NESLA Coil Mock Design Review:

Raspberry Pi Model B

SoC: Broadcom BCM2837
CPU: 4× ARM Cortex-A53, 1.2GHz
GPU: Broadcom VideoCore IV
RAM: 1GB LPDDR2 (900 MHz)
Networking: 10/100 Ethernet, 2.4GHz 802.11n wireless
Bluetooth: Bluetooth 4.1 Classic, Bluetooth Low Energy
Storage: microSD
GPIO: 40-pin header, populated
Ports: HDMI, 3.5mm analogue audio-video jack, 4× USB 2.0, Ethernet, Camera Serial Interface (CSI), Display Serial Interface (DSI)

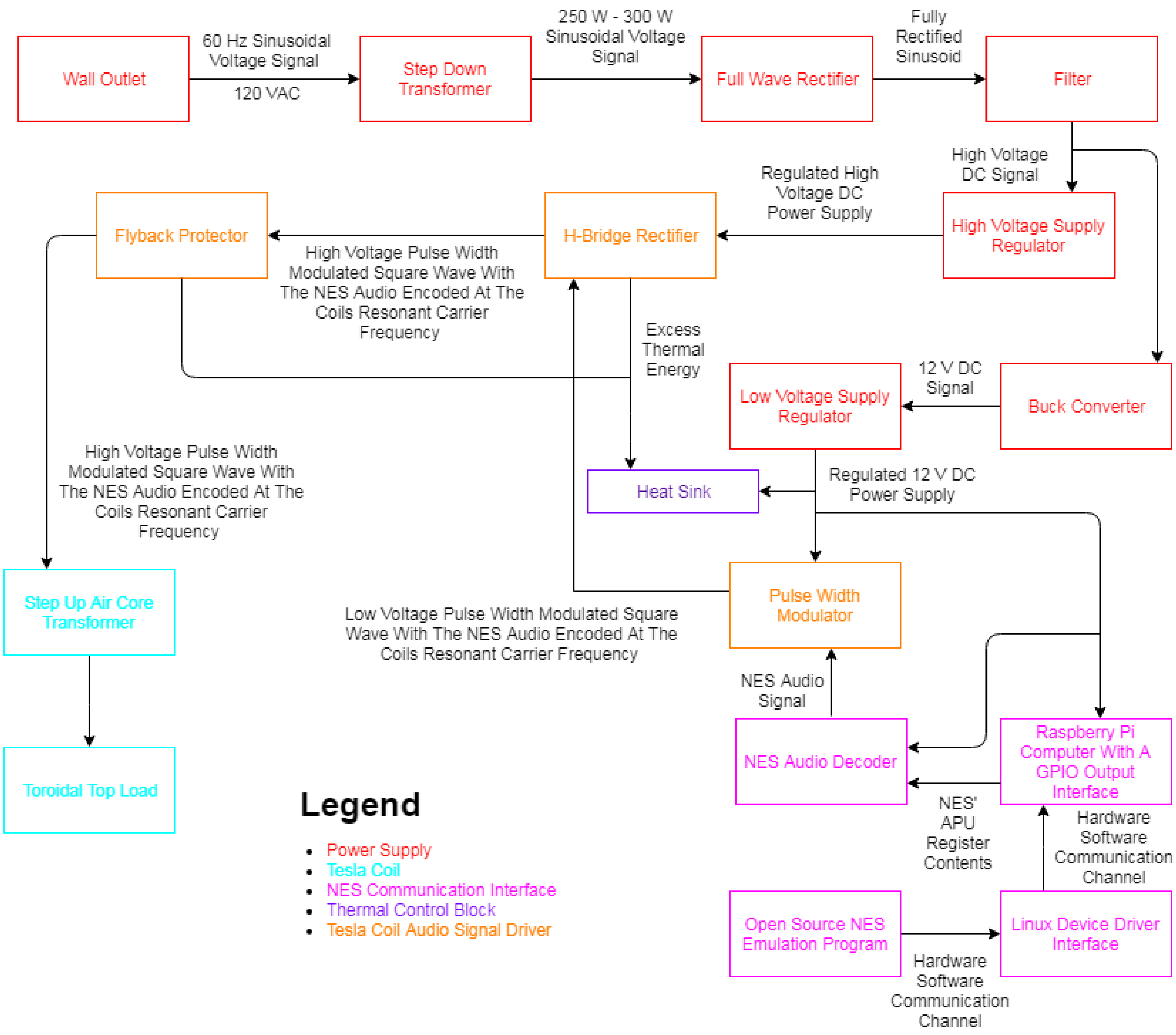
https://www.raspberrypi.org/magpi/raspberry-pi-3-specs-benchmarks/

