Designing a feature-rich polyphonic MIDI synthesizer

Sandro Razmadze

Student of Electrical and Computer Engineering at

Agricultural University of Georgia

Synthesizer:

noun

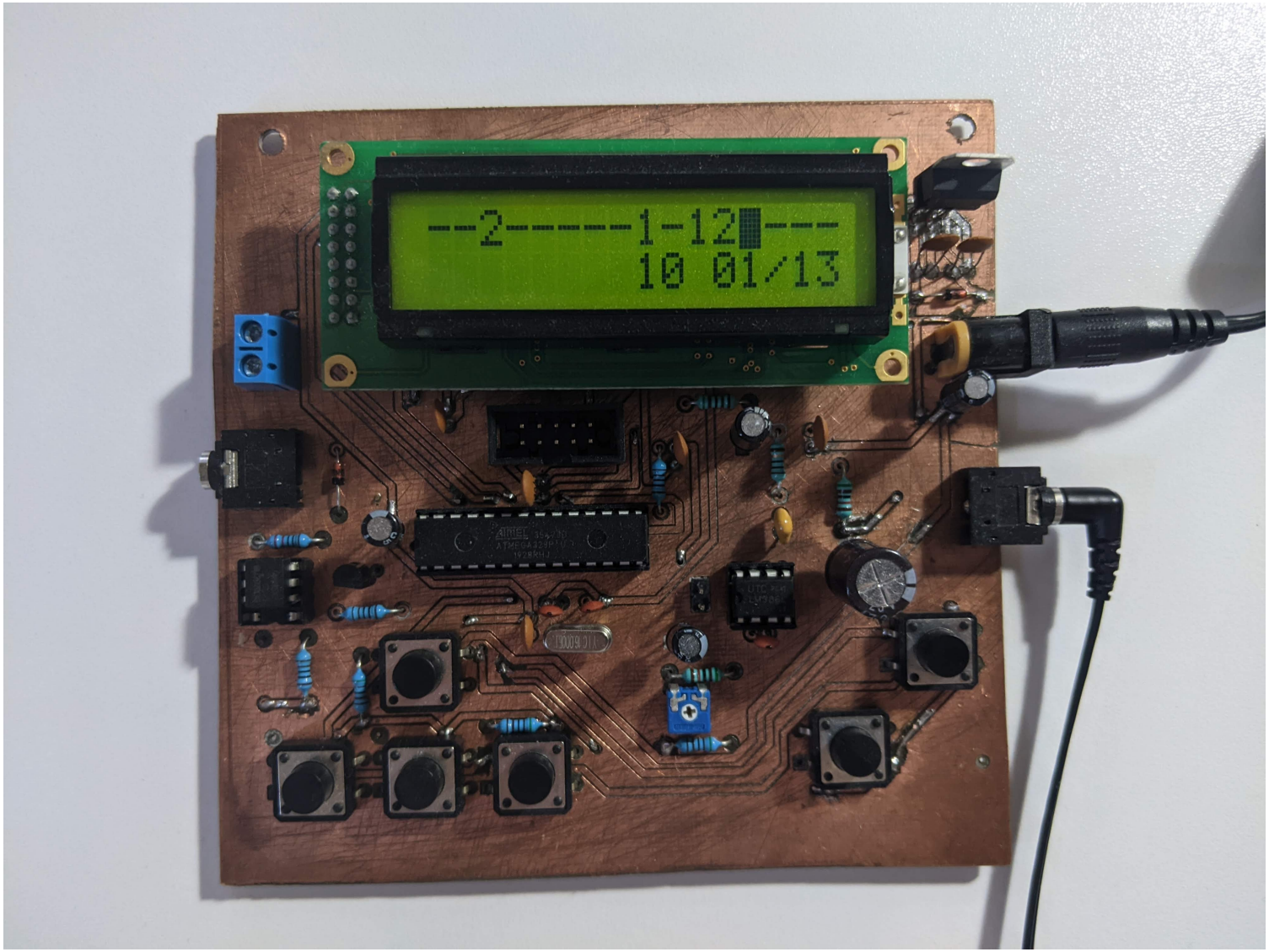
- "An electronic machine for producing different sounds ..."

