SCC.131 Digital Systems
Systems Architecture

Week(19) L1: Signal digitisation - Sound

Ibrahim Aref

2024/2025

Slides adapted with permission from Paul Dempster 23/24 slides

Systems Architecture

- Welcome to Part 4 of the module!
- We'll be exploring significant architectural sub-systems of computers and related programming.
- Signal digitisation, OS – main concepts , Memory management, Process scheduling, Linux OS intro, Intro. to cloud virtualization , Intro. to graphics , GPU –shaders , Web Assembly , Networking concepts.

Signal Digitisation - Sound

Topic overview

- Sound is one of human's basic senses, yet it is also one which is often looked at in a limited manner in computing, mainly in
  - Video
  - Games
  - The impact of sound-related disabilities.
- This topic's learning objectives:
  1. What sound is.
  2. How sound is represented on a computer.
  3. How sound is sampled.
  4. How sound is converted and stored in digital form.
  5. How sampling intervals and other factors affect the size of a sound file & quality of playback.
  6. The hardware and software involved in computer audio.

What is sound?

- Sound is a vibration in a medium.
- In more human terms, it’s a pressure wave in air.
- Our ears detect this pressure wave through a complex mechanical and electrical process
  - https://en.wikipedia.org/wiki/File:Journey_of_Sound_to_the_Brain.ogv [For interest]
- We can mechanically replicate sound with loudspeakers.
  - These receive electronic signals which (typically) cause surfaces(cones) inside them to move backwards and forwards, creating pressure waves that propagate through the air outside the speaker.

