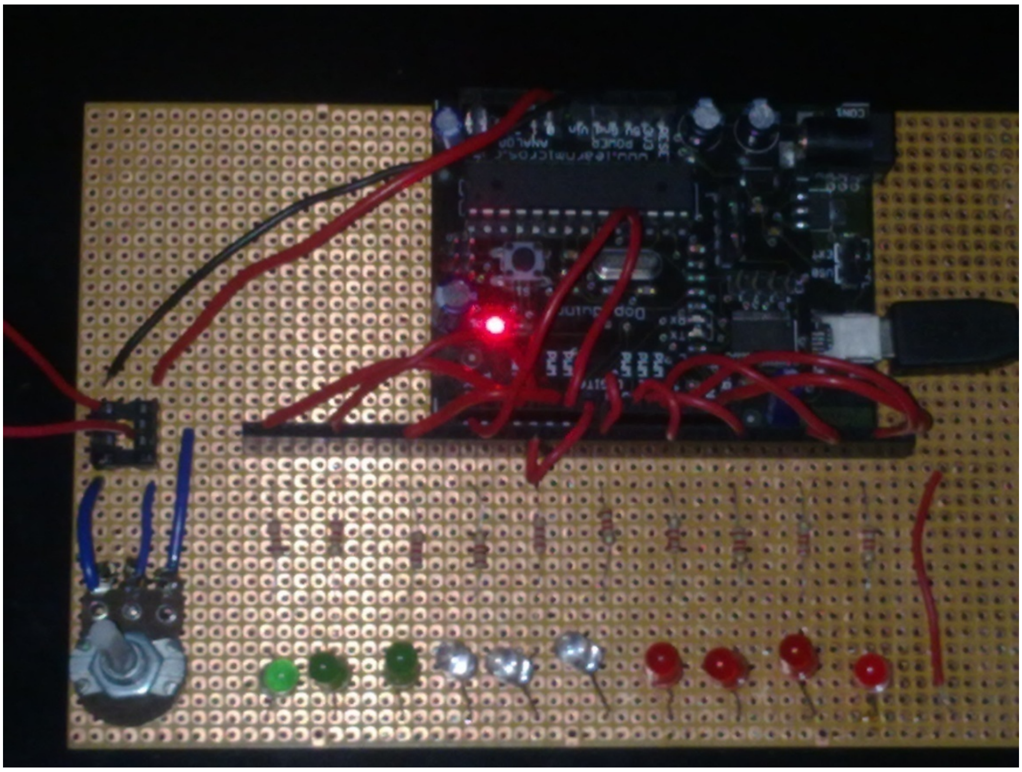
Volume Indicator
and
Control Unit



By:

Shanjit Singh Jajmann (110/EC/09)
Nilayan Das Gupta (79/EC/09)
Manish Gahalout (66/EC/09)
Richa Gupta (100/EC/09)

Winter Project 2010
3rd Semester

VU (Volume Unit) Meter

A VU (volume unit) meter is an audio metering device. It is designed to visually measure the "loudness" of an audio signal. A VU meter is often included in audio equipment to display a signal level in Volume Units; the device is sometimes also called volume indicator (VI).
The VU meter was developed in the late 1930s to help standardise transmissions over telephone lines. It went on to become a standard metering tool throughout the audio industry.
VU meters measure average sound levels and are designed to represent the way human ears perceive volume.

The VU meter has been implemented by an array of ten LED's controlled with the help of a microcontroller unit and a signal control incorporated into the circuit through use of a potentiometer.

There are literally dozens of different audio metering systems in common use around the world — and they often appear to read completely differently when supposedly displaying the same audio signal! However, there are perfectly good reasons why this should be the case and the differences are mainly due to the historical development of the various metering systems and their interpretation. Having said that, not all meters are equal and it's still a case of 'horses for courses' when choosing which system to use in particular applications.

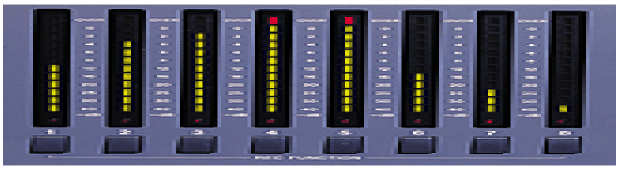
Types of Volume Unit Meters

There are two fundamental types of mechanical or moving coil meter that may be presented as a VU meter.
In a traditional sense only one type can be considered a real VU meter, and this is technically a low-level or low-range AC meter. In this type of meter, the incoming signal can be AC (Alternating Current), as is an audio signal, and within the meter this is converted to a voltage suitable for driving the coil – and thus the needle – in response to the amplitude of the original AC signal. This is achieved through rectification of the incoming AC, usually using a full wave bridge of germanium diodes.
Rectification with a full wave bridge effectively 'steers' the alternating waveform into a direct current.

