Real time digital audio processing with Arduino

André J. Bianchi
ajb@ime.usp.br

Marcelo Queiroz
mqz@ime.usp.br

Departament of Computer Science
Institute of Mathematics and Statistics
University of São Paulo

SMC 2013, July 30th - August 3rd

Real time digital signal processing

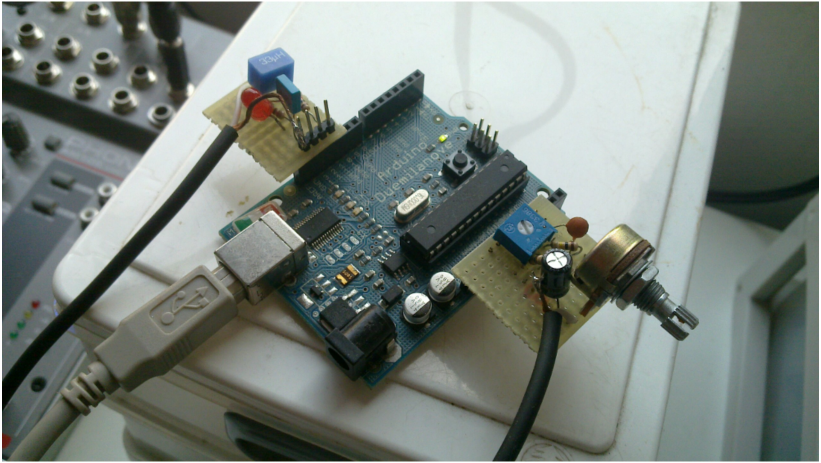
Digital audio signal processing includes:

- Acquiring samples.
- Processing.
- Outputting results.

Real time restriction:

- Block processing: N samples.
- Sampling frequency: R Hz.
- DSP cycle period: $T_{DSP} = \frac{N}{R}$ s.

Real time DSP with Arduino



http://interface.khm.de/index.php/lab/experiments/arduino-realtime-audio-processing/

Atmel AVR microcontroller (ATmega328P)

Microcontroller's characteristics:

- CPU: ALU and registers (16 MHz - 8 bits).
- Memory: Flash (32 KB), SRAM (2 KB) e EEPROM (1 KB).
- Digital I/O ports:
    - Audio input: analog to digital converter.
    - Audio output: counters capable of doing PWM.

Arduino performance for real time digital audio processing

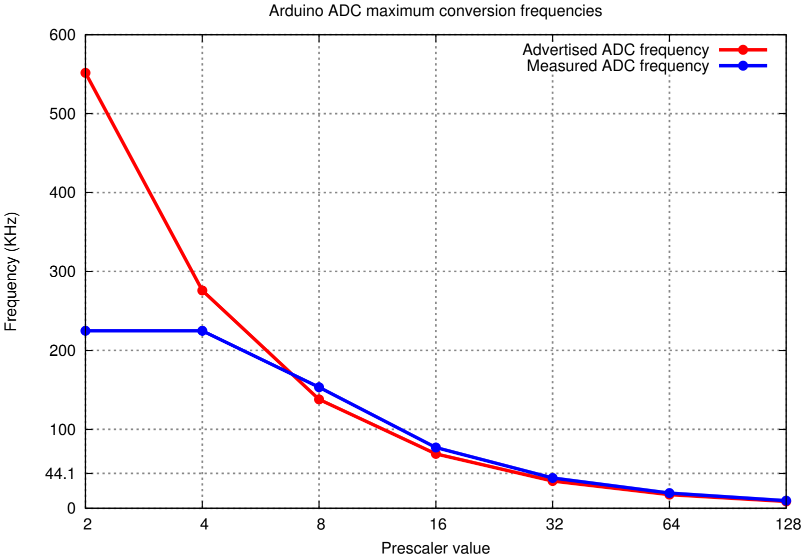
Questions:

- What is the maximum number of operations feasible in real-time?
- Which implementation details make a difference?
- What is the quality of the resulting audio signal?