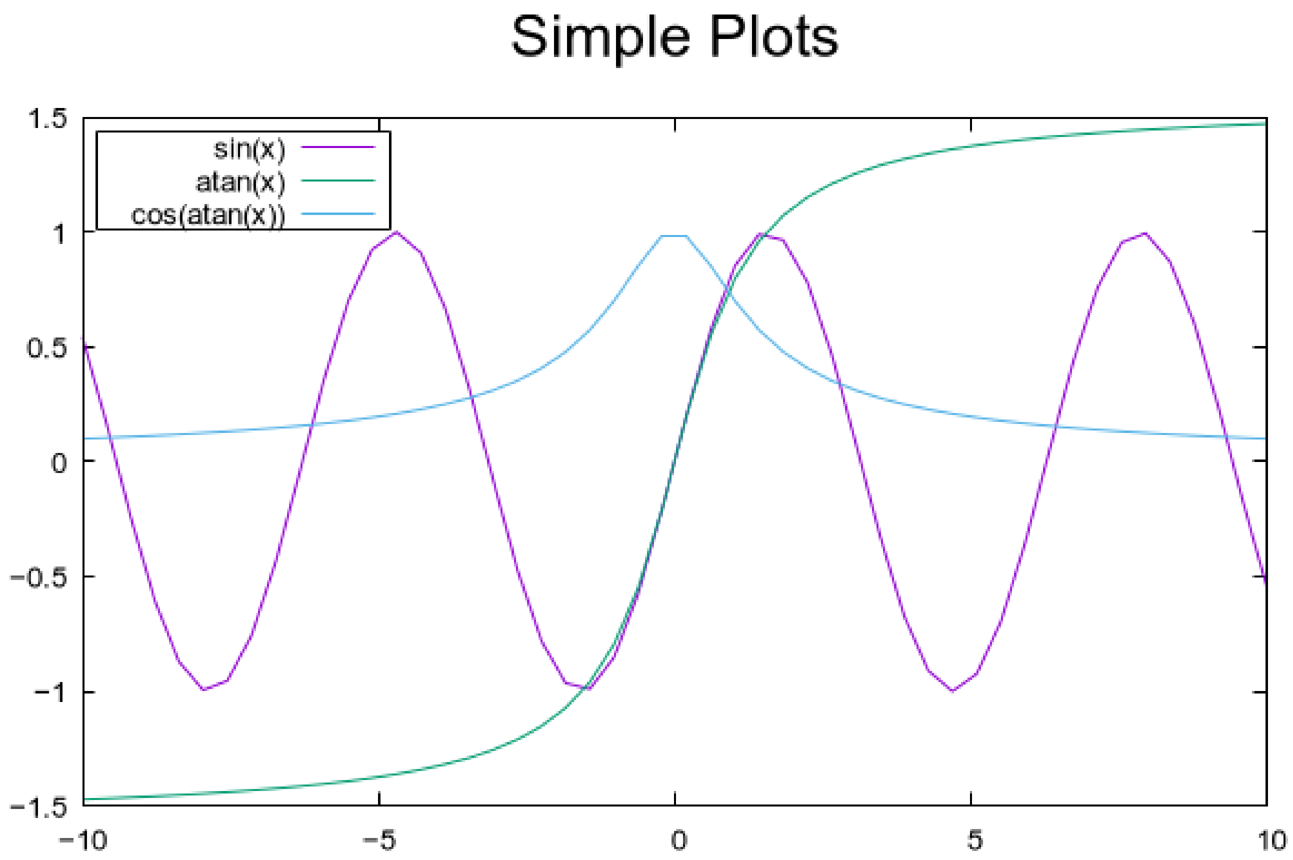
Analogue Encoding of a Sound Wave

Lancaster University

- **"Pure tones"** are sine waves with frequencies related to their perceived pitch.
- **Characteristics:** Single Frequency, Sinusoidal Shape, No Harmonics, Used in Testing & Research.
- https://en.wikipedia.org/wiki/Piano_key_frequencies has a decent table showing these for a piano and giving the formula.
- There are different *tunings* for musical instruments but we'll talk about the "standard" A440 tuning.
  - A_4 = 440 Hz (set origin point of the scale)
  - C_4 ~= 261 Hz (middle C)
  - C_8 ~=4168 Hz (highest C on a normal piano)
  - C_1 ~= 32 Hz (lowest C on a normal piano)
- Humans' "normal" hearing range is approximately 20 Hz to 20,000 Hz (dropping to 15-17 kHz in upper limit in adults).

Visualise Audio Wave (GNUplot)

- Command-line and GUI program.
- So if we want to visualise sound wave, a quick way is via the unix command line plotting tool **gnuplot**.
- Remember an A_4 at 440 Hz means we have 440 whole sin waves per second (waves per second is what frequency means).



http://www.gnuplot.info/demo/simple.html

Sound Representation

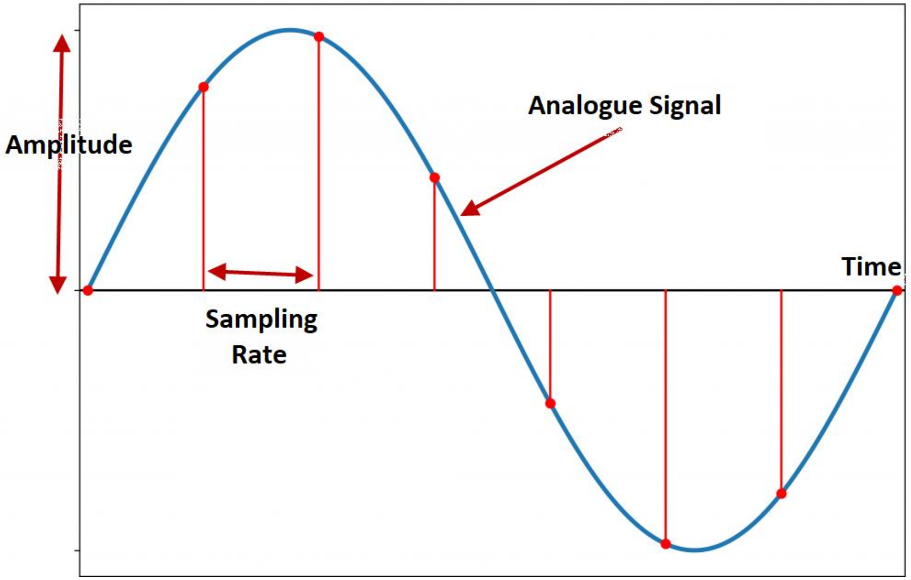
Analogue sound is represented as a **wave**.

To represent the varying values of a soundwave, it’s height must be measured at regular intervals and the measurements given binary codes.

This process is called **Sampling** and the number of samples taken in a second is called the **sampling rate**

The sampled measurements make up the digital sound file.

