejecución de música en directo, efectos de sonido, composición, análisis de audio, uso como interfaz con sensores, uso de cámaras, controlar robots o incluso interacción con sitios web.

Debido a que todos estos diversos medios se manejan como datos digitales dentro del programa, existen muchas oportunidades fascinantes para la síntesis cruzada entre ellos. El sonido se puede utilizar para manipular vídeo, que podría ser transmitido a través de Internet a otro ordenador que podría analizar ese video y usarlo para controlar una instalación motorizada.

La programación con Pure Data es una interacción única que está mucho más cerca de la experiencia de manipular las cosas en el mundo físico. El programa en sí siempre está en ejecución, no hay separación entre escribir el programa y ejecutar el programa, y cada acción tiene efecto en el momento en que se completa.

Pure Data se ejecuta en GNU / Linux, Windows y Mac OS X, así como en plataformas móviles como Maemo, iPhoneOS y Android.

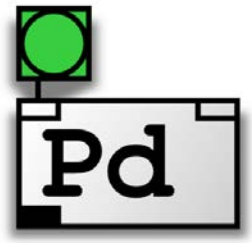


Figura 5. Logo Pure Data

3. Diseño del sistema

En este capítulo se aborda inicialmente el problema en cuestión, las necesidades que existen en el entorno de los sintetizadores de audio digitales, y como este producto sería capaz de satisfacer las carencias que se manifiestan hoy en día.

Posteriormente, se hablará de la arquitectura del sistema, explicando claramente qué función desempeña cada módulo.

3.1. Descripción del problema.

Como se ha comentado anteriormente, un sintetizador (de sonido) es un dispositivo que permite realizar un proceso de síntesis. Es decir, un sintetizador puro no modifica un sonido preexistente, sino que lo genera a partir de la combinación de elementos simples (normalmente señales periódicas y/o funciones matemáticas) que no existen fuera de los circuitos del dispositivo o del código del programa.

Si preguntamos a los músicos que trabajan con medios informáticos qué querrían poder hacer a nivel de generación de sonidos, muy posiblemente, casi todos estarían de acuerdo con las siguientes respuestas:

- Poder crear cualquier sonido imaginable
- Poder manipular cualquier sonido preexistente de cualquier manera que podamos concebir.

Estos objetivos son evidentemente utópicos, no solamente por las limitaciones tecnológicas sino también, y más importante, por los límites prácticos de nuestra imaginación. Las limitaciones tecnológicas nos obligan a valorar una serie de compromisos que hemos de tener en cuenta a la hora de diseñar o utilizar una determinada técnica de síntesis:

- Calidad del sonido.
- Flexibilidad.
- Tiempo de cálculo.
- Coste

Además de valorar estos compromisos a la hora de escoger un sistema, es necesario considerar que para que una técnica de síntesis sea útil para un músico, su control ha de ser intuitivo y por tanto ha de partir de una realidad sonora y musical existente.

El escenario planteado es el siguiente: Tenemos a nuestra disposición un módulo de control (gobernado por un Arduino) y una Raspberry PI 3. El módulo de control se va a encargar de dar una interfaz física al usuario, para ello serán necesarios una serie de actuadores que permitan al usuario operar cómodamente en su espacio de trabajo.

Adicionalmente se administrará de un sistema de radiofrecuencia para que realice la comunicación con la Raspberry PI de forma inalámbrica. Para este proyecto se ha elegido el chip de Nordic Semiconductor nRF24. El dispositivo NRF24L01, integra en un único chip, toda la electrónica y bloques funcionales precisos, para establecer comunicaciones RF (Radio Frecuencia) entre dos o más puntos a diferentes velocidades, (Hasta 2 Mb/seg) con corrección de errores y protocolo de reenvío cuando es necesario, sin intervención del control externo, lo que nos permite aislarnos de todo el trabajo sucio y complicado relacionado con la transmisión física.

En la Raspberry PI se aloja el programa que hará la síntesis de audio. Para desarrollar el programa hemos elegido el entorno de desarrollo Pure Data. Pure Data es un lenguaje de programación en sí mismo. Se usan unidades de código modulares y reusables, escritas nativamente en Pd, llamadas "parches" o "abstracciones", como programas independientes y son compartidas libremente entre la comunidad de usuarios de Pure Data.

Para el diseño de las placas donde debe ir la electrónica se ha utilizado Altium, Altium Designer proporciona una aplicación única y unificada que incorpora todas las tecnologías y capacidades necesarias para el desarrollo completo de productos electrónicos

3.2. Requisitos del sistema.

En este apartado de la memoria se van a enumerar los requisitos que se definieron para que el sistema funcione correctamente:

a. Requisitos funcionales.

- El sistema debe ser capaz de sintetizar un sonido digital a partir de un teclado MIDI modificado por un sistema de control.
- El sistema debe ser capaz de comunicarse inalámbricamente con el programa de síntesis de audio.
- La síntesis de audio se debe realizar mediante un programa diseñado en Pure Data.
- El sistema debe permitir la interacción en tiempo real con el sintetizador.
- El sistema debe mostrar en tiempo real los valores de todos los parámetros disponibles.

b. Requisitos no funcionales.

- El sistema debe ser multiplataforma.
- El sistema debe ser fácil de utilizar para cualquier tipo de usuario, sin necesidad de conocimientos previos o de algún tipo de entrenamiento.

3.3. Decisiones de diseño.

En este apartado se explica las decisiones de diseño que se han tomado a la hora de realizar el proyecto y las razones que nos han llevado a ello.

a. Arduino UNO vs Resto.

En este apartado se habla del “resto” tanto por el resto de modelos de Arduino disponibles en el mercado [6] (Mega, nano, mini, etc...), como por el uso como controlador de una Raspberry PI.

La primera decisión que se tomó fue en torno a la Raspberry Pi. Una de las dudas más frecuentes de las personas que se inician es entender las diferencias entre ambas placas, ya que en algunos casos pueden dar respuesta a necesidades similares, en realidad son dispositivos totalmente diferentes y muchas veces complementarios.

- La Raspberry Pi es un ordenador completo basado en un SoC (System on Chip) que contiene entre otras cosas, el procesador y la memoria RAM.
- Arduino es una placa de prototipado que contiene un microcontrolador. Esto significa que, a diferencia de la Raspberry PI, su funcionamiento está centrado en unas tareas muy específicas que básicamente serán leer y escribir datos utilizando sus pines de entrada y salida.
- De la misma manera podemos considerar que la Raspberry Pi es más flexible que un microcontrolador Arduino, las placas Arduino, al dedicarse a una tarea mucho más específica, pueden funcionar como un dispositivo en tiempo real, ser energéticamente más eficiente y menos propensas a fallos.

Por otro lado, había que seleccionar el tipo de Arduino a utilizar. Como la idea principal de este proyecto es que sea modular, se pensó que el chip idóneo para el proyecto era el ATMEGA328-P, junto a su placa de desarrollo Arduino UNO. Esto se debe a que es el único microprocesador de Arduino de formato convencional, lo que nos permite quitarlo y ponerlo cuantas veces queramos en nuestros módulos de control y volver a programarlo cuantas veces queramos sin tener que dañar la electrónica soldando y desoldando componentes.

La programación se ha realizado en el propio IDE de Arduino, ya que el software de Arduino consiste en un entorno de desarrollo (IDE) basado en el entorno de Processing y lenguaje de programación basado en Wiring, así como en el cargador de arranque (bootloader) que es ejecutado en la placa y que facilita su programación debido a la amplia comunidad que posee.

b. RF / Bluetooth / WIFI.

Usar conexiones inalámbricas nos ofrece unas posibilidades a las que resulta difícil renunciar cuando te acostumbras, prueba de ello es la cantidad de aplicaciones que se ejecutan bajo redes WIFI como Bluetooth.

Las WIFI nos permiten conectarnos a Internet con facilidad y podemos publicar páginas Web con valores de nuestros sensores y Arduinos o simplemente acceder a estos Duinos y pasarles parámetros u órdenes. EL BlueTooth es ideal si queremos controlar nuestros Arduinos sin más controlador que un móvil que ya llevamos habitualmente en el bolsillo.

Pero tanto el BT como el WIFI tienen unas limitaciones considerables en lo que se refiere a la distancia a la que podemos usarlo. No pasa más allá de 20 metros y eso en condiciones óptimas, pero no es raro que necesitemos más distancia, digamos 50 m, o 200 m, y por qué no, un kilómetro.

Como siempre, cuando hay una necesidad en el mercado, surge alguien dispuesto a ofrecer una solución en forma de electrónica, y en esta ocasión vamos a presentar una solución magnífica para el tipo de problema que planteamos, el módulo nRF24L01 [7]. Un medio de intercambiar órdenes entre el Arduino y la Raspberry PI, a una distancia que puede variar entre 50 metros y hasta un kilómetro. Son muy fáciles de usar y configurar y además son los enlaces más baratos que podemos encontrar para vincular nuestros Arduinos con la Raspberry Pi.

Aparte del alcance de estos módulos, la gran ventaja que tienen es que son completamente autónomos e independientes, ellos mismos crean su propia red, no como la red WIFI y

además se pueden configurar números de serie para las conexiones, por lo que solucionamos el problema de la congestión de Bluetooth.

c. Windows / Linux / macOS.

Esta ha sido la decisión más simple, ya que se ha utilizado un programa multiplataforma como es Pure Data para la síntesis de audio, por lo que podremos utilizar nuestro código en cualquiera de las plataformas existentes.

De la misma manera, el software de desarrollo Arduino también es multiplataforma, por lo que esto no ha sido un problema.

3.4. Arquitectura del sistema.

a. Diagrama.

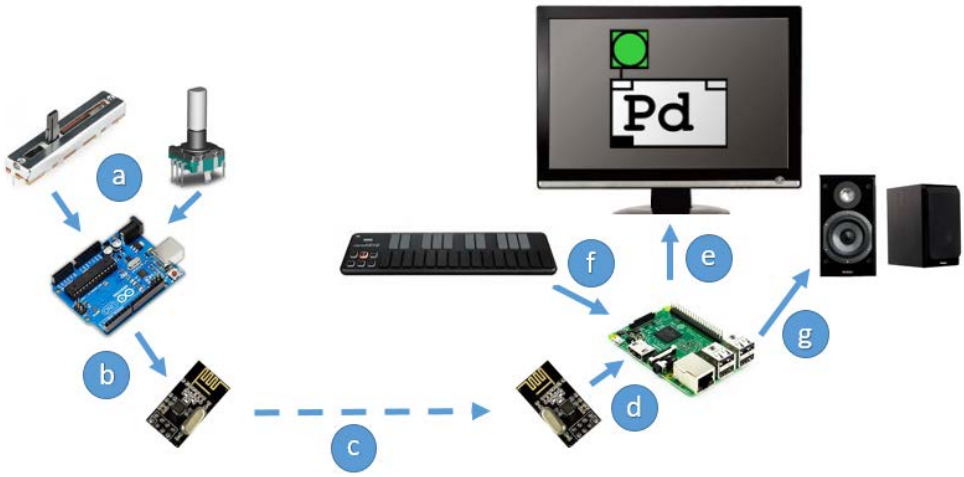


Figura 6. Estructura del sistema completo.

En la *Figura 6* se puede apreciar los diferentes módulos que componen el proyecto completo. En el apartado *3.5 Componentes del sistema* se realizará una explicación exhaustiva de todos los componentes que se utilizan en cada uno de los módulos del proyecto.

3.5. Componentes del sistema.

a. Interfaz física de usuario.

Lo primero que se va a comentar es la interfaz física que se ha dispuesto para el usuario. En ella, se disponen de una serie de actuadores que pasaremos a comentar posteriormente y unos leds identificativos.

Hay tres tipos de actuadores, con diferentes funciones. El primero de ellos, es un potenciómetro deslizante. La idea de incluir este dispositivo es poder regular la velocidad de movimiento de nuestros faders digitales. En segundo lugar, se tienen dos switches rotativos, cuya función es elegir el modo del programa en el que nos encontramos. Se ha hecho una división del programa por módulos y faders, que se explicará con detalle en el apartado *3.5.e. Programación en Pure Data*, que es controlada por estos switches de forma independiente. Por último, se ha incluido un encoder rotativo para poder controlar

independiente las variaciones de cada uno de los faders sin tener que tener almacenado el valor en cada caso junto a un sistema de posicionamiento del fader que coloque en todo momento posición del fader según esté en el programa.

Cada uno de los elementos de control de los que disponemos en el sintetizador se denominarán de aquí en adelante modos. Para la identificación por parte del usuario del modo en el que se encuentra, se ha dispuesto un sistema de leds, por medio del cual y ayudándose de una leyenda, el usuario es capaz de conocer en todo momento el modo que tiene seleccionado. El uso de la leyenda se debe al excesivo tamaño del que tendría que haber sido la placa de cobre que sustentara toda la electrónica, y así facilitar la lectura de la misma y la futura utilización de la base de madera si se reprogramara el sistema.

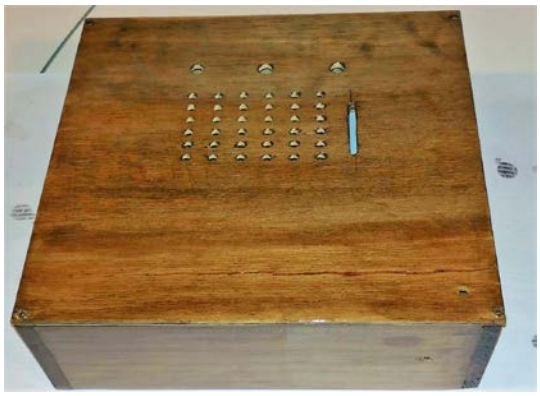


Figura 7. Caja de madera utilizada de soporte.

b. Módulo Arduino.

En el módulo de Arduino se va a englobar la electrónica y la programación tanto de los actuadores como del módulo de radio frecuencia.

El diseño de la electrónica se ha pensado para que pueda ser lo más modular posible. Se ha diseñado la placa en dos niveles, una primera capa en la que se encuentran los actuadores y los leds, y una segunda en donde se encuentran el microprocesador, el módulo de comunicación, la fuente de alimentación y la alimentación de los leds.

La programación del Arduino es la que realiza el control de los actuadores para posteriormente enviar los datos vía radio-frecuencia a la Raspberry PI. Esto facilita enormemente las cosas, ya que con una placa Arduino UNO puedes programar cuantos microprocesadores como quieras. El programa principal lee periódicamente los valores de cada uno de los actuadores, enviando el valor deseado a través del módulo de RF solamente cuando se detecta un cambio en el encoder, que es el que nos marca una modificación del valor de los faders.

c. Comunicación radio-frecuencia.

La comunicación entre la Raspberry PI y el Arduino se ha realizado a través de unos módulos de radio frecuencia, los nRF24L01 [8]. Las razones por las que se ha decidido usar estos módulos se van a enumerar a continuación:

- Muy baratos, unos pocos dólares.
- Operan en la banda de 2.4Ghz, que es de libre uso a nivel mundial.
- Velocidad configurable de 250kb, 1 Mb o 2Mb por segundo.
- Muy bajo consumo en Stand By.