  - Oxford Advanced Learner's Dictionary

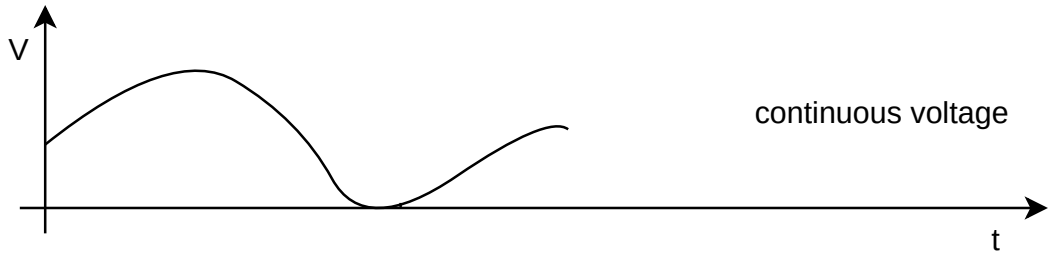

CONTROLLERS
OSCILLATOR BANK
MIXER
MODIFIERS
OUTPUT


Roland
Drumatix
TEMPO
INSTRUMENT
MODE
VOLUME
TR-606
Computer Controlled

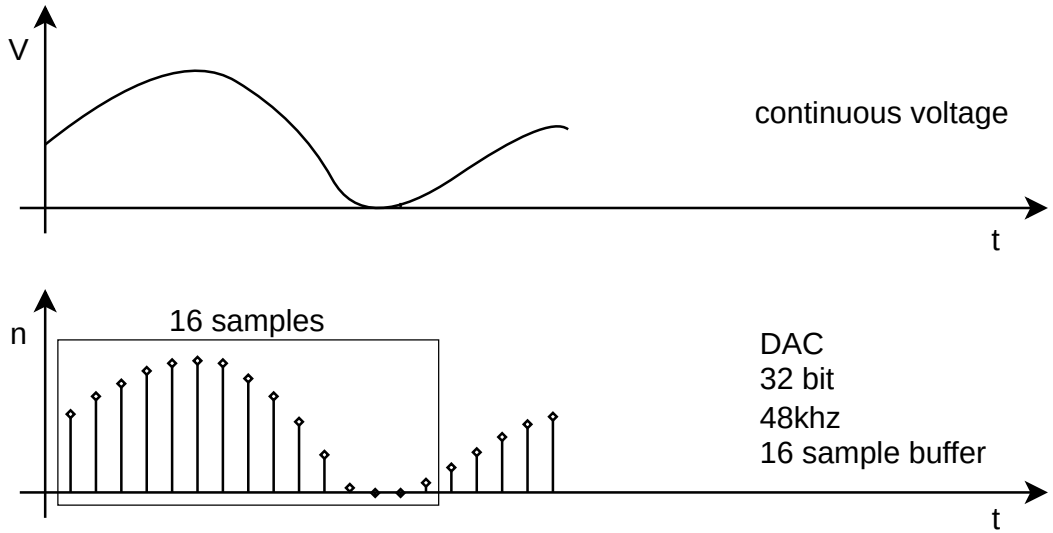

Roland
Roland Studio System
SYSTEM-100M 191-J


--2-----1-12---
10 01/13