- **Sound can be stored in a computer as binary codes**



Sampling Rate

Sample Rate

- If we want to represent this electronically as a set of discrete values then we need to decide how many samples we want to have.
  - **Less samples = less data** but a **less accurate** representation of the original sound
  - As the number of samples **tends to infinity**, we have a perfect representation of the audio.
  - A standard nowadays is **44.1kHz** (based on CDs)
  - 8 kHz was used for telephones in the past
  - 48 kHz is another common professional sampling rate.
  - 96 kHz is supported in DVD-Audio, HD DVD, Blue-Ray.

Impact of Sample Rate on Quality and File Size

- Higher sample rate = better quality
- Higher sample rate = larger file size
- A higher sample rate tends to deliver a better-quality audio reproduction.
- Balancing quality and size: Depending on the intended use, choose a sample rate that strikes a balance between high quality and manageable file size.

- Example: CD quality: A standard CD uses a sample rate of **44.1kHz**, which is considered a good balance between quality and file size for most listening situations.

Bit Depth

- Sample rate tells us how many values we will use to interpolate the audio wave per second.
  - It'll get us the shape.
  - But what kind of value will we store? Integer? Float? How big/how many bits.
- Audio is typically sampled at 8, 16, and 24 bits.
  - The bigger the bit depth, the more possible values an individual sample can have, therefore the more accurate the represented value.
  - What value range to use? 0..255? -128..127? 0.0..1.0?
- Storage required per second = sample rate * bit depth
  - (for 1 channel!)
- ***Bit depth is the number of bits that are used to represent each sample of audio.***

File Size

File Size (bits) = Sample rate (Hz - not kHz) x Bit Depth x No. of channels x Duration(seconds)

Where:

- **Sampling Rate** = Number of samples per second (Hz or samples/second).
- **Bit Depth** = Number of bits used to represent each sample.
- **Number of Channels** = Number of audio channels (e.g., 1 for mono, 2 for stereo).
- **Duration** = Recording length in seconds.
- **The result is in bits *** Total bits/8 = bytes *** Bytes/1,000,000 = megabytes or MBs**

Example: Given: Sampling Rate = 44,100 Hz (CD quality). Bit Depth = 16 bits. Number of Channels = 2 (stereo). Duration = 60 seconds (1 minute recording). Calculate File Size in Bits.

File Size (bits)= 44,100 × 16 × 2 × 60 = 84,672,000 bits

Sound hardware

PC Speakers

- A hardware device that allows you to hear sound from your computer.
- First came small speakers driven by the CPU.
- Still existed in most PC's until the last decade.
- Often can only beep at different frequencies.
- Not meant to reproduce complex sound.

PC Speaker, Wikimedia, Hans Haase, CC BY-SA 3.0

Sound Cards

- As demand for more complex audio arose, dedicated sound cards were developed with
  - Amplifiers
  - Input as well as output
  - Multiple channels of sound (stero, surround) and polyphony (multiple sounds on a channel)
  - Dedicated chips to process sound, mix channels together
  - MIDI support, wavetables, positional audio, …
  - Support for CD drives
  - Joysticks ports(!)

Sound cards (II)

- Significant models
  - AdLib (1987, Ad Lib, Inc)
  - SoundBlaster (1989, Creative Technology) and clones
  - Gravis UltraSound (1992, Advanced Gravis Computer Technology Ltd)
  - AC’97 PC Audio standard (1997, Intel)
  - A3D supporting cards (~1997, Aureal)
- Non-PC audio was hugely popular
  - Commodore 64 SID chip (1982)
  - Amiga “Paula” audio chip (1985, revisions through 1992)
  - Atari ST with built-in MIDI support (1985)

Fall of Consumer Sound Cards

- Sound cards remained popular through the 1990's and 2000's, gaining more and more features, eg,
  - Positional and spatial audio
  - Physically simulated soundscapes

- Windows Vista release in 2007 and changed the audio subsystem of Windows in a substantial way
  - Removed DirectAudio API and the sound HAL (hardware abstraction layer)
  - This killed sound cards ability to accelerate sound and be interfaced by games in an easy way. The consumer soundcard market died rapidly after.

Sound software

Generating Sound in a Program

- If we want to generate sound on a computer, then we need to generate a sound sample that appears similar to the wave we want.
- Then we need to send that to the sound card.

- Two tasks
  1. Generating our sample(s)
  2. How to keep the audio subsystem of the OS/Hardware fed with sound data.