El alcance depende de si hay visión directa entre los nodos, o por el contrario hay obstáculos, pero nos ofrece un mínimo de unos 20 m hasta un máximo e 80m en óptimas circunstancias, en el modelo básico con la antena integrada, que es el que se ha empleado en el proyecto.

En *Figura 8* se muestra los modos de operación y el funcionamiento de los módulos nRF24L01.

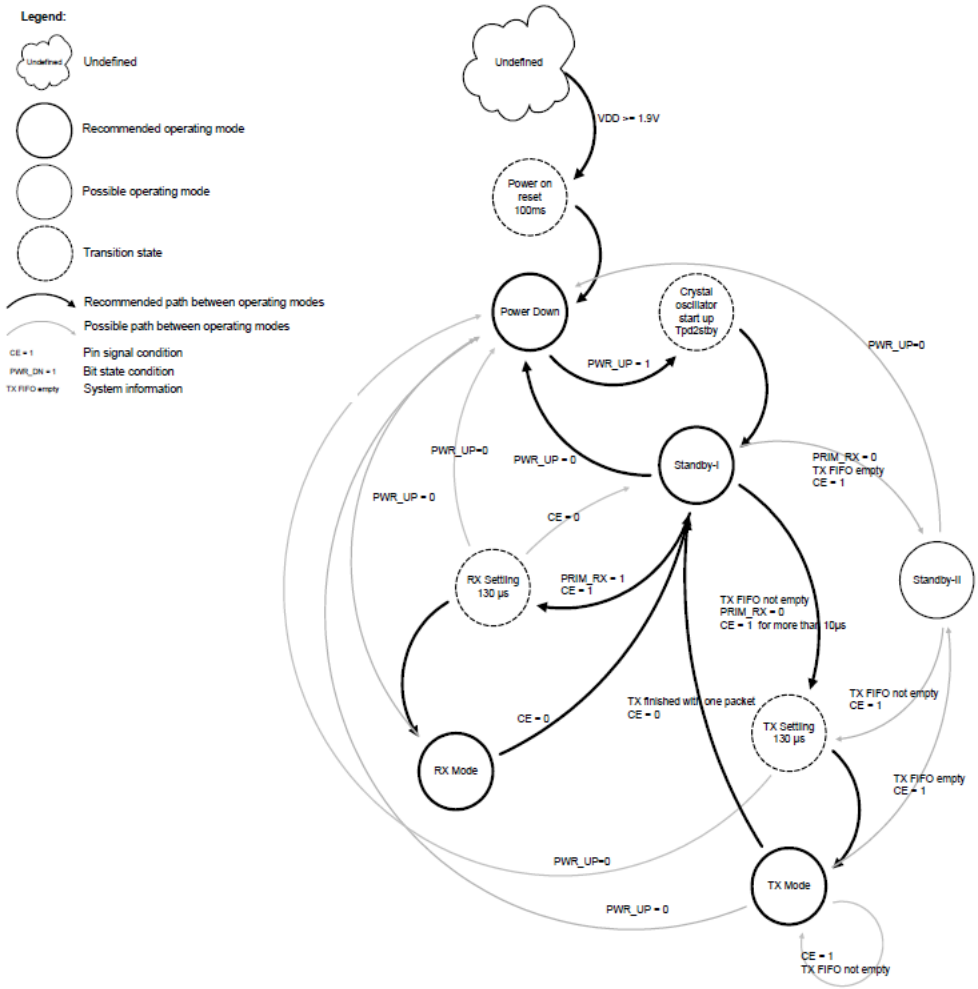


Figura 8. Diagrama de estados del control de la radio del módulo nRF 24L01 [8]

Otra de las funcionalidades que tienen estos módulos, es la posibilidad de elegir la frecuencia central del canal de radio-frecuencia sobre el que trabajará. El canal ocupa un ancho de banda de menos de 1MHz para una transmisión de 250kbps hasta 1Mbps y de menos de 2MHz para una transmisión a 2Mbps. El rango de frecuencias en el que puede operar varía de 2.400GHz hasta 2.525GHz, todo configurable por el usuario.

También tienen la gran ventaja de tener un protocolo de comunicación integrado en el chip, de manera que, con la correcta configuración de los mismos, es posible disponer de una comunicación segura.

Enhanced ShockBurst es una capa de enlace de datos basada en paquetes que incluye ensamblaje automático de paquetes, reconocimiento automático y retransmisiones de paquetes. Enhanced ShockBurst permite la implementación de comunicaciones de muy bajo consumo y alto rendimiento con microcontroladores de bajo coste.

Las características de Enhanced ShockBurst permiten mejoras significativas en la eficiencia energética para sistemas bidireccionales y unidireccionales, sin añadir complejidad en el lado del controlador host.

El formato de paquetes de Enhanced ShockBurstse detalla en la *figura 9*. Éste contiene un preámbulo, una dirección, un campo de control, el mensaje y el CRC.

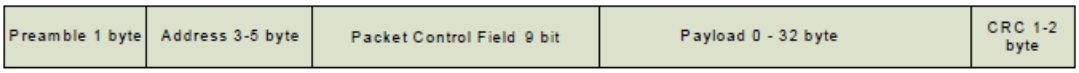


Figura 9. Datagrama del protocolo Enhanced ShockBurstse [8]

d. Módulo Raspberry.

En el módulo de Raspberry [9] es donde realmente se hace la síntesis de audio. En este apartado se va a hablar del modo de trabajo de la Raspberry PI más que de los periféricos que la acompañan, ya que merecen un apartado especial cada uno.

Lo primero que hay que tener en cuenta cuando se trabaja con Raspberry Pi es que es un entorno Linux y, por lo tanto, si no se está acostumbrado, tiene un proceso de adaptación. A cambio te da una gran flexibilidad para trabajar con los pines incorporados, pudiendo acoplar infinidad de accesorios a la misma.

Gracias a la flexibilidad que nos proporciona Raspberry, en un mismo dispositivo se ha integrado el módulo de recepción de datos (nRF24L01), la entrada del teclado MIDI a través de un puerto USB, la salida de audio a través de un códec de audio a través de USB y la salida a un monitor para tener una visión del programa que se está ejecutando.

La programación del módulo de RF en este caso se ha hecho en Python 3.4. Python es un lenguaje de script o interpretado (a diferencia de lenguajes como C, donde el código se compila) y permite que el código generado se pueda ejecutar sin cambios en prácticamente cualquier plataforma que disponga de un intérprete de Python (Windows, MacOSX, Linux, ...). Otra de las grandes ventajas de Python es la de disponer de una sintaxis sencilla y clara, muy cercana al lenguaje natural, que facilita el proceso de aprendizaje y, a la vez, es muy potente para el programador experto.

Figura 10. Raspberry Pi 3 model B.

e. Programación en Pure Data.

Para la síntesis de audio se ha elegido Pure Data. Esto es debido a que su entorno de programación gráfico se asemeja mucho a la idea de un sintetizador, permitiendo realizar bloques de código simulando cada bloque del sintetizador.

Las unidades donde se programa el código se llaman "patch" y son utilizadas como programas independientes. Los patchs constan de diferentes objetos interconectados entre ellos. En su parte superior encontraremos las entradas, donde se les enviaran valores numéricos u otros tipos de datos, y en la inferior la salida de estos.

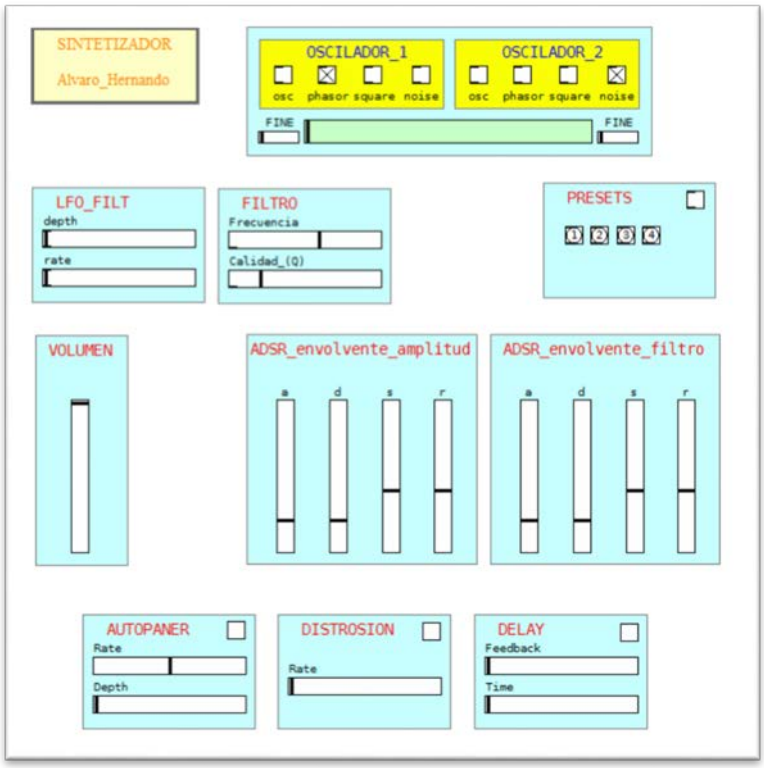


Figura 11. Pantalla principal del programa en Pure Data

También existe la posibilidad de crear patchs secundarios conocidos como subpatchs, que se encuentran dentro del patch principal. Se crean escribiendo en un objeto las letras "pd" seguidas de un espacio y el nombre que se le quiera dar a ese subpatch. Clicando encima se nos abre la ventana donde encontramos el código de nuestro subpatch.

El programa tiene dos estados en los que se puede encontrar el usuario. En modo de edición o en modo de ejecución. Cuando estamos en el modo edición, podemos modificar el contenido de las cajas, o la conexión entre ellas. En el modo de ejecución tenemos la posibilidad de poner en marcha todo el patch, e ir modificando valores durante su reproducción o cuando este, esté parado. Podemos enviar bangs, modificar el valor de variables dentro de los objetos "números", o activar y desactivar sectores del código con el objeto "toggle", activado cuando tiene una cruz y desactivado cuando no.

Esto último resulta muy interesante, ya que en tiempo de ejecución te permite tanto modificar nuestra síntesis tanto con el controlador que se ha creado como mediante teclado y ratón.

f. Entrada teclado MIDI.

Para tener la generación de las notas musicales se ha utilizado un teclado MIDI, concretamente el nanoKEY2 de KORG. La serie Nano de KORG, reconocida marca de sintetizadores y teclados, es la serie más básica de todas las disponibles en la compañía. Esta serie está pensada o dirigida únicamente a músicos aficionados o personas que desean hacer uso del teclado MIDI para usos de edición en tiempo real de vídeo.

Este teclado cuenta con tres diseños diferentes, estando cada uno de ellos dirigido o pensado para un uso completamente diferente. Disponemos de un teclado normal, un teclado de grandes botones y sin escala (pensado para ser usado con un sintetizador) y el teclado de niveles para ajustes.

El que hemos podido usar nosotros es el teclado tradicional, con una distribución normal de teclas que recuerda a un piano, aunque con un diseño más moderno y con teclas sin contrapeso. No se han olvidado de añadir un sensor de presión, así que, aunque el teclado no cuenta con teclas balanceadas, sí que es capaz de detectar la presión ejercida sobre las mismas. Un aporte adicional de este teclado es que cuenta con unos botones extra para poder cambiar de octava de forma cómoda.

De esta manera extraeremos tanto la frecuencia de la nota que estamos tocando como la intensidad y momento de la pulsación, datos que utilizaremos en nuestro programa para la síntesis del sonido.

g. Salida de audio.

Paspberry Pi dispone de una salida de audio donde podemos conectar unos altavoces auto amplificados o unos auriculares, pero si buscamos conseguir mejor calidad de sonido o disponer de una entrada de audio para grabar, tendremos que tirar de una tarjeta de sonido externa. Disponemos de dos opciones igual de válidas:

- Una solución es comprar una tarjeta Wolfson Audio que va conectada a los pines GPIO de nuestra Raspberry y dispone de varias entradas y salidas de audio. Esta placa cuesta unos 25-30€.
- Otra solución es usar una tarjeta de sonido USB, las funciones de estas tarjetas USB están mucho más limitadas ya que suelen tener tan sólo una entrada y una salida de audio, pero lo bueno que tienen es su precio, aproximadamente 2 euros, lo que supone un ahorro de más de 20€ respecto a las tarjetas Wolfson.

Para nuestro proyecto se ha decido comprar una tarjeta de sonido USB genérica, ya que con eso es suficiente para los requerimientos de nuestro proyecto.

4. Implementación del sistema

En este capítulo se presentan los detalles del desarrollo del controlador y de la síntesis de audio desarrollados en este proyecto, así como las herramientas que se han utilizado para su implementación.

En la sección 4.1, lo primero que se va a detallar es la funcionalidad y distribución de las PCBs que se han creado, mostrando una vista en planta de las mismas para poder explicar a qué se dedica cada una.

Posteriormente se explicará brevemente el código Arduino necesario para hacer la lectura y envío de los datos, haciendo especial énfasis en las funciones de lectura del encoder y de transmisión de datos, ya que el resto son muy comunes en el entorno.

Por último, se hablará del programa desarrollado en Pure Data, explicando la generación de la síntesis de audio. De igual manera que en el apartado anterior, solo se expondrán detalladamente los bloques más importantes de la síntesis.

4.1. Desarrollo del sistema.

A continuación, se explicará brevemente la implementación de los diferentes componentes de nuestro sistema.

a. PCB.

Se han realizado 3 PCBs para este proyecto. Dos de ellas se utilizan en la interfaz física de usuario, estas a su vez, se van a incluir en el interior de una caja de madera, dándole así un aspecto clásico al mismo. La tercera se va a colocar como si fuera un shield en los pines GPIO de la Raspberry PI.

Para el desarrollo de las placas de la interfaz, necesitábamos conocer el diseño final de nuestro sintetizador, ya que se sabía de antemano que iba a condicionar enormemente el desarrollo del proyecto. Esto se debe a que como es una interfaz física, debe ser cómoda de manejar, pero a su vez, se debía intentar que el tamaño de la PCB fuera el mínimo posible, por lo que hubo un compromiso entre tamaño y coste de la PCB.

Figura 12. Vista en alzado de la electrónica del sistema.

Finalmente, se ha decidido hacer una estructura de dos capas, esto es debido a que se quería dejar los actuadores a una determinada altura, *Figura 12*. De esta manera, se ha podido dejar en la parte superior los switches rotatorios, el encoder, el potenciómetro deslizante y todos los leds. A su vez, con esto conseguimos aislar una parte del circuito, de manera que si queremos modificar alguna parte de la estructura solo necesitaremos rehacer una de las dos placas.

El diseño de la misma se muestra en las *ilustraciones 13 y 14*, denominándola de aquí en adelante PCB de control.