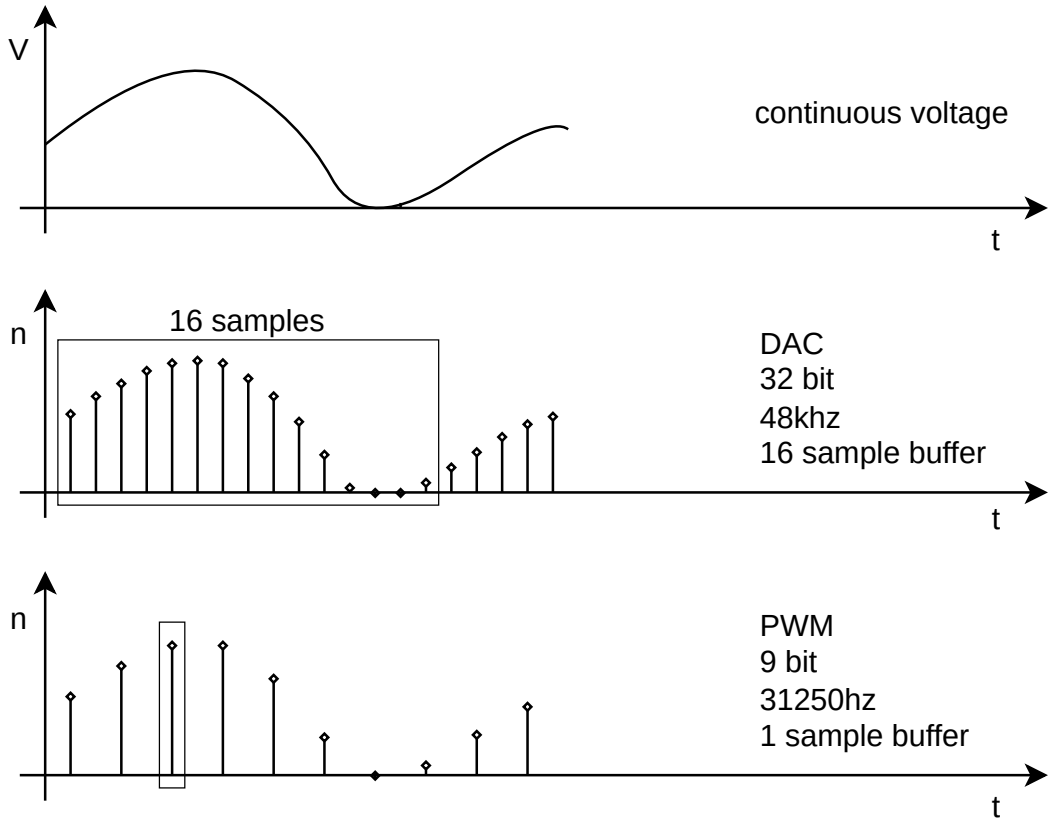
digital audio



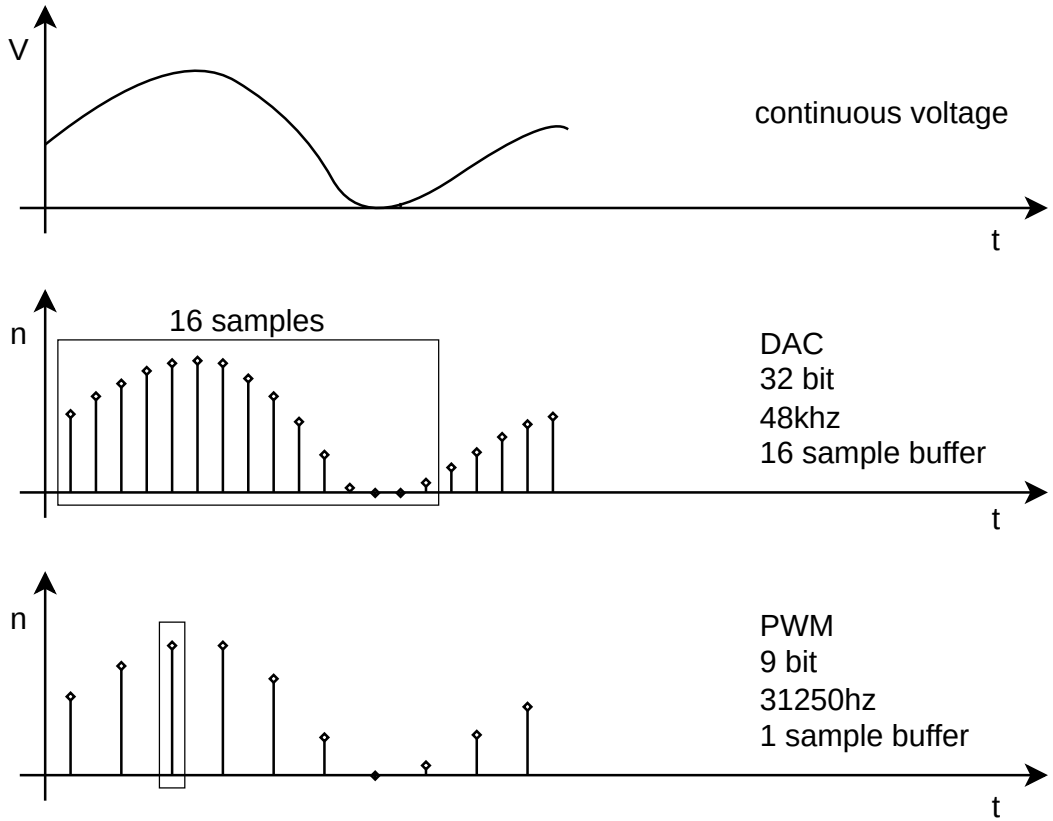
digital audio



digital audio



digital audio



$$\frac{16000000}{31250} = 512 \text{ (cycles)}$$

generating waveforms

sine wave (naive implementation):

```
float time=0;
float delta=0.1;

float generate_output(){
    time=time+delta;