1- Generating Samples

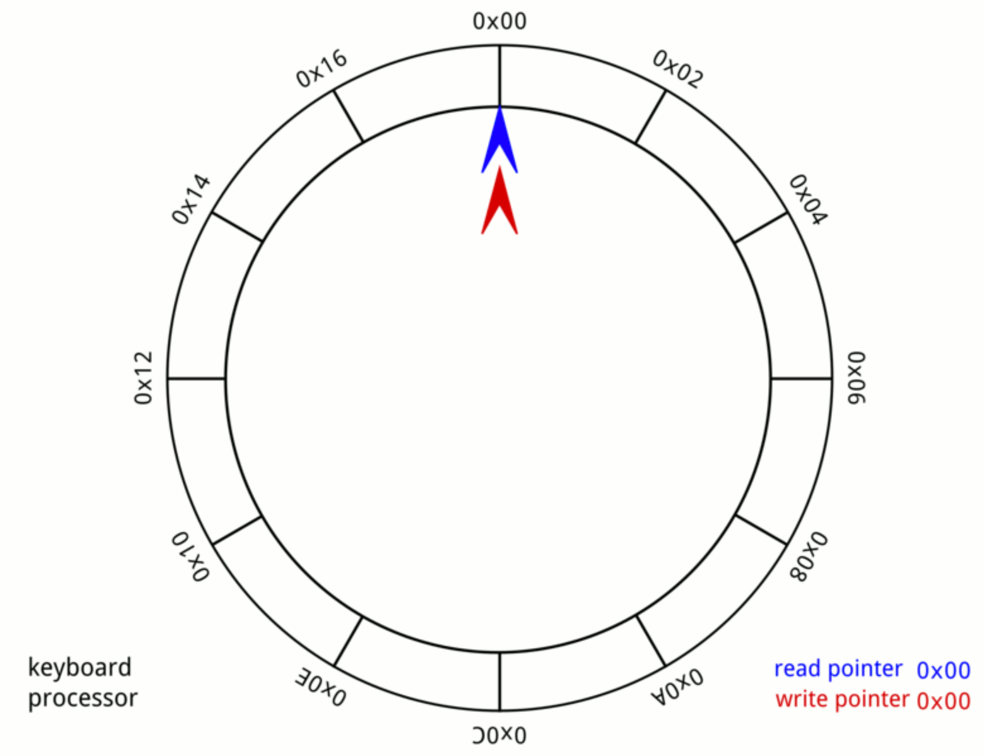
- Ignoring just taking the easy route and loading a sample from a file…
- Sounds are periodic things, especially pure sound tones which can be described by 1 or more sin waves of different frequencies.
- Loops are a perfect way to iterate through the values in a sample.
- Store them in an array (if you know the length) or a list (if you don't).
    - But be careful of your algorithmic complexity of appends to your list!
- Examine your output data by playing the same (hard but rewarding) or saving the sample values to a file and using ***gnuplot*** to show the waveform (easy and good for debugging but only for short samples)

2- Feeding the Audio Beast

- When you start **playing sound**, you can need a lot of data and it might be an open-ended commitment!
  - **For a short audio sample** play like a jump noise, you can generate the entire sample and send it to the sound system in one go, and let it play through it.
  - **For longer samples**, music, or dynamically generated audio, you need to start playing it before you generate it all (if there even is an end!)
- Streaming audio data into the sound system.
- Fill a buffer, let the audio subsystem “drain” it, refill before it’s empty.

Ring Buffer

- A useful common data-structure at operating system/hardware level is a ring buffer. This is an ordered, sequential data structure that has no end but just loops back to the start.
- Used for many different purposes, not just audio.
- How might you implement one with what programming and data-structures you already know?



Summary

- Quick overview of an entire architectural sub-system to kick off the last block of the module.
- Introduces what sound is and how it may be represented in a computer.
- Brief history of PC sound hardware.
- Learning a bit of **gnuplot**
- Generating samples in software (exercising SCC.111 skills)
- Ring buffer data-structure (exercising SCC.121 skills)

Jumping off points

- **[Web Audio API]**
  - Implemented in browsers so you can easily write some audio code without worrying about which computer it will run on.
  - Choose audio sources, add effects to audio, create audio visualizations, and much more.
  - It's in JavaScript (since it's in a browser), but if you fancy a try, there are simple and more complex introductions to it easily available.
- **Your micro:bit v2** has a speaker on the back and there is support in MakeCode drag&drop programming and C to play things over it.

Thank you for attending, any questions?