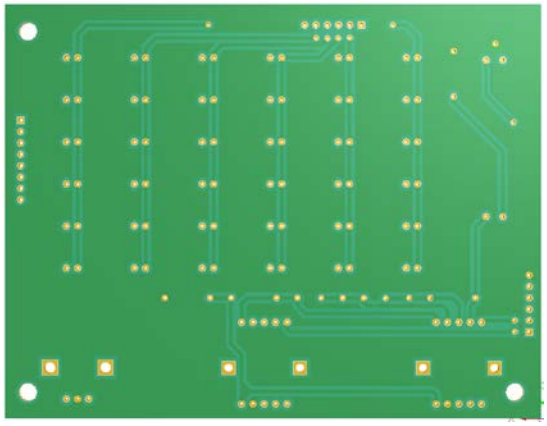


Figura 13. PCB control - Bottom layer

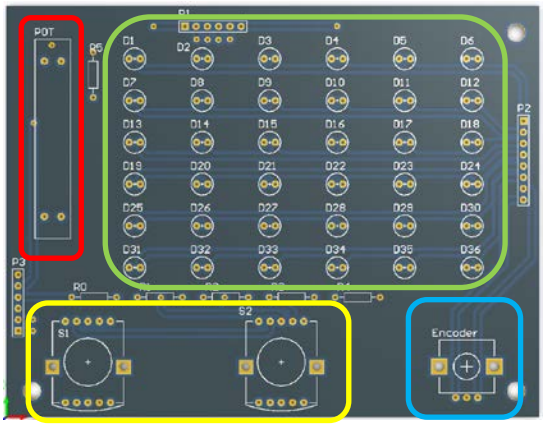


Figura 14. PCB control - Top layer

En la *Figura 13* se muestra la cara bottom de esta PCB, mientras que en la *Figura 14* se muestra la cara top de la misma. En esta cara Top se pueden apreciar la posición de los leds (marcado en verde), el potenciómetro deslizante (marcado en rojo), los switches rotatorios (marcados en amarillo) y el encoder (marcado en azul).

Esta PCB controla las funciones que van a estar activas en el programa de Pure Data, así como el valor que le vamos a mandar al mismo. Las funciones de cada uno se explican a continuación:

- Leds. Los leds se han utilizado para señalizar el estado en el que nos encontramos en el programa. Se dispondrá de una breve leyenda en la parte superior de los mismos que indiquen el propósito de cada uno. Esto se ha decidido hacer así para poder poner diferentes plantillas de texto sobre los leds y de esa manera poder ampliar la utilidad del módulo tanto como se quiera.
- Potenciómetro deslizante. El potenciómetro se va a usar en este módulo como control general de volumen, pero podría tener otros usos dada su similitud con un fader, elemento muy utilizado en la industria audiovisual.
- Switches rotatorios. Estos switches están conectados a una serie de resistencias de manera que su función es la de seleccionar un valor de tensión y pasárselo al micro para que haga una lectura analógica. De esta manera, somos capaces de saber la posición de cada switch. Esto se utiliza para que el micro seleccione el modo de operación en el que nos encontramos y lo muestre en los leds mediante la matriz fila – columna.
- Encoder. El encoder es el encargado de las variaciones de nuestros parámetros en Pure Data. Es lo que determinará que valores toma cada módulo del sintetizador. Se ha elegido este componente para este propósito ya que no necesita conocer la posición anterior del mismo, indicando únicamente el incremento o decremento de una variable, haciendo que la comunicación se simplifique enormemente.

La segunda PCB se encarga del procesado y envío de los datos recogidos de la PCB anterior.
Como se ha hecho anteriormente, se va a mostrar una imagen de la cara Top (*Figura 15*) y otra de la cara Bottom (*Figura 16*) marcando los bloques esenciales que la integran.

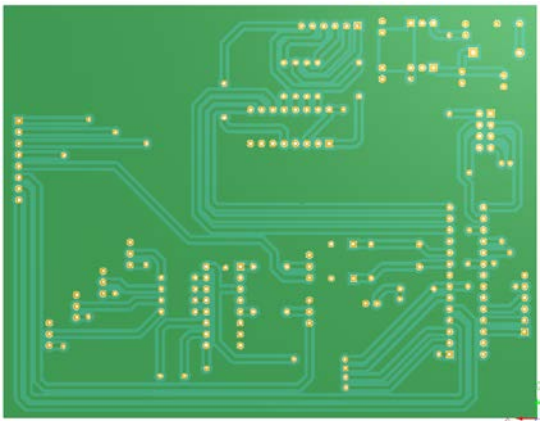


Figura 15. PCB Arduino - Bottom layer.

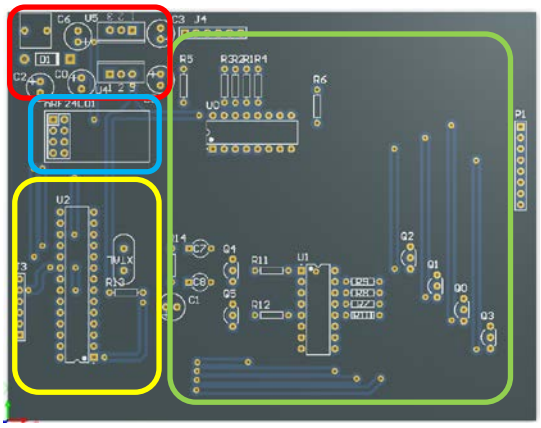


Figura 16. PCB Arduino - Top layer

En la cara Top vemos cuatro bloques principales en los que se divide la PCB.

- La fuente de alimentación (marcada en rojo). Este bloque se encarga de regular y proteger la alimentación del circuito. Está formada por un diodo protector y dos reguladores de tensión, uno a 5V que alimenta los integrados y leds, y uno de 3'3V que se encarga de alimentar al módulo de RF.
- El módulo de RF, nRF20L01 (marcado en azul). Es el encargado de realizar la transmisión de datos a la Raspberry PI. Está conectado al chip Arduino a través de los puertos SPI.
- El micro Arduino (marcado en amarillo). Es el cerebro de ambas PCBs, se encarga de analizar los datos recogidos en la PCB de control, analizarlos y tanto configurar correctamente los multiplexores en cada caso como de mandar el comando de envío al módulo de RF.
- Bloque selector de LEDs (marcado en verde). Este bloque se compone de dos multiplexores, una serie de transistores NPN y las resistencias asociadas. Uno de los multiplexores es el encargado de poner 5V en el ánodo de los leds con una resistencia de limitación de corriente, el otro es el encargado de seleccionar uno de los transistores para ponerlo en modo corte y el resto en abierto para que el led correspondiente luzca.

En la tercera PCB, solo tendremos un adaptador para colocar en la Raspberry PI el módulo de RF. Se puede ver su diseño en las *ilustraciones 17 y 18*.

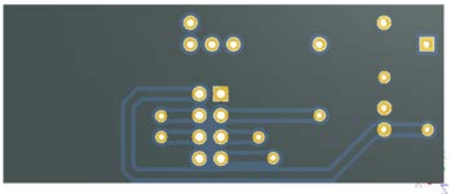


Figura 17. PCB Raspberry Pi - Bottom layer.

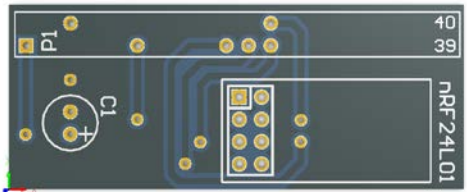


Figura 18. PCB Raspberry Pi - Top layer.

b. Software.

El programa de Arduino se ha simplificado al máximo posible para evitar el envío constante de datos que pudiera dar lugar a una congestión en las comunicaciones. Para ello se ha creado una estructura en la que solamente se van a enviar datos cada vez que el sistema detecte un cambio o bien del encoder o del potenciómetro deslizante.

En nuestro sistema se ha aprovechado la librería Timer para poder leer periódicamente los valores de los actuadores y así que el envío no se demore lo más mínimo. Esto se debe a que en un futuro se ha pensado en incluir varios módulos, y en el caso de tener un entorno muy ruidoso el módulo de RF tiene activada la opción de reenvío.

Otra de las cosas que se ha empleado es la interrupción integrada en los pines digitales. El micro Arduino UNO tiene integrado en los pines 2 y 3 la posibilidad de usar dichos pines como interrupción del sistema, de esta forma, se puede leer el encoder en cualquier momento.

El encoder de rotación es un potenciómetro electromecánico que convierte la posición angular del potenciómetro a un valor digital por pulsos como se muestra en la *Figura 19*.

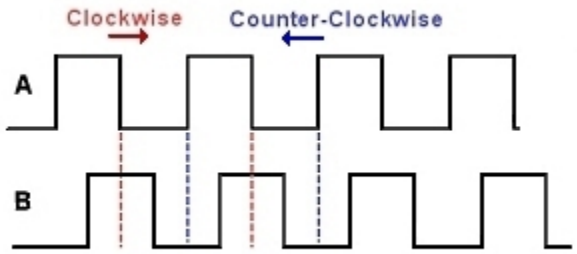


Figura 19. Funcionamiento de un encoder rotativo.

Digamos que es un potenciómetro sin fin con el cual se puede obtener una precisión muy alta. Para ello, se ha creado una función llamada doEncoder que lee las variaciones de los pulsos vistos en la *figura 19*. En ella, se ve en la línea marcada como A la salida de una de las patas del encoder, y en la línea B la de la otra. Lo que se hace es leer en el momento de cambio de uno de los pines y ver si es a favor o contra el sentido de las agujas del reloj.

```
void doEncoder() {
    int MSB = digitalRead(encoderPinA);
    int LSB = digitalRead(encoderPinB);

    int encoded = (MSB << 1) | LSB;
    int sum = (lastEncoded << 2) | encoded;

    if(sum == 0b1101 || sum == 0b0100 || sum == 0b0010 || sum == 0b1011)
       {varEncoder--;cambioDatos = true;}
    if(sum == 0b1110 || sum == 0b0111 || sum == 0b0001 || sum == 0b1000)
       {varEncoder++;cambioDatos = true;}

    lastEncoded = encoded;
}
```

Respecto al software incluido en la Raspberry PI, solamente se utiliza para hacer la recepción de los datos mandados desde Arduino y transmitírselos al programa de Pure Data que comentaremos en la siguiente sección.

c. Pure Data.

En la *Figura 20* se muestra la interfaz gráfica [11] de usuario. En ella, se ven los bloques principales que componen el sintetizador que se ha diseñado (Osciladores, filtros de frecuencia, presets, control de volumen, ADSRs y efectos) y que se explicarán al final de la sección.

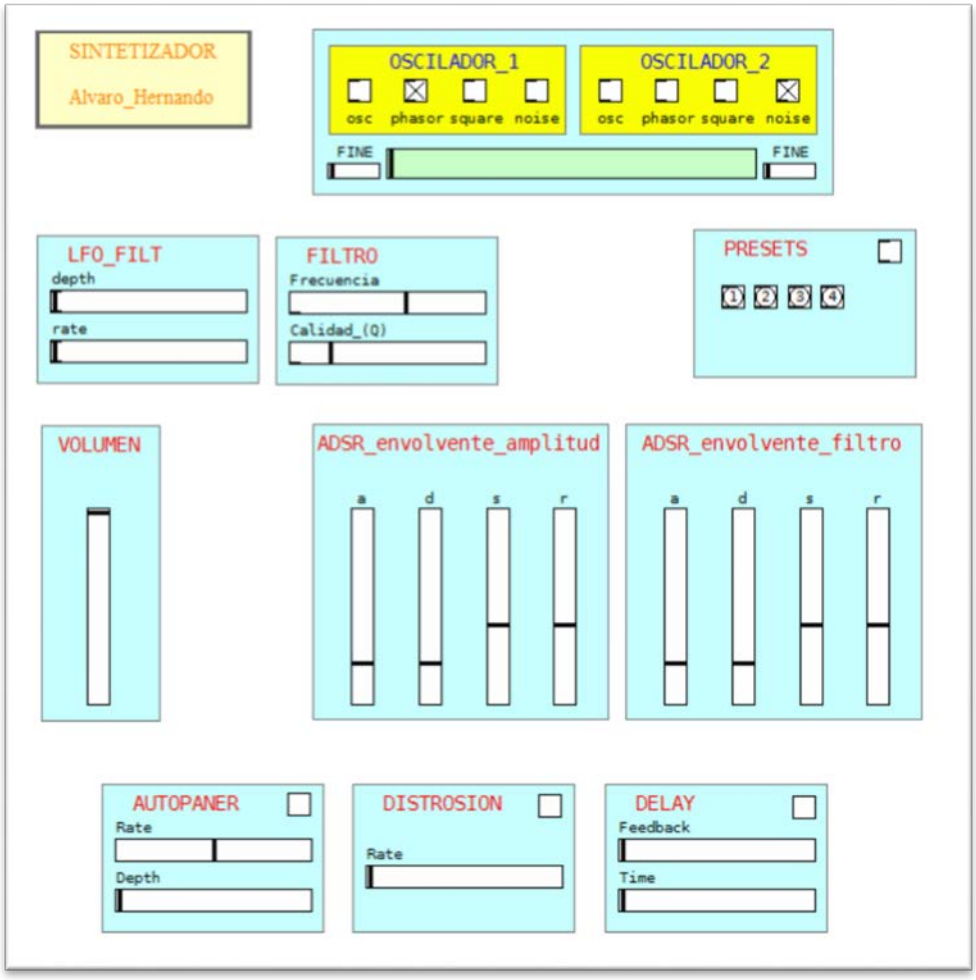


Figura 20. Interfaz gráfica de usuario diseñada en Pure Data.

Antes de entrar a analizar la estructura creada para la síntesis se va a exponer un diagrama del mismo en la *Figura 21*.

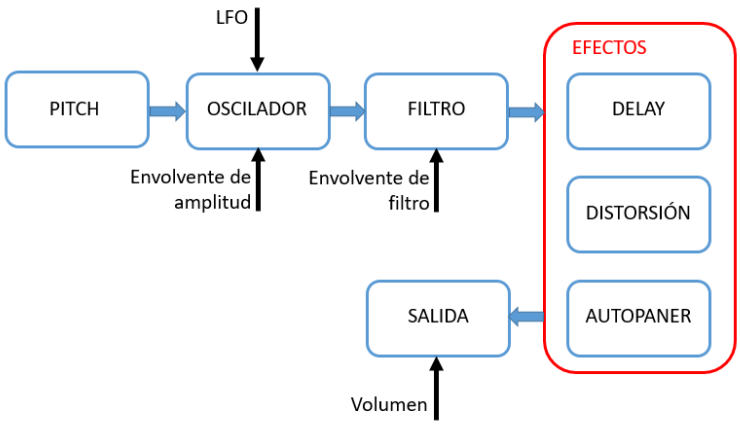


Figura 21. Esquema del sistema de síntesis.