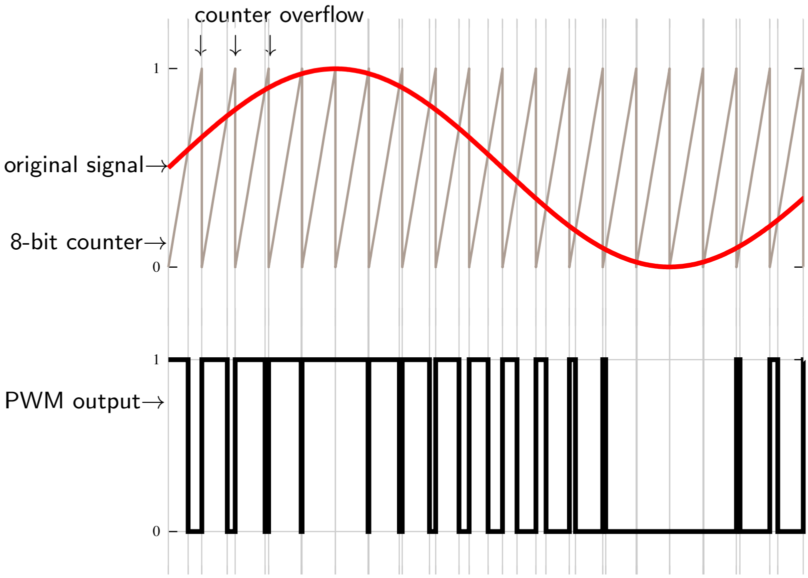
DSP algorithms implemented:

- Additive synthesis.
- Time-domain convolution.
- FFT.

Audio input: analog to digital converter



Pulse Width Modulation



Audio output: Pulse Width Modulation

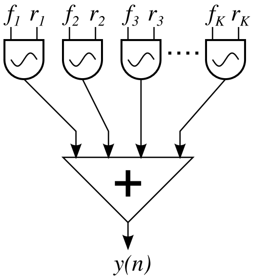
8-bit counter frequencies for different prescaler values:

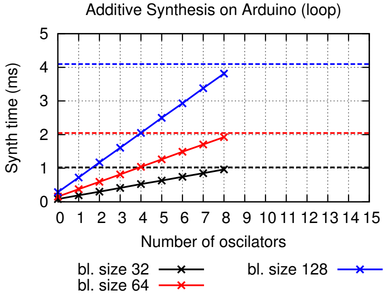
prescaler	f_{incr} (KHz)	f_{overflow} (Hz)
1	16000	62500
8	2000	7812
32	500	1953
64	250	976
128	125	488
256	62.5	244
1024	15.625	61

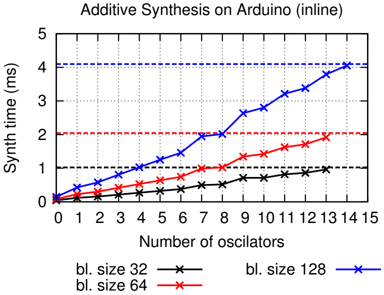
PWM overflow interrupt allow for periodically triggering:

- ADC conversion.
- Signal manipulation.
- PWM mechanism value set.

Additive synthesis




Additive Synthesis on Arduino (loop)


Additive Synthesis on Arduino (inline)

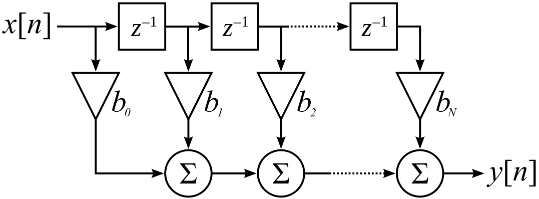
Additive synthesis

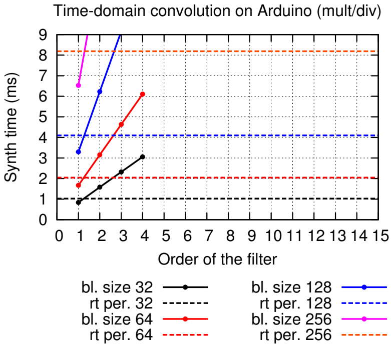
Example

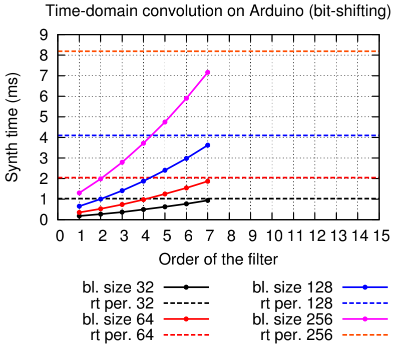
Sum of harmonics with $f_0=200$ Hz:

$$y[n] = \sum_{k=1}^{?} \sin\left(2\pi k 200 \frac{n}{R}\right).$$

Time-domain convolution

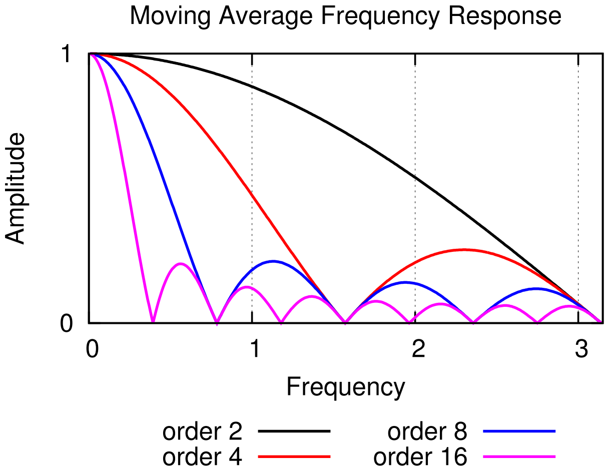



Time-domain convolution on Arduino (mult/div)

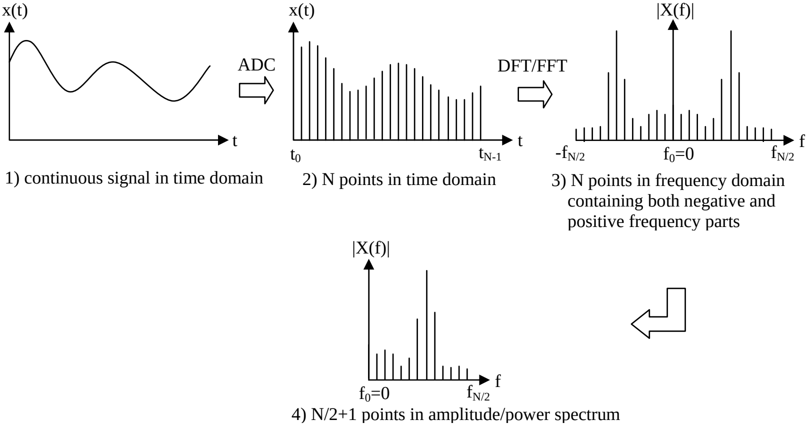

Time-domain convolution on Arduino (bit-shifting)

Time-domain convolution

Example: moving average



Fast Fourier Transform



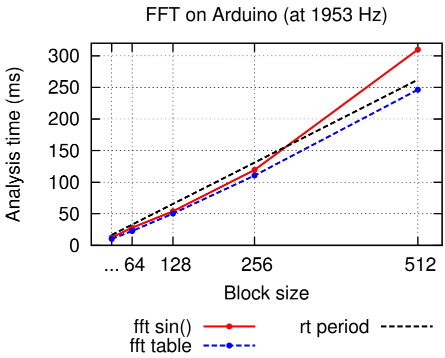
1) continuous signal in time domain

2) N points in time domain

3) N points in frequency domain containing both negative and positive frequency parts

4) N/2+1 points in amplitude/power spectrum

Fast Fourier Transform



Maximum frequency for block size 256:

- Mean calculation time ≈ 428,15 μs per sample.
- Maximum frequency ≈ 2.335 Hz.
- PWM prescaler value 32 ⇒ $R = 1.953$ Hz.

Conclusions

- Many implementation details make a difference:
    - Types used (byte, unsigned long, int, float, etc).
    - Type of operations: integer (multiplication, division, sum) and bitwise.
    - Presence of loops.
    - Use of variables and vectors.

- Families of algorithms can be found to make it feasible to use the Arduino in real time audio processing.

Thank you for your attention!

Contact:

- Email: {ajb,mqz}@ime.usp.br
- This presentation: http://www.ime.usp.br/~ajb/
- CM at IME: http://compmus.ime.usp.br/

Attribution of figures taken from wikipedia:

- PWM: Zurecs (zureks@gmail.com).
- Additive synthesis: Chrisjonson.
- FFT: Virens.