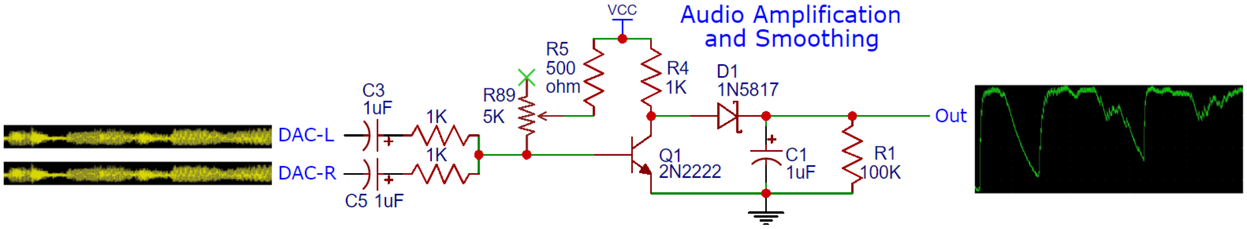
VUMeter Analysis

Animated Christmas Display

1. In 2019 a set of 16 Thunderbird car taillights were purchased at an estate sale. These lights were perfectly suited to make a large VUMeter display. The display was added and integrated into the Animated Christmas Display.
2. The Master Arduino commands the DFPlayer to play MP3 (Audio) files. The stereo output of the DFPlayer goes to the FM Transmitter, and to a VUMeter circuit that is then fed to an Analog input on the Arduino.
    a. The purpose of the VUMeter circuit is to keep the VUMeter display from being too “jumpy”. If the raw audio signal were sampled, one sample might catch the signal at a peak, then just a few microseconds later at zero. A pleasing refresh rate for the VUMeter display is about 100ms, so it is desired that a smoothed representation of an average of the previous 100ms be displayed, not some arbitrary instantaneous result.
3. Byte 3 bit 7 of the Light commands, if set, commands the Arduino to activate the VUMeter display
4. When activated, Analog input A0 senses the voltage on the pin to determine how many of the 16 lights on the physical VUMeter display should be lit
5. By experimentation it appears that a refresh rate of about 100ms is pleasing for the VUMeter display
6. The fastest update of the Christmas lights in the display is 20 frames/second, or 50ms per frame
    a. Transmission of each set of light commands takes 17ms and dissemination of the commands to five shift registers takes 1ms.
    b. Therefore, at the fastest refresh rate about 32ms is left over where the Master unit is idle.
    c. The VUMeter takes advantage of this idle time to:
        i. Sample the smoothed audio signal every 100ms
        ii. Modify the existing Light Commands to update only the VUMeter light bytes (2 bytes)
        iii. Retransmit the existing Light Commands with the modified VUMeter bytes to all units
    d. When the next 100ms interval comes up the Master unit might still be in its idle time – or – a new set of Light Commands may have come due. In either case, the appropriate VUMeter bits are set and the Light Commands are issued to the display.
7. The smoothing circuit is as follows:



8. The approach used by the circuit is as follows:
    a. The stereo Line Audio outputs of the DFPlayer (DAC-L and DAC-R) go through isolation capacitors then are combined into a monaural input. The signals are low voltage (1VRMS), too small to provide a sufficient signal for the Arduino’s Analog input whose value range is 0 to 5VDC. Therefore, an NPN transistor is added to strengthen the signal.
    b. The transistor is overbiased so that with no signal input, the transistor is driven to a fully saturated state; the output voltage is virtually zero.
    c. When the signal output of the DFPlayer goes positive the output remains fully saturated since more current is applied to the transistor’s base

d. When the signal output of the DFPlayer goes negative some of the current from the base bias resistor R1 is drawn off by the 1K input buffer resistors. The transistor is thus drawn out of saturation and the transistor output voltage rises.