A continuación, se va a mostrar la estructura interna de nuestro sintetizador en *la Figura 21*. En ella se muestra la secuencia que va a seguir el valor MIDI leído mediante el objeto *notein (marcado en rojo)*. Este objeto de Pure Data tiene tres salidas, la frecuencia de la nota, su velocity y el canal utilizado. La frecuencia la utilizaremos para saber la nota que estamos tocando en el teclado MIDI, mientras que el velocity para saber cuándo se ha dejado de pulsar.
Lo primero que hacemos es utilizar el objeto *poly (marcado en azul)*. *Poly* permite una polifonía manteniendo la información de las notas que han llegado, de manera que se pueden tocar hasta 'x' teclas simultáneamente, en nuestro caso hasta 4 notas. *Poly* a su vez nos indica con un número indicador la nota que estamos tocando. Esto nos servirá poder luego mediante el objeto *route* separar dichas notas y así poder hacer la síntesis de todas ellas a la vez.
Posteriormente pasamos al subpatch *voz* cada una de estas notas identificadas del 1 al 4, de manera que el sintetizador las pueda tratar en paralelo. La estructura del subpatch *voz* (marcado en verde) integra tanto la etapa de osciladores como el bloque del oscilador de baja frecuencia (LFO).
En este proyecto se han implementado 4 tipos de osciladores, uno sinusoidal, uno de onda cuadrada, uno de onda triangular y un generador de ruido blanco.
Posteriormente, hacemos pasar nuestra señal por el LFO. Un LFO es un oscilador secundario que opera a una frecuencia significativamente inferior (de ahí su nombre), típicamente alrededor o por debajo del límite del oído humano (que está aproximadamente en unos 20 Hz). Esta señal de baja frecuencia o de control es utilizada para modular la señal de sonido, cambiándola sin introducir ninguna otra fuente de sonido.
La utilización de un oscilador de baja frecuencia como medio para modular otra señal introduce una serie de complejidades en el sonido resultante que hacen que se pueda lograr multitud de efectos. El resultado específico varía tremendamente dependiendo del tipo de modulación, la frecuencia relativa de la señal LFO (incluido en el subpatch voz) y la señal que sea modulada.

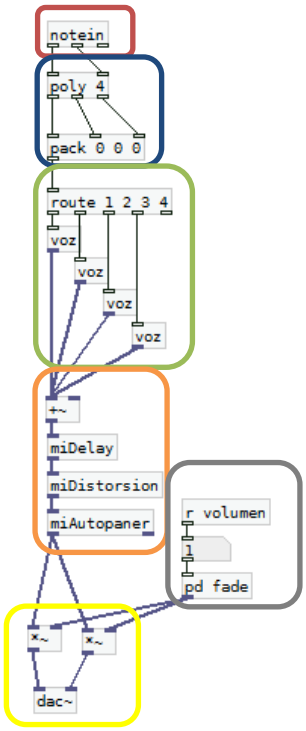


Figura 22. Síntesis realizada en Pure Data.

Posteriormente (marcado en naranja), combinamos todas las señales resultantes del subpatch voz y les aplicamos los efectos de *delay*, *distorsión* y *autopaner*. En el efecto de *delay* graba un intervalo de audio y lo reproduce periódicamente creando un efecto eco. Con el efecto *distorsión*, se ha aplicado la función tangente hiperbólica de manera que se crean armónicos de la señal original.
La última etapa del bloque de efectos es el autopan. Panning es la distribución de una señal de sonido en un nuevo campo acústico estéreo o multicanal determinado por un ajuste de control panorámico. El efecto de *autopaner* hace un barrido entre las salidas en un altavoz estéreo, de manera que variando los ajustes de velocidad y profundidad somos capaces de ajustar dicho efecto.
Por último, tenemos el control de volumen (marcado en gris) y la salida de audio (marcado en amarillo).

A continuación, una vez conocidos los bloques de los que se compone el sintetizador, vamos a analizar la interfaz gráfica de usuario:

– Osciladores (*Figura 23*). En este bloque se controla el tipo de oscilador que queremos usar, así como la mezcla que realizamos de cada uno de los dos disponibles. Adicionalmente se ha añadido un ajuste fino para poder hacer un cambio de octava desde el sintetizador.

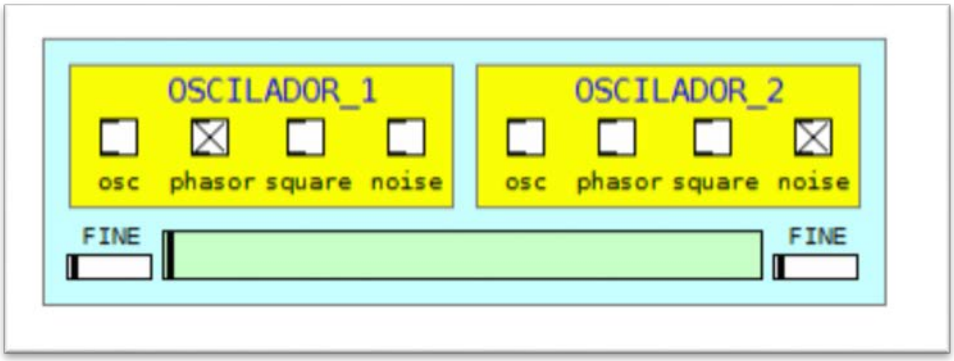


Figura 23. Control Osciladores Pure Data

– Control de frecuencia (*Figura 24*). En estos bloques se controlan los valores de variación de frecuencia. Estos se dividen en un filtro de baja frecuencia y un filtro de frecuencia. En el filtro de baja frecuencia se va a controlar la oscilación que va a promover el oscilador de baja frecuencia en la señal original. El filtro de frecuencia se comporta como un oscilador controlado por voltaje, pudiendo modificar tanto la frecuencia como el factor de calidad del mismo de manera que cambia la resonancia del sistema.

Figura 24. Controles Frecuencia Pure Data

– Envolvente de amplitud y de frecuencia (*Figura 25*). En estos bloques se controla el tipo de envolvente. La 'envolvente acústica' permite definir, en términos de cuatro parámetros globales, la evolución temporal en amplitud de cualquier sonido.

  Está determinado por cuatro principales parámetros:
  - Ataque: Es el tiempo de entrada. Lo que tarda en escucharse el sonido después de haber sido ejecutado el instrumento.
  - Decaimiento: Es el tiempo que tarda la amplitud en reducirse a la de sostenimiento, después de haber alcanzado la amplitud máxima, sin despegar la tecla o punto de inducción vibratoria.
  - Sostenimiento: Después del decaimiento, es la amplitud que se mantiene constante hasta que se deja de inducir vibración (en el caso de los sintetizadores, hasta cuando se suelta una tecla o cable que controla este fin, por ejemplo).
  - Relajación: El tiempo que tarda el sonido en perder toda su amplitud después de despegar la tecla o punto de inducción vibratoria.

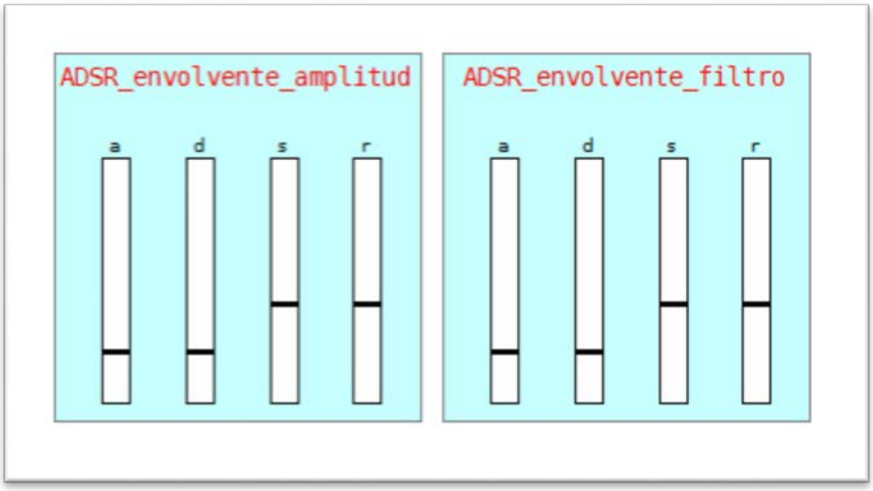


Figura 25. Controles ADSR Pure Data.

– Efectos (*Figura 26*). En estos bloques se controlan los parámetros que se van a usar en los efectos que introduzcamos en nuestro sintetizador.

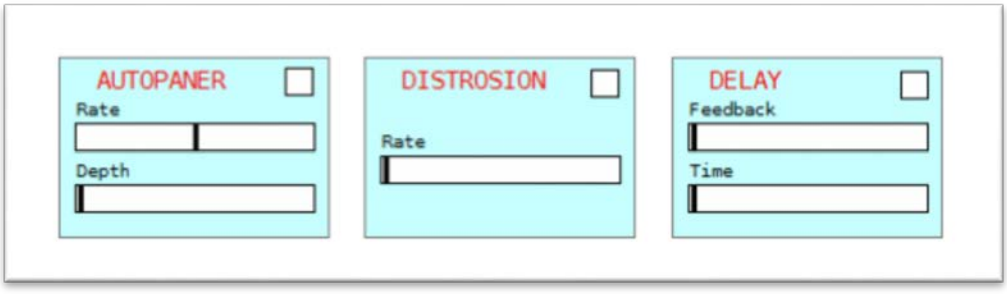


Figura 26. Control efectos Pure Data.

– Volumen (*Figura 27*). En este bloque se controla el volumen general del sintetizador.

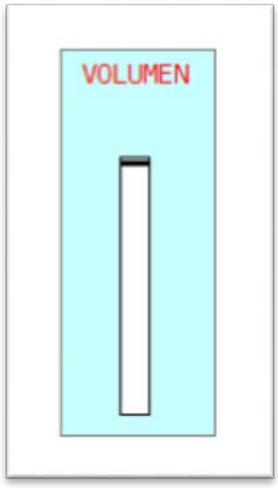


Figura 27. Control Volumen Pure Data.

– Presets (*Figura 28*). En este bloque se tiene la opción de cargar o grabar una serie de presets que están incluidos en el sintetizador, y que se pueden modificar en cualquier momento.

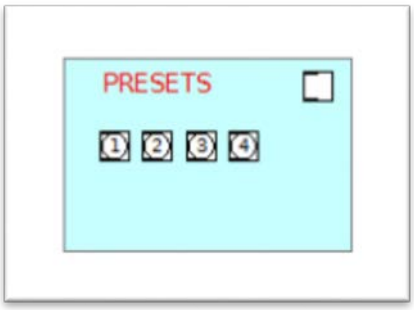


Figura 28. Presets Pure Data.

5. Planificación y presupuesto

En este apartado de la memoria se presenta la planificación y el presupuesto que se ha estimado para conseguir los objetivos que plantea este proyecto.

En primer lugar, se expone la planificación del proyecto, identificando todas las fases por las que ha pasado y representando su evolución mediante un diagrama de Gantt.

A continuación, se detalla el material que se ha necesitado para llevar a cabo el proyecto, así como los costes personales del mismo. Por último, se expondrá el cómputo del coste total del proyecto.

5.1. Fases del proyecto.

Fase 1: Estudio del Estado del Arte.

Duración: 15 días.

- Lectura de estudios previos.
- Estudio de las principales características de los protocolos de comunicaciones inalámbricos.
- Estudio de las principales diferencias entre las diferentes plataformas de desarrollo tanto hardware como software.

Fase 2: Estudio del protocolo Enhance ShockBurst.

Duración: 7 días.

- Comprensión del funcionamiento del protocolo Enhance ShockBurst.
- Realización de pruebas en un entorno controlado con dos dispositivos conectados a través de los módulos nRF24L01, utilizando C como lenguaje de programación.

Fase 3: Desarrollo del prototipo electrónico.

Duración: 20 días.

- Diseño de un primer prototipo en placa blanca para probar diferentes componentes y cual se adapta mejor a las necesidades del proyecto.
- Decisión de cambiar los componentes pensados inicialmente, como un display conectado al controlador, para obtener una experiencia más sencilla e intuitiva.
- Estudio y compresión de las librerías de comunicación a través de SPI de Arduino

Fase 4: Diseño del protocolo de comunicaciones.

Duración: 30 días.

- Diseño del protocolo de comunicación que se va a utilizar en la transmisión de mensaje en este proyecto.
- Implementación del protocolo de comunicación en Arduino. Para ello nos basaremos en las librerías de Arduino y programación en C.

- Implementación del protocolo de comunicación en Raspberry PI. Para ello nos basaremos en la programación en Python 3.4.

Fase 5: Diseño de la aplicación en Pure Data.

Duración: 45 días.

- Lectura de trabajos previos.
- Estudio de las principales características del entorno de desarrollo de Pure Data.
- Realización de pruebas en un entorno controlado con un ordenador antes de crear el software definitivo en Raspbian.
- Implementación del patch de Pure Data que se encarga de la recepción de datos del programa.

Fase 6: Diseño de la electrónica.

Duración: 30 días.

- Estudio de las principales características del programa de diseño electrónico que se va a utilizar.
- Decisión de diseño definido según:
  - Tamaño final de la placa.
  - Complejidad en el enrutamiento.
  - Posibilidad de hacer todo el enrutamiento en una cara.
- Implementación de la solución final.

Fase 7: Diseño de la interfaz de usuario física.

Duración: 20 días.

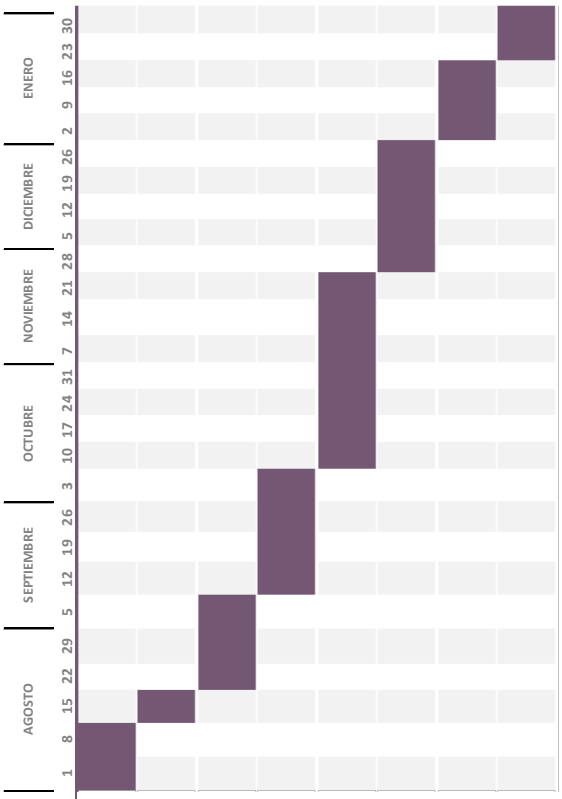
- Selección del material idóneo para tener una interfaz de usuario cómoda y a su vez vistosa, finalmente se escogió madera.
- Tratamiento de la misma con diferentes técnicas, tales como la aplicación de colorante, laca tapa poros y goma laca.
- Diseño e impresión de los accesorios de los actuadores.

Fase 8: Documentación.

Duración: 17 días.

- Asistencia a reuniones periódicas para la sincronización y actualización del proyecto.
- Generación de la presentación describiendo el trabajo realizado.
- Realización de la memoria final basada en los resultados obtenidos.

La duración total del proyecto ha sido de 184 días. Comenzando el día 3 de agosto de 2016 y finalizando el día 2 de febrero de 2017. Se ha realizado un diagrama de Gantt para ilustrar la planificación que se ha seguido en la realización del proyecto. Podemos ver dicha planificación en el diagrama mostrado en la figura 29.