Within the second type of meter this signal rectification circuit is not present, and thus, the meter will only respond to an audio signal if it has been converted to a suitable form before being presented to the meter – these meters are technically a DC (Direct Current) meter.

Thus there are two basic meter types: one that we can present audio too directly and another that requires some form of external circuitry to make it work with audio. Ideally, it's better to purchase the first type (a low-level AC meter) and forget the complexity of extra parts required to make the direct current (DC) meter read audio.

Audio Meters



All audio material has a certain dynamic range — the difference between the highest and lowest acceptable levels. We typically arrange for the loudest peaks to be below the maximum level which the system can handle and for the quietest signals to be kept well above the noise floor. If signals roam beyond these boundaries then your ears will usually tell you something is wrong, irrespective of whether you are using analogue or digital systems. However, metering can help to make the process of setting optimum signal levels much quicker and easier, warning you of potential problems before they occur.

Difference between VU and PPM meters

1. Because the VU (Volume Unit)meter measures 'average' levels, a sustained sound reads much higher than a brief percussive one, even when both sounds have the same maximum voltage level: the reading is dependent on both the amplitude and the duration of peaks in the signal. In addition, the standard VU response and fallback times (around 300 milliseconds each) exaggerate this effect, so transients and percussive sounds barely register at all and can cause unexpected overloads.

2. VU meters are inherently cheap, though, whether in the form of a moving-coil meter or as a bar-graph of LEDs. This is principally because there is no complex peak-sensing driver circuitry involved — as a consequence, VU meters tend to be used in order to cut costs where there is a requirement for a large number of meters, or where the meter needs only to provide an indication that sound is reaching a particular channel (such as on a multitrack recorder or large console).

3. 'Peak Programme Meters' or PPMs are considerably more expensive than VU meters, partly because of the much more elaborate circuitry and partly because of the precisely defined characteristics of the physical meter itself. Yet even PPM displays aren't designed to catch the very fastest of transient peaks, and are often termed 'quasi-peak' meters for this reason. They only show transients which are sustained for a defined.

Audio Metering Scales

While the VU meter has now become fairly standardised — zero point at +4dBu with a decibel scale ranging non-linearly from 20dB below this point to 3dB above — the PPM meter has a number of recognised scaling systems.

Where multiple meters are used together (such as in stereo or multi-channel systems), each meter's dynamic response must match to within a tenth of a second and their amplitude responses should be within 0.3dB of each other in the critical areas of the meter range. In order to calibrate VU and PPM meters, it's best to use a mid-frequency sine tone (typically 1kHz), as these signals are the most accurately read by meters which are not truly peak-reading.

Metering and Loudness

Although the VU meter was designed to provide some indication of volume, level meters in general display information about signal voltages rather than their perceived loudness. This is why it is important to realise that meters are only an aid to judging the acoustic balance of audio material. However, there are specialist metering systems designed to measure and display the absolute loudness of a programme, taking into account the characteristics of human perception. This kind of metering is becoming increasingly important as broadcasting organisations are now transmitting hundreds of channels via satellite, cable and the Internet and it is impossible to monitor all of them acoustically. Also, with the growing use of sophisticated multi-band compressors, it is possible to create audio material which appears completely normal on quasi-peak meters yet sounds extremely loud. This is starting to cause problems at programme junctions and, in the cinema, has lead to an increasing number of complaints about excessive playback volumes.

The perception of loudness depends not only on the level of a signal, but also on its frequency and bandwidth — the wider the bandwidth, the louder a signal seems to be, even if its peak level remains constant. The ear is known to be most sensitive around the 2-4kHz region, so signals in this frequency band will sound much louder than low- or high-frequency signals of similar peak level. For example, band-limited 1/3-octave noise signals at 100Hz and 10kHz can be almost 15dB higher in level than noise centred on 4kHz, yet all will be perceived as sounding equally loud.