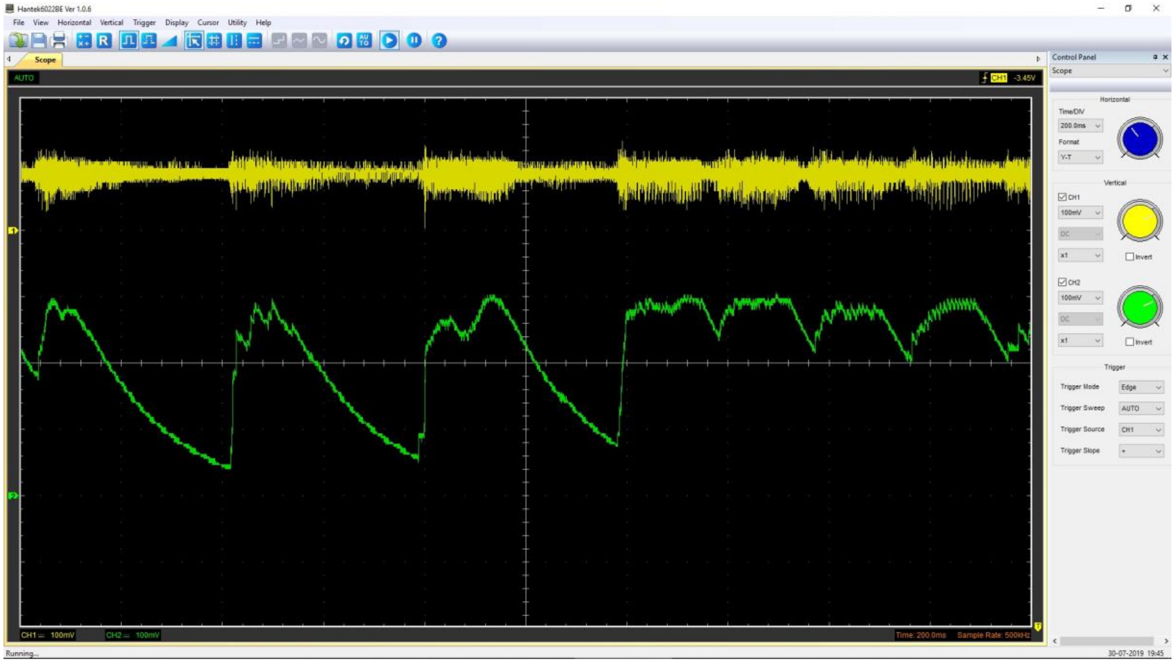
e. Depending on the magnitude of the negative voltage, the transistor can be switched off, which permits the output voltage of the transistor to rise to +5V.

f. Trimming potentiometer is adjusted so that the loudest audio signal just barely causes the 16th VUMeter LED to light.

g. The output of the transistor goes to a Schottky diode, which is used to offset the slight positive collector voltage when the transistor is saturated, and to isolate the smoothing circuitry from the transistor's output. The signal feeds a smoothing circuit.

h. Smoothing is accomplished by a simple RC circuit with values selected to give an appropriate time constant for smoothing of the audio input and decay of high values back to a lower value or to zero.

i. The following oscilloscope image shows an example small Line Audio input signal (yellow trace) and the smoothed output signal (green trace) that results from the circuit



Time/DIV 200.0ms

Format Y-T

☑ CH1 100mV DC x1 □ Invert

☑ CH2 100mV DC x1 □ Invert

Trigger Mode Edge

Trigger Sweep AUTO

Trigger Source CH1

Trigger Slope +

CH1 = 100mV

CH2 = 100mV

Time: 200.0ms

Sample Rate: 500kHz

j. The smoothed output voltage is connected to the Arduino Analog input pin A0 for processing by the software.

k. I experimented with 3dB per step filtering of the input signal, but the most pleasing effect ended up being a simple division of the input signal into 17 equal "buckets" (no LEDs on, to 1..16 LEDs on)