	ACTIVIDAD	FECHA INICIO	FECHA FINALIZACIÓN	PREDECESORA	PORCENTAJE COMPLETADO
1	**Estudio del estado del arte**	01/08/2016	14/08/2016		100%
2	**Estudio del protocolo Enhance ShockBurst**	15/08/2016	21/08/2016	1	100%
3	**Desarrollo del prototipo electrónico**	22/08/2016	11/09/2016	2	100%
4	**Diseño del protocolo de comunicaciones**	12/08/2016	09/11/2016	3	100%
5	**Diseño de la aplicación en Pure Data**	10/10/2016	27/11/2016	4	100%
6	**Diseño de la electrónica**	28/11/2016	30/12/2016	5	100%
7	**Diseño de la interfaz de usuario física**	02/01/2017	15/01/2017	6	100%
8	**Documentación**	23/01/2017	02/02/2017	7	100%

Figura 29. Diagrama de Gantt.

5.2. Presupuesto.

NOMBRE	Cantidad	PVP (€)
Terminal para PCB	1	1.00
Header 8 pines hembra 2.54mm	1	2.21
Header 6 pines hembra 2.54mm	2	2.42
Tira pines 40ctos 2.54mm	1	0.67
Condensador 10 uF	5	0.18
Condensador 100 uF	2	0.05
Condensador 22 pF	2	0.13
Resistencia 1 KΩ	12	0.29
Resistencia 10 KΩ	7	0.17
Resistencia 1 MΩ	1	0.02
Led rojo	37	3.55
Diodo rectificador	1	0.05
Transistor bipolar NPN	6	0.51
Regulador tensión 5V	1	0.22
Regulador tensión 3V3	1	0.41
Encoder rotativo	1	2.50
Potenciómetro deslizante	1	1.94
Interruptor giratorio	2	13.50
ATMEGA328-PU	1	4.84
Multiplexor 8 canales	2	0.47
Módulo nRF24L01	2	7.87
Raspberry PI 3 model B	1	32.99

COSTE TOTAL: 77.52 € (44,53 € sin Raspi)

6. Conclusiones y trabajo futuro

6.1. Conclusiones.

Una de las razones por las que acepté este proyecto es la de poder integrar el mundo de Arduino con el de Raspberry PI. Ambas son plataformas que me interesan enormemente y quería realizar un proyecto en el que involucrara un poco de ambos mundos. Adicionalmente, incluir un programa de síntesis de audio como es Pure Data me ha abierto la puerta al conocimiento del audio digital, así como conocer una forma de programación en la que no se escribe una sola línea de código, sino que interactuando con elementos de la interfaz y manipulando sus características se pueden obtener los resultados esperados.

Este proyecto me ha ayudado a familiarizarme con una serie de herramientas que podré usar en un futuro. Hasta ahora no había utilizado prácticamente Altium, Pure Data o el compilador de Python, y gracias a este proyecto he aprendido a poder desarrollar una serie de habilidades de diseño electrónico y de programación que no tenía anteriormente.

El mayor de los problemas que he tenido a la hora de realizar el proyecto ha sido programar la conexión del módulo de radio frecuencia nRF24L01 con la Raspberry PI. Esto es debido a que debido a que en el sistema operativo Raspbian los externals no se compilan igual que en Windows, por lo que el external compilado en Windows no me servía. Finalmente, esto se pudo arreglar creando una red virtual y utilizando las herramientas que proporciona Pure Data para redes TCP y UDP.

Se ha logrado un sistema muy intuitivo y sencillo, de manera que cualquier usuario, sin necesidad de conocimientos musicales previos, pueda usarlo.

Vamos a recuperar los objetivos expuestos en la introducción, explicando cómo se ha logrado la consecución de los mismos.

- Diseñar e implementar un sintetizador de audio digital con controladores inalámbricos modulares.
  Con este objetivo se buscaba que el módulo que se diseñara pudiera ser reutilizable, es decir, que dependiendo de la programación del micro controlador realizara diversas funciones aprovechando el mismo hardware. Este punto se ha cumplido, ya que disponemos de tres elementos clave (los leds, el potenciómetro deslizante y el encoder) que pueden configurar a nuestro gusto.
- Conseguir un sistema simple, que cualquier usuario pueda utilizar sin un entrenamiento previo.
  Para la consecución de este objetivo se buscaron los componentes más utilizados en una mesa de mezclas, como lo son el potenciómetro deslizante (realizando la función de fader) o el encoder, haciendo que la apariencia sea muy similar. A su vez, se va a disponer de una leyenda en cada uno de los leds, de manera que una persona experta, podrá manejar el sistema sin problema, y una persona novata tendrá la ayuda continua de dicha leyenda.
- Diseñar un sistema de síntesis de audio digital que sea multiplataforma.
  Esto se ha conseguido gracias a las plataformas utilizadas: Arduino, Raspberry Pi y Pure Data. Gracias a ello, se puede utilizar tanto el programa de Pure Data como el de Arduino sin tocar nada, y el de la Raspberry Pi compilando el código en la plataforma deseada.

En general, trabajar en este proyecto ha sido una gran experiencia debido a que he aprendido muchas cosas sobre diferentes plataformas de hardware y software libre como Raspberry PI y Arduino, así como mejorar mis conocimientos de programación en C e introducirme en el mundo de Python.

6.2. Trabajo futuro.

Este proyecto tiene una serie de elementos en los que se puede trabajar en un futuro y mejoraría sustancialmente el mismo.

Lo primero de todo sería hacer un estudio detallado del consumo de cada módulo y estimación del tiempo que debe estar en reposo o funcionamiento para maximizar la duración de la batería.

Otro punto a mejorar sería la síntesis. En este proyecto se ha realizado un programa de síntesis de audio funcional, de manera que se pueda utilizar con un único módulo de control. La idea, es utilizar este programa como base para poder posteriormente utilizar un módulo para una serie de funciones y que con dos o tres módulos de control ser capaz de controlar una mesa de mezclas completa.

El tercer punto en el que se puede trabajar en un futuro es la reducción de los costes de hardware, tanto en superficie de impresión, como el uso de componentes SMD en lugar de utilizar convencionales. De esta manera se lograría realizar todo el diseño en una sola PCB.

Enlazando con el tercer punto, y debido a los problemas surgidos para testear la placa, se deberían dejar una serie de puntos de test y de programación. De esta manera, colocando una serie de pines donde corresponda se puede ver el valor de esa pista, así como poder programarla con un programador externo.

El último punto a tener en cuenta a evolucionar son las comunicaciones. Pese a que el diseño del sistema está pensado para aguantar hasta seis módulos de control, no se rige por ningún protocolo de música específico. Sería bueno adaptar nuestro sistema para poder conectarle a nuestros controladores actuadores externos y que puedan comunicarse con nuestro sistema sin añadir plugins.

7. Referencias

7.1. Referencias bibliográficas.

Jeremy Blum. “Arduino a fondo” 2014.

Ferrán Fabregas. “Aprender Raspberry con 100 ejercicios prácticos” 2016.

Digital audio – Wikipedia, the free encyclopedia.
https://en.wikipedia.org/wiki/Digital_audio

Síntesis y generación digital de sonido – ccapitalia.net
http://www.ccapitalia.net/reso/articulos/audiodigital/pdf/09-Sintesis.pdf

Raspberry Pi Tutorials – Youtube
https://www.youtube.com/watch?v=tB_GebqaGas&t=25s&list=PLNnwglGGYoTvy37TSGFlv-aFkpg7owWrE&index=36

Micro Tutorial sobre Síntesis con Alex Martín. – Youtube
https://www.youtube.com/watch?v=PPRPGCRkxnA&list=RDPPRPGCRkxnA

[1] Pure Data – Floss Manuals
http://write.flossmanuals.net/pure-data/introduction2/

[2] The house of Synthesis
http://www.houseofsynthesis.com/

[3] Raspberry Pi.
https://www.raspberrypi.org/

[4] Arduino – Wikipedia, the free encyclopedia.
https://en.wikipedia.org/wiki/Arduino

[5] MIDI – Wikipedia, the free encyclopedia.
https://en.wikipedia.org/wiki/MIDI

[6] Arduino.
https://www.arduino.cc/

[7] Módulo nRF20L01 – Prometec
http://www.prometec.net/

[8] nRF20L01 Datasheet – Nordic Semiconductor.
https://en.wikipedia.org/wiki/Digital_audio

[9] Pure Data – Wikipedia, the free encyclopedia.
https://en.wikipedia.org/wiki/Pure_Data

[10] Altium Designer.
http://www.captura-el.com/79-lineas/altium/71-altium

[11] How to design a virtual sinthesizer with Pure Data – Youtube
https://www.youtube.com/watch?v=2LgiDwJXs04&index=1&list=PLAi--S_bkmmupoOrqr8RSmK4u8_hq1wpp

7.2. Índice de figuras.

Anexo A – Software de Arduino.

En el anexo A se va a escribir el código utilizado tanto en el módulo de Arduino como en la Raspberry Pi. Se ha seguido una estructura jerárquica, teniendo una función principal en bucle y una serie de funciones que son llamadas por esta.

a) Arduino – Función principal

```
#include "Comunicacion.h"
#include "lectura_actuadores.h"
#include "lectura_encoder.h"
#include "interrupcion.h"

extern bool f_100ms;
extern bool cambioDatos;
extern int varEncoder;
extern int fila;
extern int columna;

int lastVarEncoder = 0;
int aux = 0;

void setup() {
  // put your setup code here, to run once:
  setup_comunicacion();
  setup_actuador();
  setup_encoder();
  setup_interrupcion();
  Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
  if (cambioDatos){
    aux = varEncoder/4;
    if (aux != lastVarEncoder){
      envio_datos();
      cambioDatos = false;
      varEncoder = 0;
      lastVarEncoder = varEncoder/4;
    }
  }
//  lectura_encoder();
  if(f_100ms){
    f_100ms = false;
    lee_sw_fila();
    lee_sw_columna();
    lee_x10();
    Serial.print("Fila: ");
    Serial.print(fila);
    Serial.print("\t Columna: ");
    Serial.println(columna);
  }
}
```

b) Arduino – Función comunicación

```
#include<Arduino.h>
#include<SPI.h>
#include<RF24.h>


#define MAX_outBUFFER_SIZE 16
#define MAX_nrf24BUFFER_SIZE 32

#define outBUFFER_SIZE 4
#define nrf24BUFFER_SIZE 8

int outBuffer[outBUFFER_SIZE]= {0};
char num_placa;
int fila;
int columna;
int valor;

// ce, csn pins
RF24 radio(9,10);

void setup_comunicacion(){
  radio.begin();
  radio.setPALevel(RF24_PA_MAX);
  radio.setChannel(0x60);
  radio.openWritingPipe(0xF0F0F0F0E1LL);
  radio.enableDynamicPayloads();
  radio.powerUp();
}

void envio_datos(){
  outBuffer[1] = 0;
  outBuffer[2] = 0;
  outBuffer[0] = 0;
  outBuffer[3] = 0;
//  // transmit data over SPI to the NRF module
  radio.write(outBuffer, nrf24BUFFER_SIZE);
//  delay(500);
}
```

c) Arduino – Función interrupción

```
#include <TimerOne.h> //Librería del timer

bool f_100ms = false;

void funcion_100ms(){
  f_100ms = true;
}

void setup_interrupcion(){
  //Configuracion de las interrupciones TIMER
  Timer1.initialize(100*(10^3));         // Dispara cada x us
  Timer1.attachInterrupt(funcion_100ms); // Activa la interrupcion y
la asocia a ISR_Blink
}
```

d) Arduino – Función lectura de los actuadores

```
#include<Arduino.h>

#define pinColumna  A3
#define pinFila     A2
#define pinX10      A1

extern int fila;
extern int columna;
extern int potX10;

int contador;

void setup_actuador(){
  pinMode(pinColumna,INPUT);
  pinMode(pinFila,INPUT);
  pinMode(pinX10,INPUT);
}

void lee_sw_fila(){
  fila = analogRead(pinFila);
  //fila = map(fila,0,1023,0,5);
  delay(30);
}

void lee_sw_columna(){
  columna = analogRead(pinColumna);
  //columna = map(columna,0,1023,0,5);
  delay(30);
}

void lee_x10(){

}
```

e) Arduino – Función lectura del encoder

```
#include <Arduino.h>

bool cambioDatos = false;

volatile int varEncoder = 0;
volatile long lastEncoded = 0;

long lastInt = 0;
long lasEncoderValue = 0;

int lastMSB = 0;
int lastLSB = 0;

#define encoderPinA  2
#define encoderPinB  3

void doEncoder() {

  //if((millis() < lastInt ) < 400)
  //{
    int MSB = digitalRead(encoderPinA);
    int LSB = digitalRead(encoderPinB);

    int encoded = (MSB << 1) | LSB;
    int sum = (lastEncoded << 2) | encoded;

    if(sum == 0b1101 || sum == 0b0100 || sum == 0b0010 || sum ==
0b1011) {varEncoder--;cambioDatos = true;}
    if(sum == 0b1110 || sum == 0b0111 || sum == 0b0001 || sum ==
0b1000) {varEncoder++;cambioDatos = true;}

    lastInt = millis();
    lastEncoded = encoded;

  //}
}

void setup_encoder(){
  pinMode(encoderPinA, INPUT);
  digitalWrite(encoderPinA, HIGH);       // turn on pullup resistor
  pinMode(encoderPinB, INPUT);
  digitalWrite(encoderPinB, HIGH);       // turn on pullup resistor

  attachInterrupt(0, doEncoder, CHANGE);  // encoder pin on interrupt
0 - pin 2
  attachInterrupt(1, doEncoder, CHANGE);  // encoder pin on interrupt
0 - pin 2

}
```