Audio Vectorscope

The audio vectorscope is, in principle, an oscilloscope where the left and right sides of a stereo signal modulate the position of a dot along the display's X and Y axes respectively. The resulting two-dimensional pattern, called a Lissajous figure, is characteristic of the amplitude, frequency and phase relationships between the two signals. Most audio vectorscope displays work like this, though usually the X and Y axes are rotated by 45 degrees in order to provide a more easily understandable correlation between the displayed image and the stereo positioning of the audio signal — identical signals on both channels will produce a vertical line (representing a central, mono signal).

Components Used

1. Dopeduino, Arduino board that integrates Atmel ATmega168.
2. Ten 220Ω resistors.
3. Ten LED’s
4. A potentiometer (POT) of 10kΩ.
5. 3.5mm jack data transfer cable (for input to the circuit system).
6. USB cable for interfacing Arduino board with computer system.

Working of the VU meter circuit

Arduino Board:

The Arduino board in the circuit uses AVRATmega168 microcontroller IC. The microcontroller has been programmed with the help of matlab. The input pins to the microcontroller are programmed to receive the input from the computer in the form of audio signals, which are then processed by the ADC and the output then obtained through the ADC pins, fed into a parallel array of ten LED's through ten configured output pins. The audio input being sent to the microcontroller are then serially communicated back to the computer interface i.e. matlab in order to generate a real time X-Y plot of the response (analog voltage) against the collecteddata samples.

Resistors:

The resistors are connected to LED's through the output pins and save the circuit from burning out in case high current flows due to short-circuiting of the two pins accidentally.

LED's:

The LED's light up corresponding to the output voltage levels from the ADC pins of the microcontroller.

3.5mm Jack Cable:

It has been used for giving the audio signal input to the arduino board from the computer.

Potentiometer:

The potentiometer of 10kΩ has been incorporated in the circuit for introducing input signal level control in the circuit. The signal is fed first into the POT, and then into the arduino board for processing.

Applications of VU meter-

1. VU meters are generally designed to help monitor sound levels and help sound engineers.
2. Since it employs an ADC it is very easy to convert an Analog signal to its digital form and obtain its corresponding plot using various mathwork software.
3. Use of arduino board guarantees portability to different platforms.

Arduino Code:

```
//Arduino VU Meter

int led[10] = { 3, 4, 5, 6, 7, 8, 9, 10, 11, 12};            // Assign the pins for the leds

intleftChannel = 0;  // left channel input

int left, i;

void setup()
```

```
{

for (i = 0; i < 10; i++)                    // Tell the arduino that the leds are digital outputs

        pinMode(led[i], OUTPUT);
        Serial.begin(1200);

}
void loop()
{

left = analogRead(leftChannel);                          // read the left channel
Serial.println(left);                                     // uncomment to check the raw input.
left = left / 5;                                          // adjusts the sensitivity
//  Serial.println(left);                          // uncomment to check the modified input.
// left = 1500;                                           // uncomment to test all LEDs light.
// left = 0;             // uncomment to check the LEDs are not lit when the input is 0.

if (left == 0)                    // if the volume is 0 then turn off all LEDs
  {
for(i = 0; i < 10; i++)
   {
digitalWrite(led[i], LOW);
   }
  }
else
  {
for (i = 0; i < left; i++)                    // turn on the LEDs up to the volume level
   {
digitalWrite(led[i], HIGH);
   }
for(i = i; i < 10; i++)                       // turn off the LEDs above the voltage level
   {
digitalWrite(led[i], LOW);
   }
  }
}
```

Matlab Code:

```
clearall; clc; close all;


try

% Initialize serial port
    s = serial('COM30');
set(s,'BaudRate', 1200);
set(s,'DataBits', 8);
set(s,'StopBits', 1);
fopen(s);
s.ReadAsyncMode = 'continuous';
```

```
% Various variables
numberOfDatas = 100000;
data = zeros(1, numberOfDatas);
    i = 1;


% Main graph figure
figure(1);
holdon;
title('Incomming Data from External Device');
xlabel('Data Number');
ylabel('Analog Voltage (0-1023)');


% Start asynchronous reading
readasync(s);


while(i<=numberOfDatas)


% Get the data from the serial object
data(i) = fscanf(s, '%d');

% Plot the data
figure(1);
stem(i, data(i), 'm*');

% Ensure there are always 10 tick marks on the graph
if(i>10)
xlim([i-10 i]);
end

% Draw and flush
drawnow;

%Increment the counter
        i=i+1;


end


% Give the external device some time…
pause(10);

return;

catch

stopasync(s);
fclose(s);

fprintf(1, 'Sorry, too much data accumulated.');

return

end
```