The Raspberry Pi will be running a distribution of Linux, with support for Simple DirectMedia Layer (SDL), for creating a multimedia interface our NES emulator can use to play various ROM sounds and their accompanying graphics.

The NES Emulator must be open source, and compatible with SDL1, SDL2 and Linux. A few candidate emulators are:

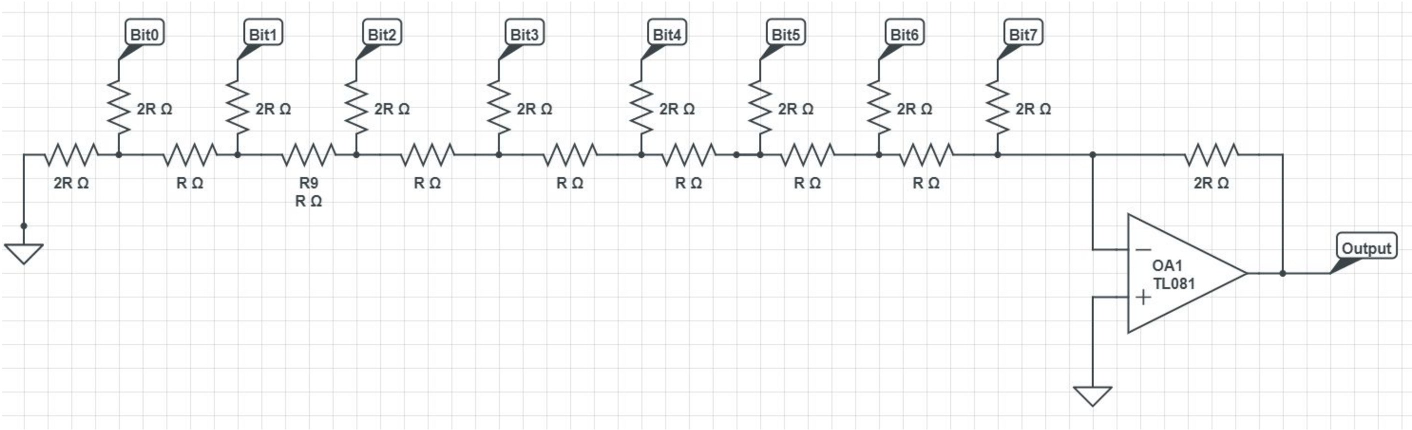
- https://github.com/ahefner/tenes
- https://github.com/AndreaOrru/LaiNES
- https://github.com/amhndu/SimpleNES
- https://github.com/randrews7/rnes

Although they are different emulators their dependency on the SDL library to create sound makes the overall mechanism for exporting a digital audio signal the same. We will take advantage of this similarity, by creating a pluggable intermediate software layer that sits between SDL's audio output buffer, and the stream of sound samples being fed into it. Resting our software design dependency assumptions on SDL rather than a specific emulator, gives us the freedom to choose which specific emulator we use based upon, how well the emulator performs when implemented in the Raspberry Pi environment. The software layer we are creating will be transparent and will take on the task of duplicating the real-time audio samples produced by the APU and re-routing the amplitude sample duplicates into an alternate buffer. Using an alternate buffer, we now have a data source to write our GPIO pins with. To write to the Raspberry Pi's GPIO pins, we will link-in a library called Wiring Pi during the compilation process. WiringPi is a library written in C, that is compatible with the Broadcom BCM2837 SoC used in the Raspberry Pi (http://wiringpi.com/). Using WiringPi, we can gain access to the 40 GPIO pins on the Raspberry Pi board. At this point, we now have a way of exporting a real-time digital audio signal produced by the NES into our coil circuitry.