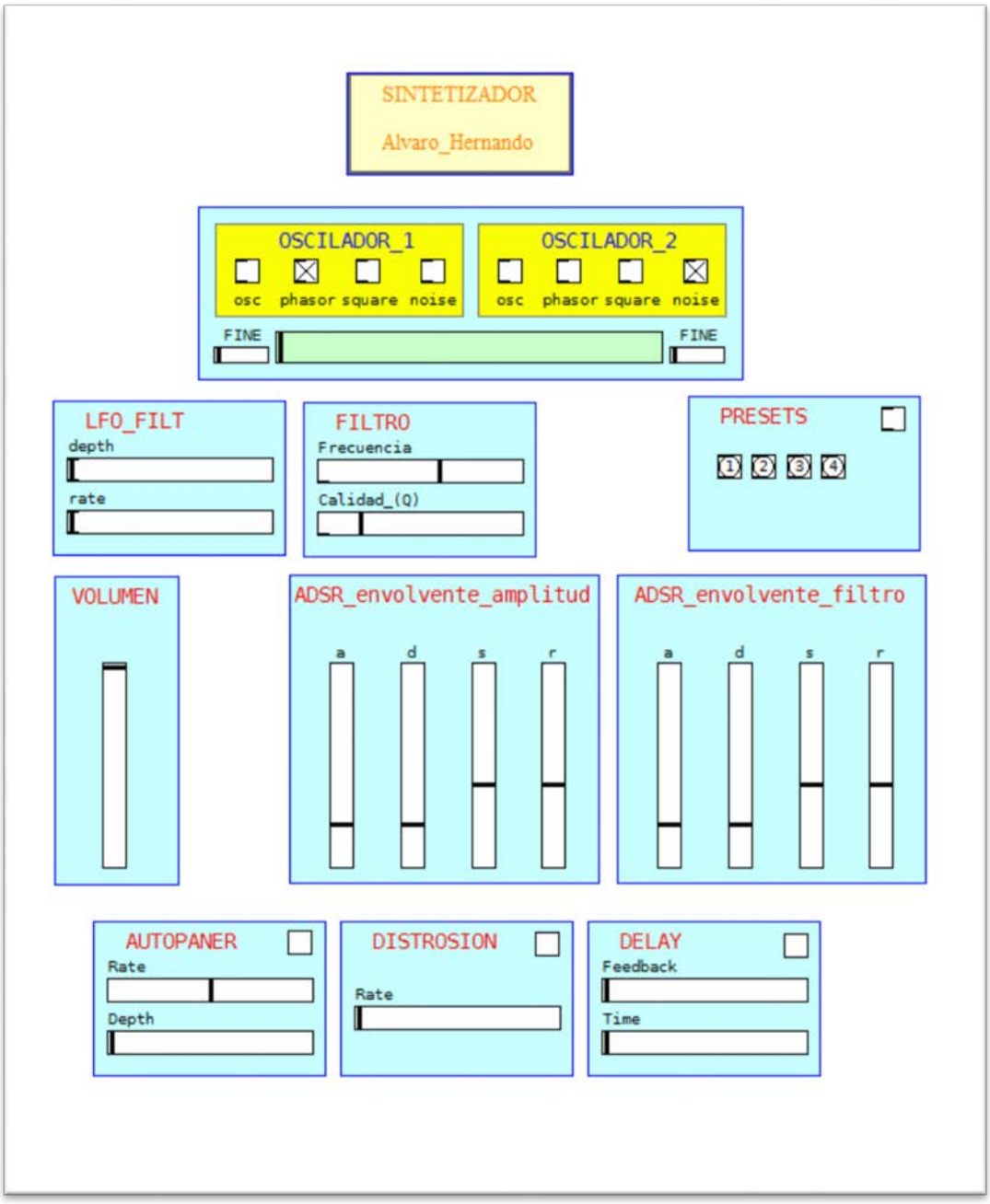
f) Raspberry Pi – Lectura del módulo de RF y envío a Pure Data

```
import RPi.GPIO as GPIO
from lib_nrf24 import NRF24
import time
import spidev
import socket
import sys

GPIO.setmode(GPIO.BCM)
pipes = [[0xE8, 0xE8, 0xE0, 0xE0, 0xE1],[0xF0,0xF0,0xF0,0xF0,0xE1]]

radio = NRF24(GPIO, spidev.SpiDev())
radio.begin(0,17)
radio.setPayloadSize(32)
radio.setChannel(0x60)

radio.setDataRate(NRF24.BR_1MBPS)
radio.setPALevel(NRF24.PA_MIN)

radio.setAutoAck(True)
radio.enableDynamicPayloads()
radio.enableAckPayload()

radio.openReadingPipe(1, pipes[1])
radio.printDetails()

radio.startListening()

s=['socket00', 'socket01', 'socket02']
t=['socket10', 'socket11', 'socket12']

for i in [0, 1, 2]:
    s[i] = socket.socket()
    s[i].connect(("localhost",3000+i*2))
    t[i] = socket.socket()
    t[i].connect(("localhost",3001+i*2))

while True:

    while not radio.available(0):
        time.sleep(1/100)
    receivedMessage=[]
    radio.read(receivedMessage, radio.getDynamicPayloadSize())
    print("Recibido: {}".format(receivedMessage))
    print("Mensaje   1:
          {}".format(receivedMessage[0]+receivedMessage[1]*128))
    print("Mensaje   2:
          {}".format(receivedMessage[2]+receivedMessage[3]*128))
    print("Mensaje   3:
          {}".format(receivedMessage[4]+receivedMessage[5]*128))

    for i in [0, 1, 2]:
        a = chr(receivedMessage[2*i])
        s[i].send(a.encode('utf-8'))
        b = chr(receivedMessage[1+2*i])
        t[i].send(b.encode('utf-8'))
```

Anexo B – Programa en Pure Data.

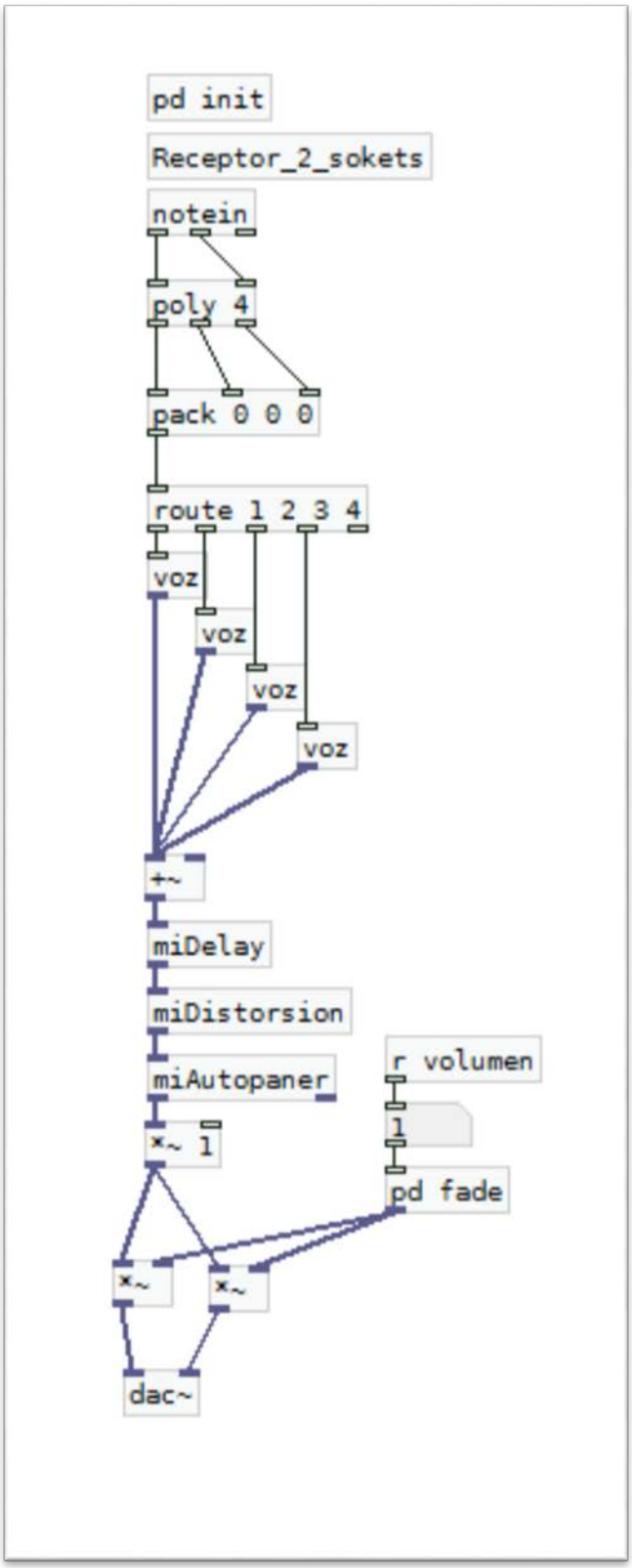
En el anexo B se van a representar todos los patches principales del programa realizado en Pure Data. Se expondrá finalmente la abstracción voz en detalle mostrando todos los subpatches que lo contienen. Se llama abstracción al patch de Pure Data que no está generado en el mismo proyecto, sino que se llama desde el cómo su fuera una función, pudiendo escribirlo tantas veces como sea necesario.

a) Pantalla principal – Función principal

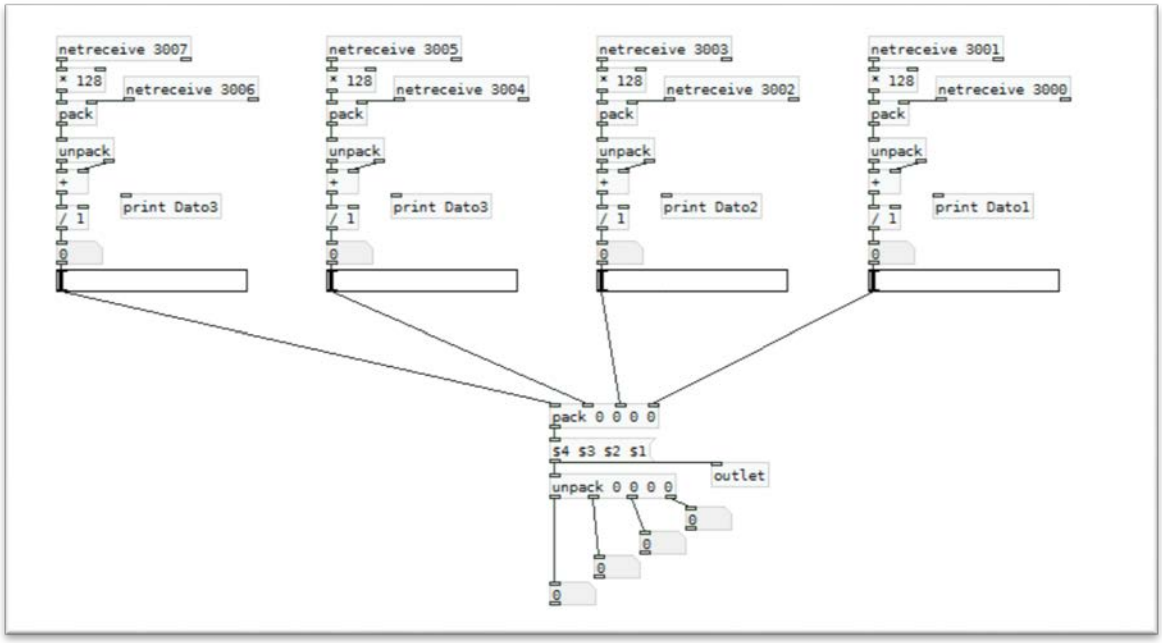


Patch Pure Data 1. Interfaz de usuario

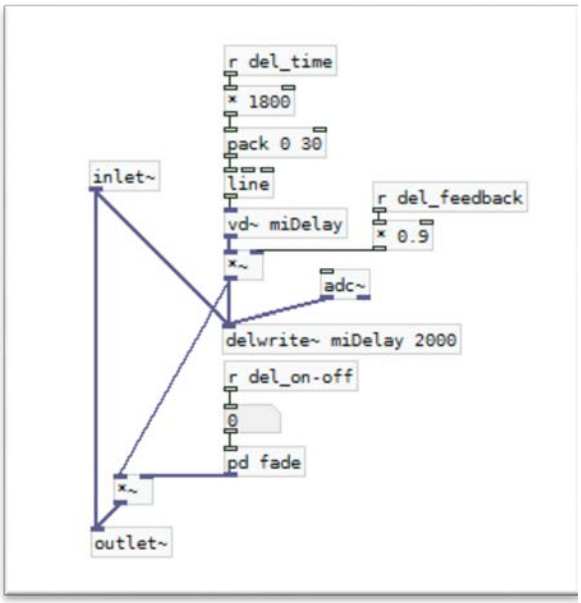
b) Estructura del sintetizador



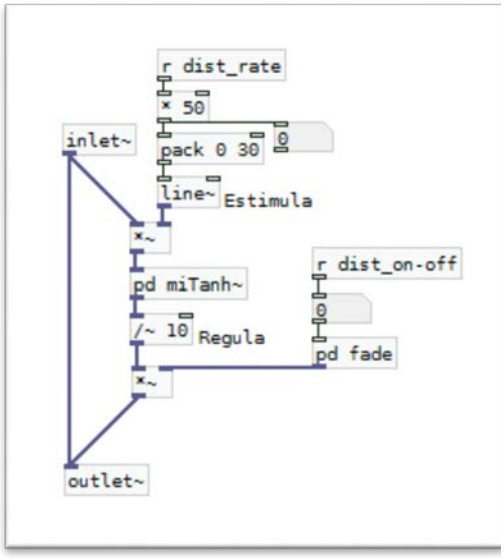
Patch Pure Data 2. Estructura del sintetizador



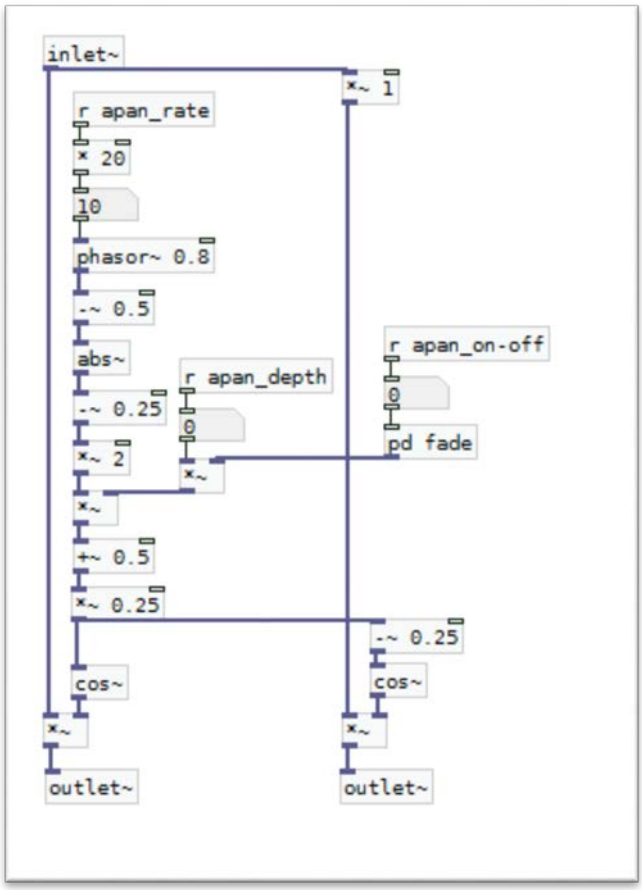
Patch Pure Data 3. Subpatch Recepción_2_socket



Patch Pure Data 4. Subpatch miDelay

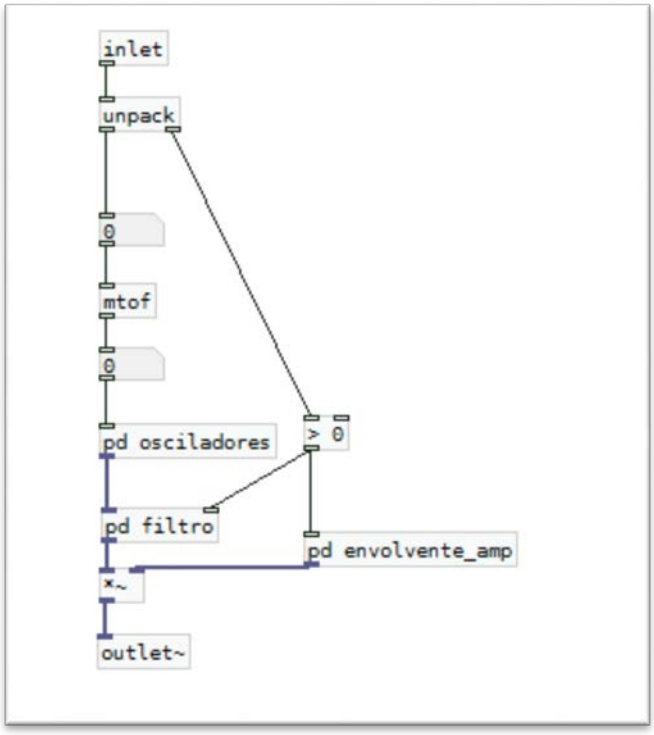


Patch Pure Data 5. Subpatch miDistorsión

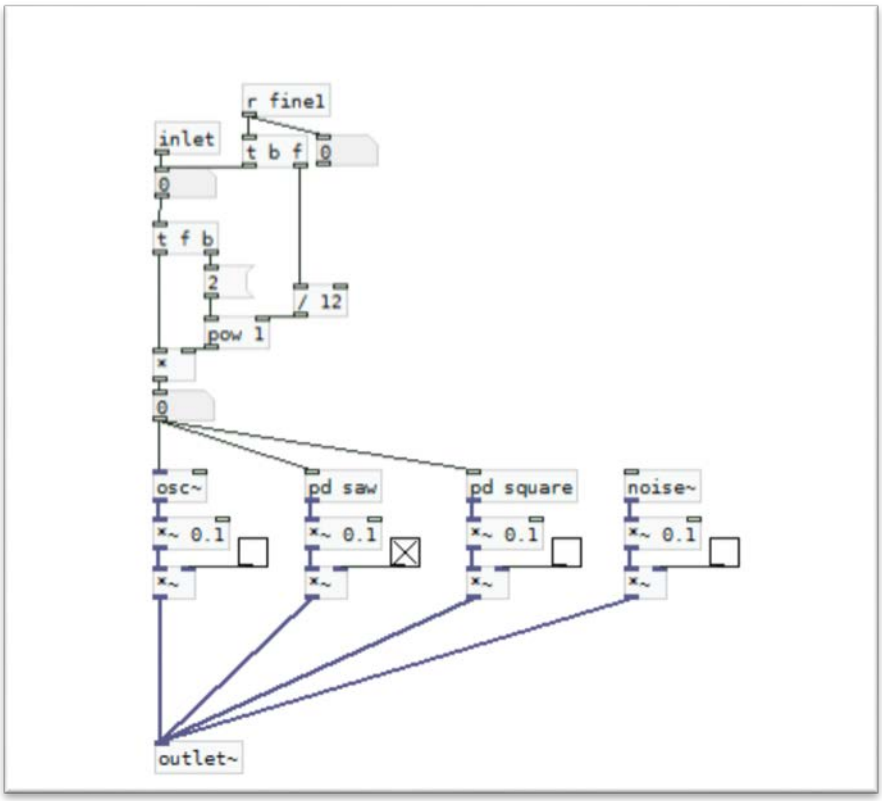


Patch Pure Data 6. Subpatch miAutopaner

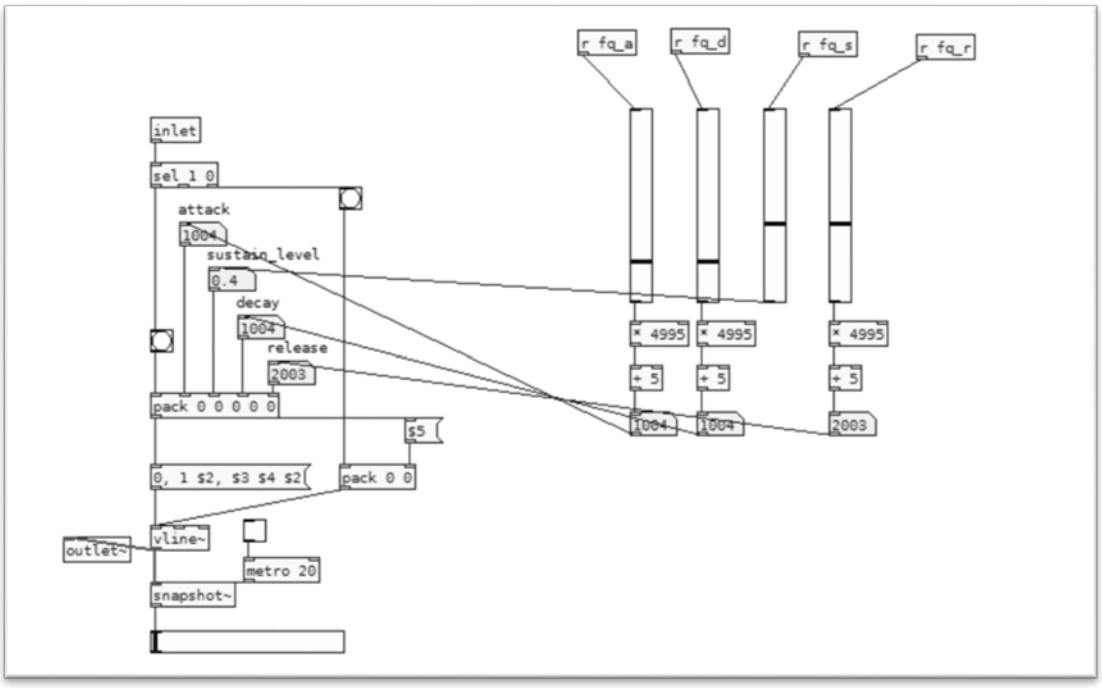
c) Abstracción voz



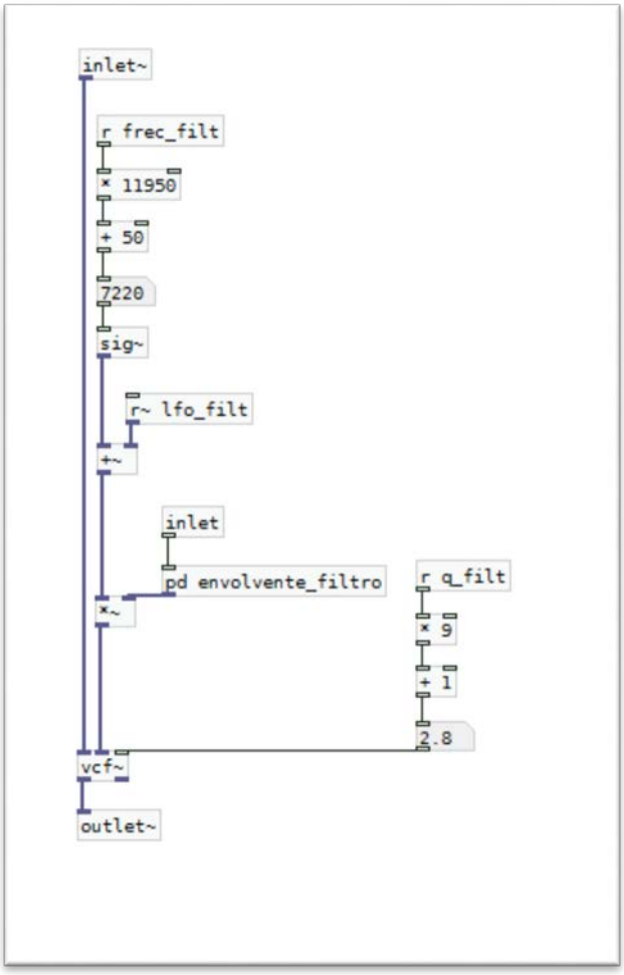
Patch Pure Data 7. Abstracción voz



Patch Pure Data 8. Subpatch osciladores



Patch Pure Data 9. Subpatch envolvente



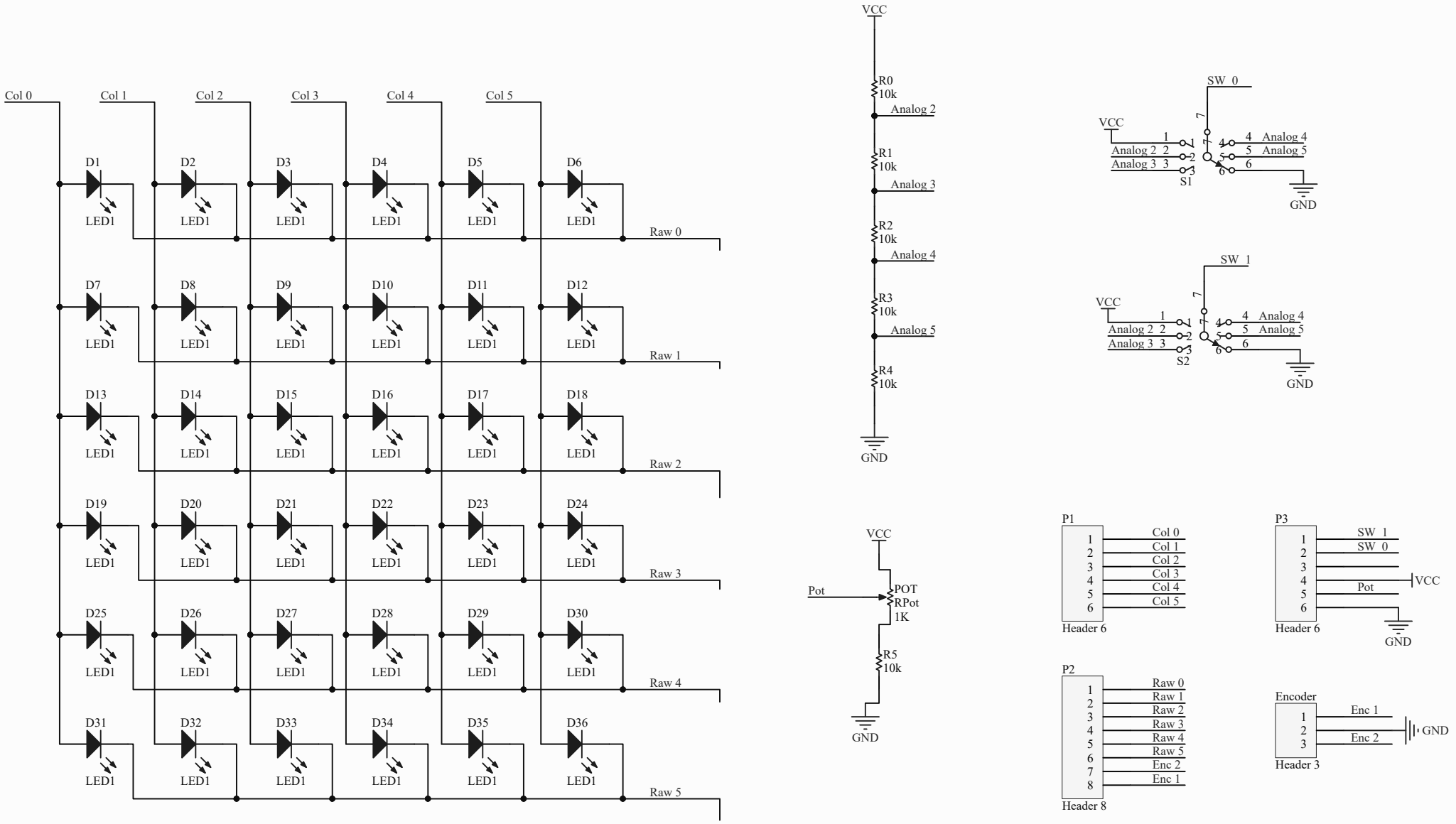
Patch Pure Data 10. Subpatch filtro (LFO)

Anexo C – Diseños de Altium.

En el anexo C se van a mostrar los esquemáticos completos junto a las caras top y bottom de los diseños de cada placa.

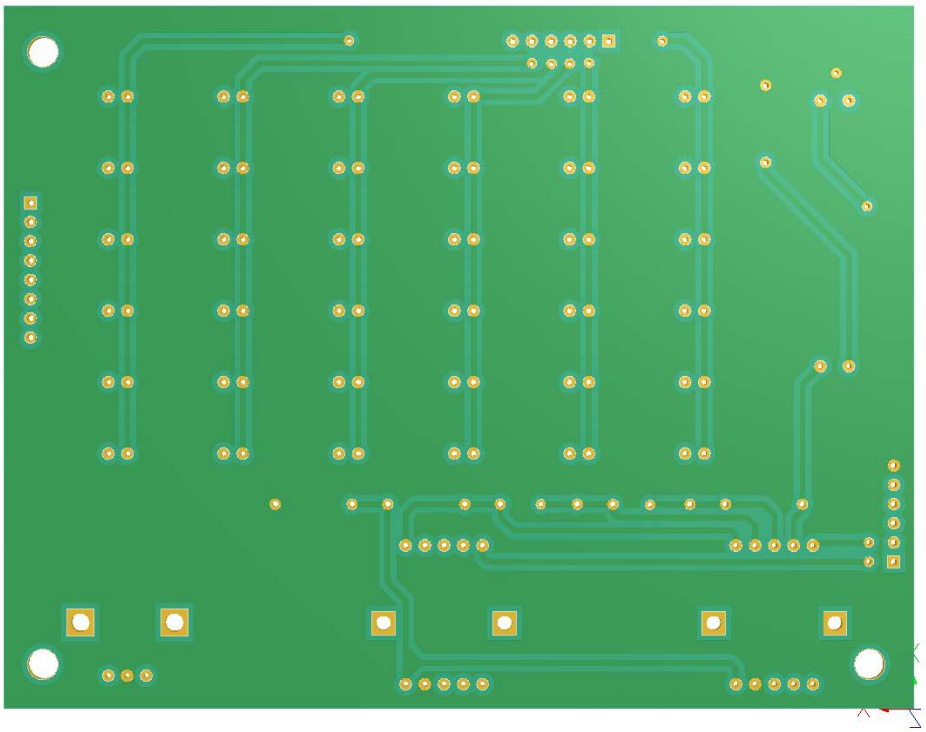
La estructura va a ser la siguiente: Primero se mostrará a pantalla completa el esquemático de cada PCB, para posteriormente mostrar tanto la cara Bottom como la Top de cada PCB.

Por motivos de visibilidad, se ha optado por mostrar el esquemático en una página y las vistas de la PCB en otra.