Wall Outlet
60 Hz Sinusoidal Voltage Signal
120 VAC
Step Down Transformer
250 W - 300 W Sinusoidal Voltage Signal
Full Wave Rectifier
Fully Rectified Sinusoid
Filter
High Voltage DC Signal
High Voltage Supply Regulator
Regulated High Voltage DC Power Supply
H-Bridge Rectifier
High Voltage Pulse Width Modulated Square Wave With The NES Audio Encoded At The Coils Resonant Carrier Frequency
Flyback Protector
Excess Thermal Energy
Buck Converter
12 V DC Signal
Low Voltage Supply Regulator
Heat Sink
Regulated 12 V DC Power Supply
High Voltage Pulse Width Modulated Square Wave With The NES Audio Encoded At The Coils Resonant Carrier Frequency
Step Up Air Core Transformer
Toroidal Top Load
Low Voltage Pulse Width Modulated Square Wave With The NES Audio Encoded At The Coils Resonant Carrier Frequency
Pulse Width Modulator
NES Audio Signal
NES Audio Decoder
NES' APU Register Contents
Raspberry Pi Computer With A GPIO Output Interface
Hardware Software Communication Channel
Open Source NES Emulation Program
Hardware Software Communication Channel
Linux Device Driver Interface
Legend
Power Supply
Tesla Coil
NES Communication Interface
Thermal Control Block
Tesla Coil Audio Signal Driver

Digital Audio Converter

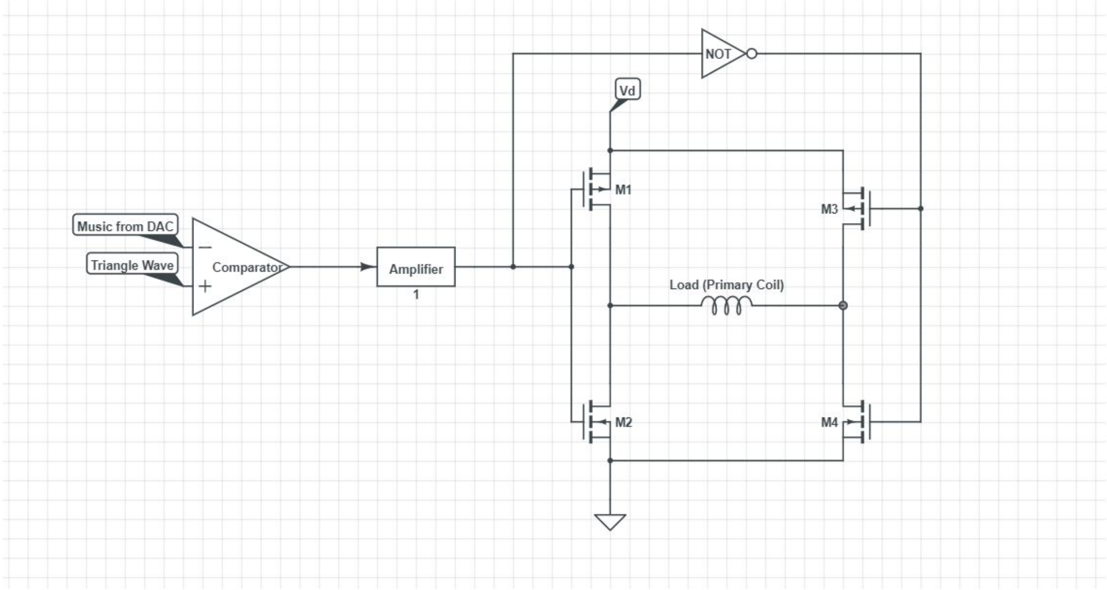
At this point in the data path, a sequence of bits is being stream-lined from the GPIO on the Raspberry Pi. The purpose of the Digital Audio Converter is to process the digital sound signal to a corresponding analog signal. The digital signal supply voltage needs to be considered in the design of the DAC, as well as the resistance values. These values will need to match the specifications that can be accurately handled by the op amp for correct operation. After the DAC stage the output should be an analog signal that steps from 0 to a negative max. An offset added to this value will give a good range, allowing the PWM circuit to process.