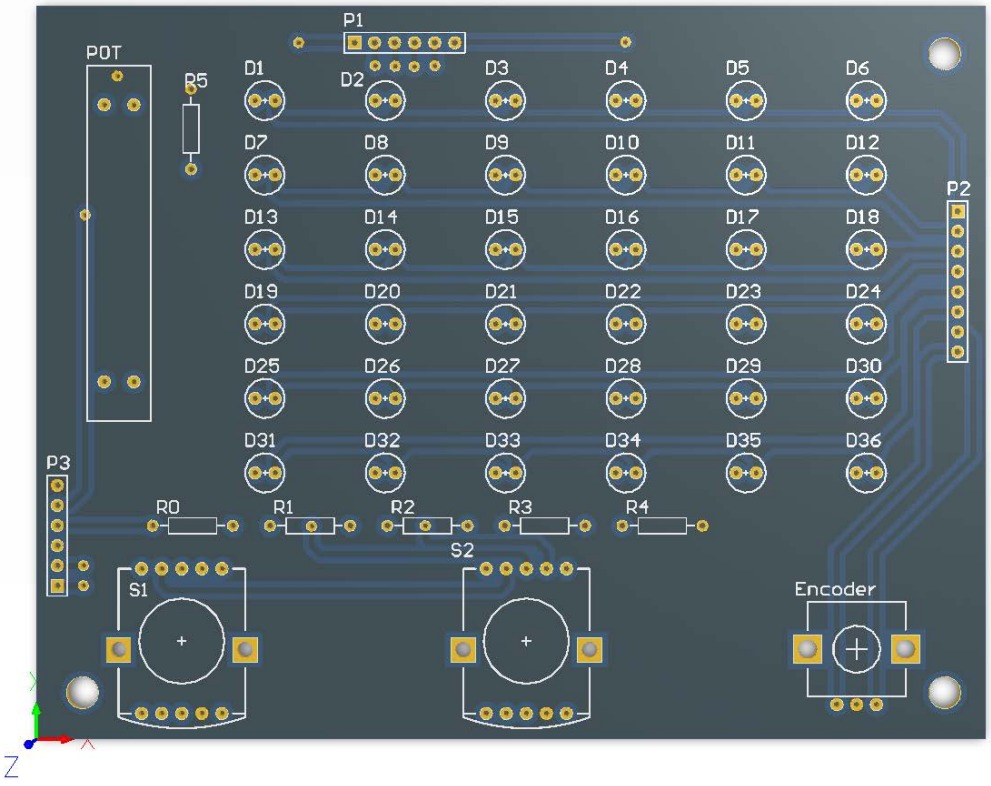


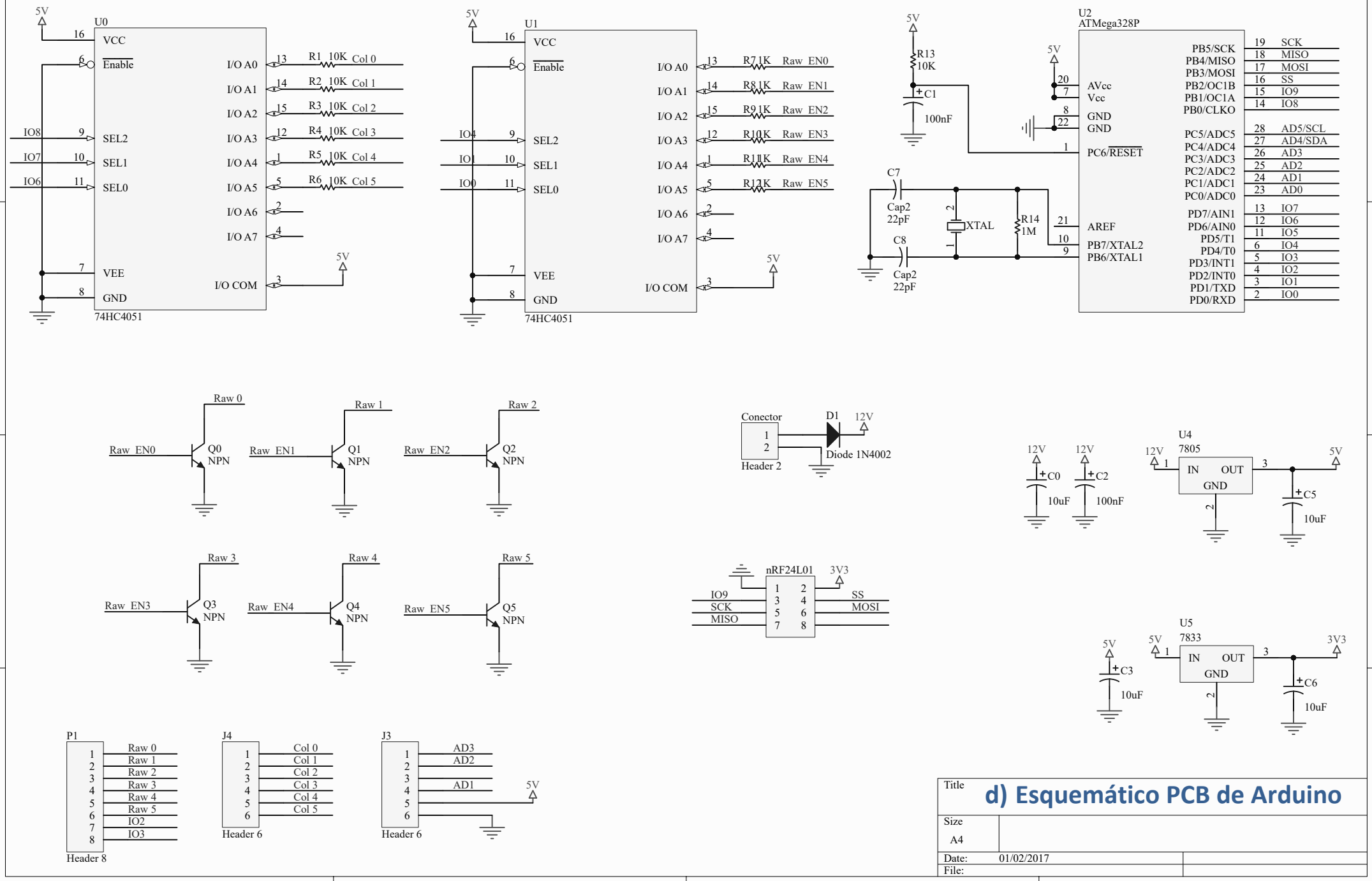
Title	a) Esquemático PCB de control
Size	A4
Date:	01/02/2017

b) Bottom layer PCB de control

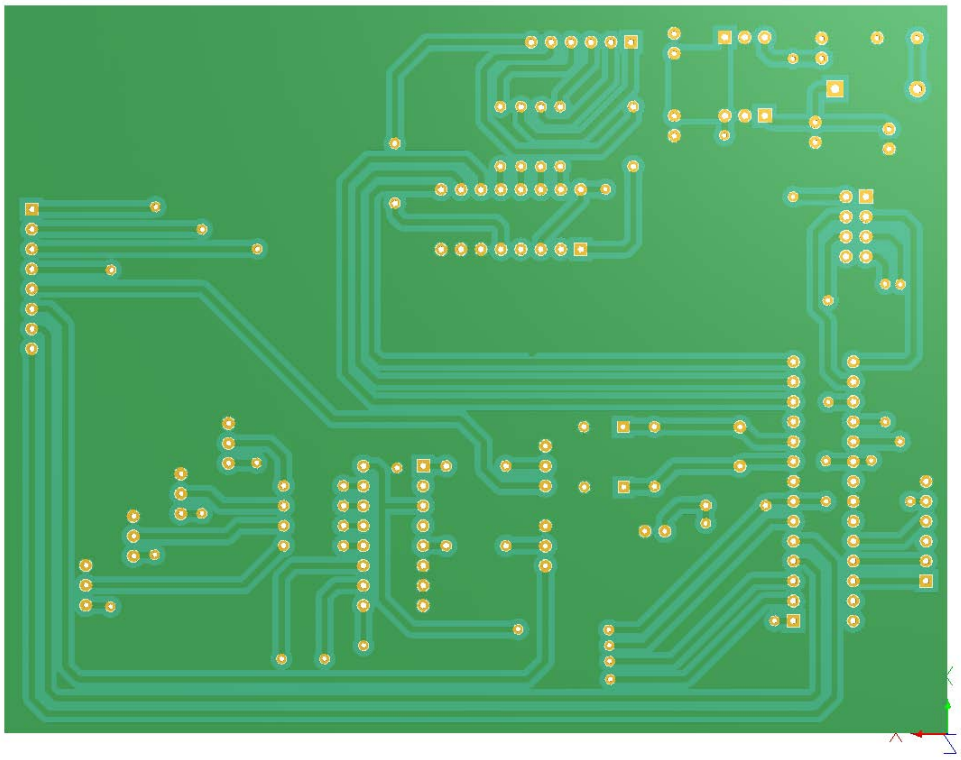


c) Top Layer PCB de control

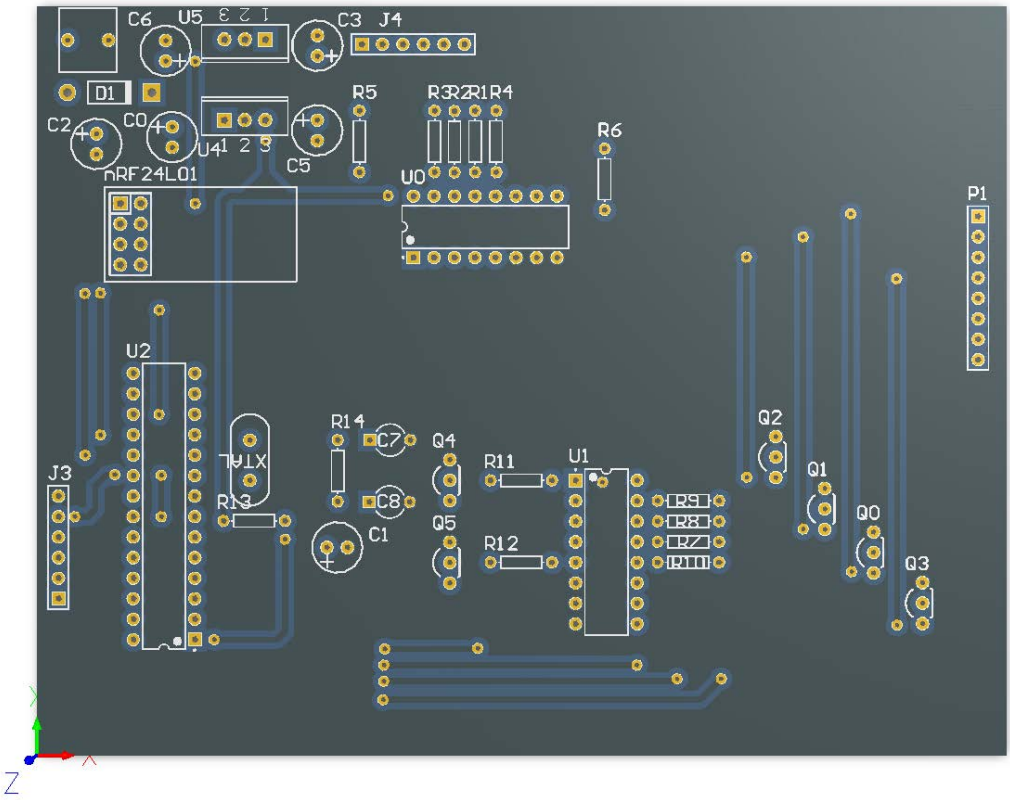


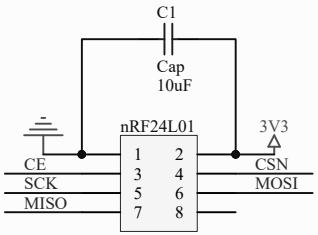

U0
74HC4051
VCC
Enable
SEL2
SEL1
SEL0
VEE
GND
I/O A0
I/O A1
I/O A2
I/O A3
I/O A4
I/O A5
I/O A6
I/O A7
I/O COM
R1 10K Col 0
R2 10K Col 1
R3 10K Col 2
R4 10K Col 3
R5 10K Col 4
R6 10K Col 5
IO8
IO7
IO6
5V
U1
74HC4051
R7 1K Raw EN0
R8 1K Raw EN1
R9 1K Raw EN2
R10 1K Raw EN3
R11 1K Raw EN4
R12 1K Raw EN5
IO4
IO1
IO0
U2
ATMega328P
R13 10K
C1 100nF
C7 Cap2 22pF
C8 Cap2 22pF
XTAL
R14 1M
AVcc
Vcc
GND
PC6/RESET
AREF
PB7/XTAL2
PB6/XTAL1
PB5/SCK
PB4/MISO
PB3/MOSI
PB2/OC1B
PB1/OC1A
PB0/CLKO
PC5/ADC5
PC4/ADC4
PC3/ADC3
PC2/ADC2
PC1/ADC1
PC0/ADC0
PD7/AIN1
PD6/AIN0
PD5/T1
PD4/T0
PD3/INT1
PD2/INT0
PD1/TXD
PD0/RXD
Raw 0
Raw 1
Raw 2
Raw 3
Raw 4
Raw 5
Raw_EN0
Raw_EN1
Raw_EN2
Raw_EN3
Raw_EN4
Raw_EN5
Q0 NPN
Q1 NPN
Q2 NPN
Q3 NPN
Q4 NPN
Q5 NPN
Conector
Header 2
D1
Diode 1N4002
12V
U4
7805
IN
OUT
GND
C0 10uF
C2 100nF
C5 10uF
nRF24L01
3V3
IO9
SCK
MISO
SS
MOSI
U5
7833
C3 10uF
C6 10uF
P1
Header 8
J4
Header 6
J3
Header 6
AD3
AD2
AD1
Col 0
Col 1
Col 2
Col 3
Col 4
Col 5
IO2
IO3
Title
d) Esquemático PCB de Arduino
Size
A4
Date: 01/02/2017
File:

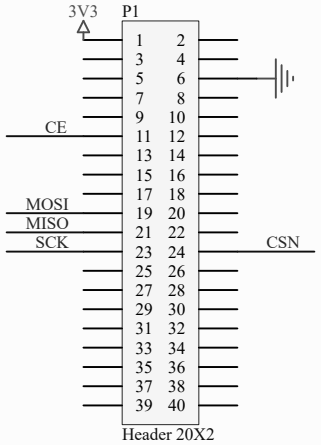
e) Bottom layer PCB de Arduino



f) Top Layer PCB de Arduino

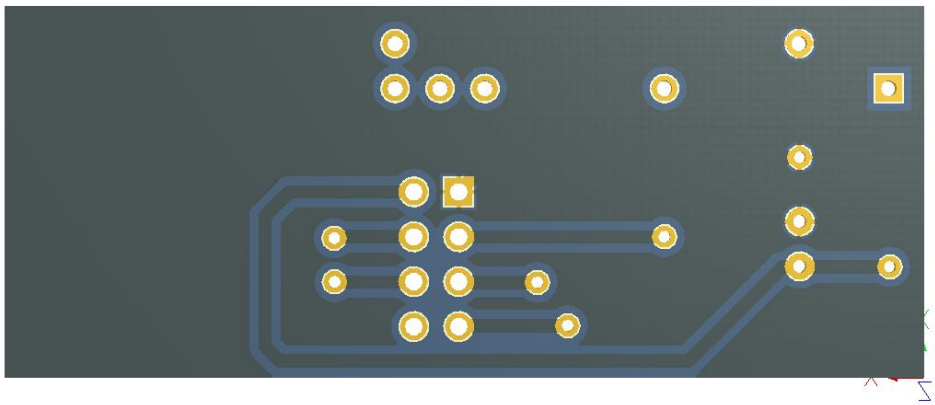






Title	g) Esquemático PCB de Raspberry Pi
Size	A4
Date:	01/02/2017
Sheet	of

h) Bottom layer PCB de Raspberry Pi



i) Top Layer PCB de Raspberry Pi