Pulse Width Modulator

This component will modulate our processed analog signal into a pulse width modulated triangle wave. To perform pulse width modulation of our input analog signal, we will use a 555 timer circuit. Specifically, the NA555 as its output specifications grant us the ability to modulate pulse widths for the entire range of audible frequencies. Since we need to drive the coil, at a specified resonant frequency within the kilohertz range, we will run another NA555 in parallel to the pulse width modulating NA555. This NA555 will output a triangle wave that has a fixed frequency, and matches the resonant frequency of our coil. Using TTL AND gates, we will mask the pulse width modulated triangle wave with the resonant triangle wave, so that the resonant frequency triangle wave acts as a carrier wave for our sound data.

http://www.ti.com/lit/ds/symlink/ne555.pdf



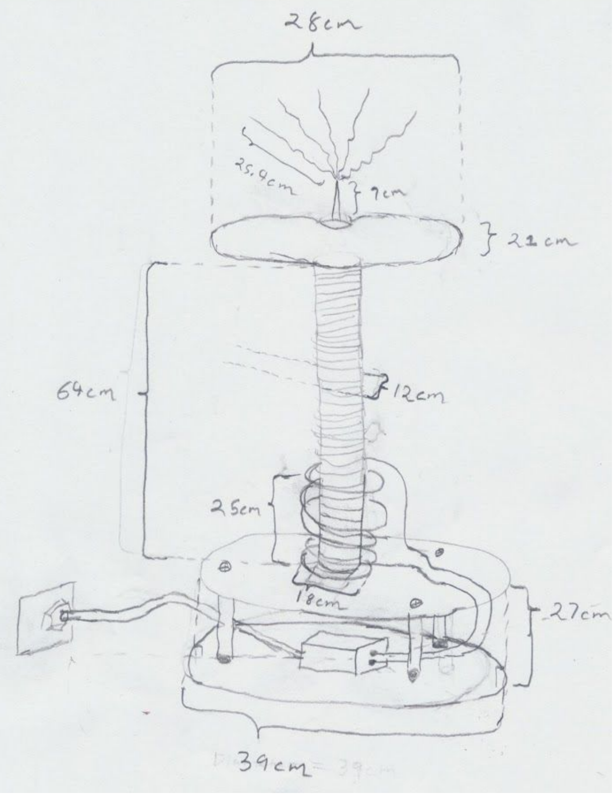
https://www.maximintegrated.com/en/app-notes/index.mvp/id/3977

H Bridge

With our pulse width modulated square-wave, we will feed this signal into the gate terminals of the MOSFETs that drive our H-Bridge. The purpose of the H-Bridge is to rapidly switch the Tesla-Coil's primary between a high voltage of about 300 V and ground, according to the output of our previously produced pulse width modulated square wave signal. By modulating whether or not the primary is connected to high or low, we can control the amount of power being input into the Tesla Coil at any given time. The overall goal, is to input power through pulse width modulated impulses that result in discharge being emitted out of the coil's top load at a frequency that creates a desired audible tone. The resulting sound's waveform will resemble the analog signal being input into the pulse width modulator. However, we expect some distortion as a result of using plasma as our sound producing mechanism. This distortion is not a concern, because the hardware limitations of the NES APU already restricts the sound resolution, so deteriorations will be less noticeable in comparison to a high-resolution signal.

Coil

The coil operates as a resonant transformer with an LRC circuit on the secondary side. The transformer itself is a 200:1 step-up air-core transformer. We are using an air-core to prevent magnetic flux saturation. In order to achieve the desired effect of musical electrical arcs it is very important that the coil be driven at its resonant frequency. We will determine this frequency by measuring the secondary coil's inductance and the toroidal top load's capacitance.


28cm
25.9cm
7cm
21cm
64cm
12cm
25cm
18cm
27cm
39cm