    return sin(time)*512;
}
```

generating waveforms

sine wave (naive implementation):

```
float time=0;
float delta=0.1;

float generate_output(){
    time=time+delta;
    return sin(time)*512;
}
```

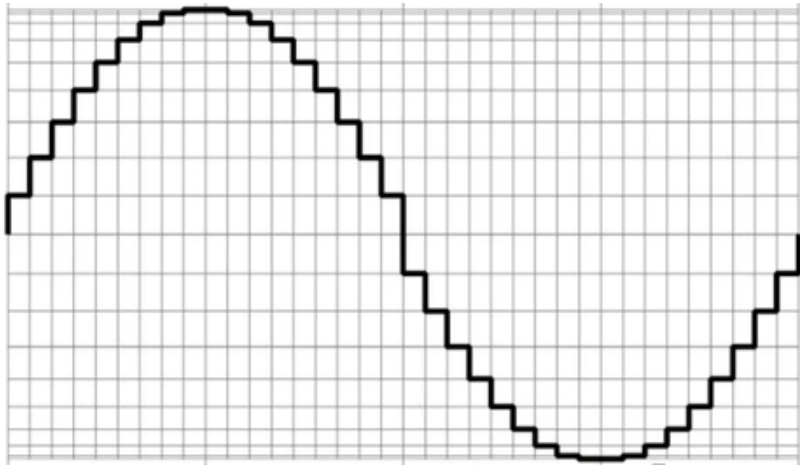
- -Os build
- just for one `sin` call
- huge amounts of asm = *SLOW*

generated asm:

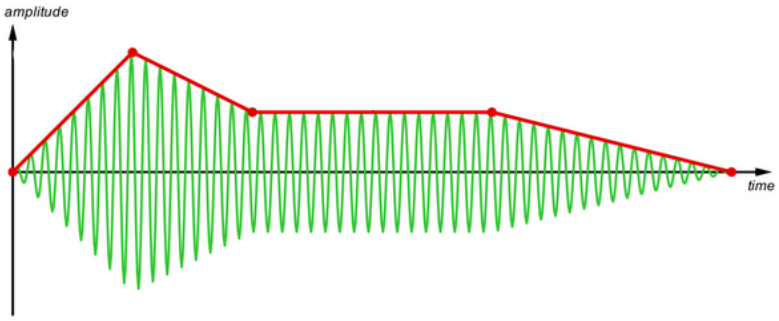
```
424:   6f 5f           subi    r22, 0xFF   ; 25
426:   7f 4f           sbci    r23, 0xFF   ; 25
428:   8f 4f           sbci    r24, 0xFF   ; 25
42a:   9f 4f           sbci    r25, 0xFF   ; 25
42c:   08 95           ret

0000042e <__fp_sinus>:
42e:   ef 93           push    r30
430:   e0 ff           sbrs    r30, 0
432:   07 c0           rjmp    .+14        ; 0x
434:   a2 ea           ldi r26, 0xA2   ; 162
436:   2a ed           ldi r18, 0xDA   ; 218
438:   3f e0           ldi r19, 0x0F   ; 15
43a:   49 ec           ldi r20, 0xC9   ; 201
43c:   5f eb           ldi r21, 0xBF   ; 191
43e:   0e 94 3c 02     call    0x478   ; 0x478
442:   0e 94 06 02     call    0x40c   ; 0x40c
446:   0f 90           pop r0
448:   03 94           inc r0
44a:   01 fc           sbrc    r0, 1
44c:   90 58           subi    r25, 0x80   ; 12
44e:   e8 e6           ldi r30, 0x68   ; 104
450:   f0 e0           ldi r31, 0x00   ; 0
452:   0c 94 9f 02     jmp 0x53e   ; 0x53e <__f
456:   0e 94 cd 01     call    0x39a   ; 0x39a
45a:   38 f0           brcs    .+14        ; 0x
45c:   0e 94 d4 01     call    0x3a8   ; 0x3a8
460:   20 f0           brcs    .+8         ; 0x
462:   39 f4           brne    .+14        ; 0x
464:   9f 3f           cpi r25, 0xFF   ; 255
466:   19 f4           brne    .+6         ; 0x
468:   26 f4           brtc    .+8         ; 0x
46a:   0c 94 ca 01     jmp 0x394   ; 0x394 <__f
46e:   0e f4           brtc    .+2         ; 0x
470:   e0 95           com r30
472:   e7 fb           bst r30, 7
474:   0c 94 c4 01     jmp 0x388   ; 0x388 <__f

00000478 <__addsf3x>:
478:   e9 2f           mov r30, r25
47a:   0e 94 26 01     call    0x24c   ; 0x24c
47e:   58 f3           brcs    .-42        ; 0x
480:   ba 17           cp  r27, r26
482:   62 07           cpc r22, r18
484:   73 07           cpc r23, r19
486:   84 07           cpc r24, r20
```

generating waveforms - using lookup tables

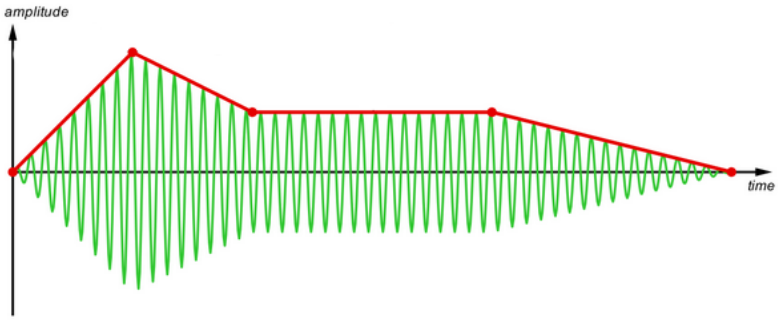
```
sine_wave[253]=
{
127, 130, 133, 137, 140, 143, 146, 149, 152, 155, 158, 162, 165, 168,
171, 174, 177, 179, 182, 185, 188, 191, 193, 196, 199, 201, 204, 206,
209, 211, 214, 216, 218, 220, 223, 225, 227, 229, 231, 232, 234, 236,
237, 239, 240, 242, 243, 244, 246, 247, 248, 249, 250, 251, 251, 252,
253, 253, 254, 254, 254, 254, 254, 255, 254, 254, 254, 254, 254, 253,
253, 252, 251, 251, 250, 249, 248, 247, 246, 244, 243, 242, 240, 239,
...}
```



scaling by using fixed point arithmetic:

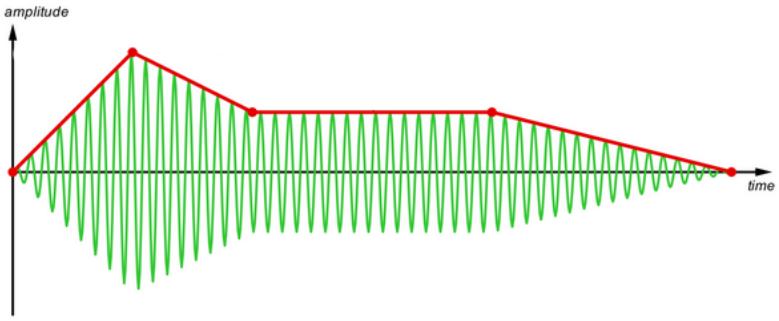


scaling by using fixed point arithmetic:



$$232 \cdot 0.32 = 74.24$$

scaling by using fixed point arithmetic:



$$232 \cdot 0.32 = 74.24$$

instead of using real numbers, we can pre-scale both operands by same factor and operate using integers:

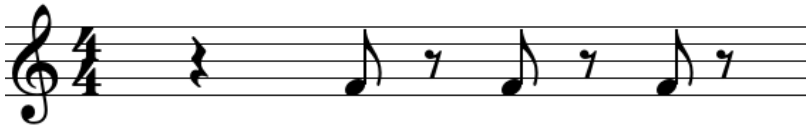
$$(232 \cdot 2^8) \cdot 81$$

and later we just divide it by our scaling factor squared:

$$\frac{(232 \cdot 2^8) \cdot 81}{2^{16}} = 73$$

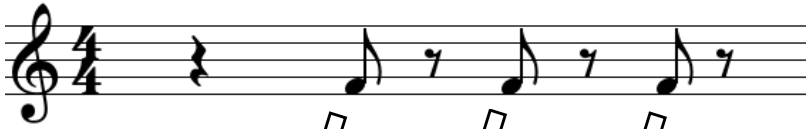
sequencer - definition

musical notation:

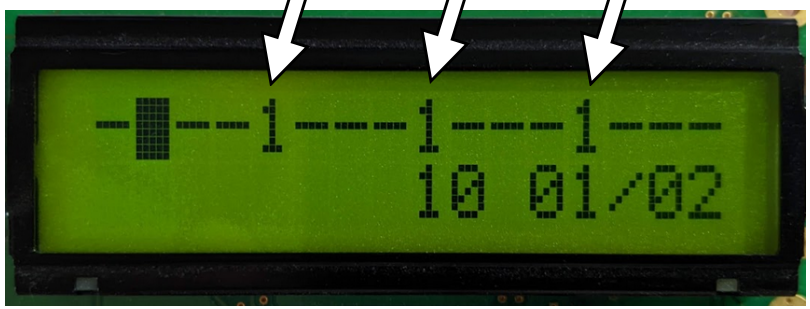


sequencer - definition

musical notation:

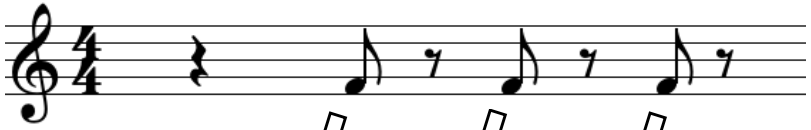


sequencer representation:

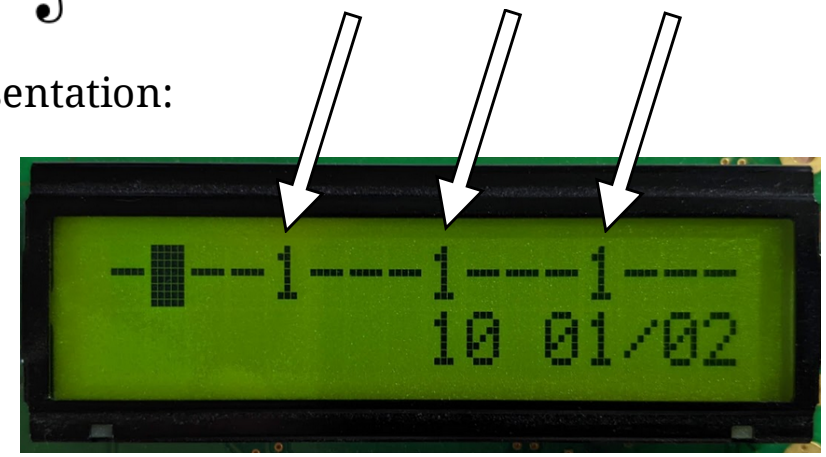


sequencer - definition

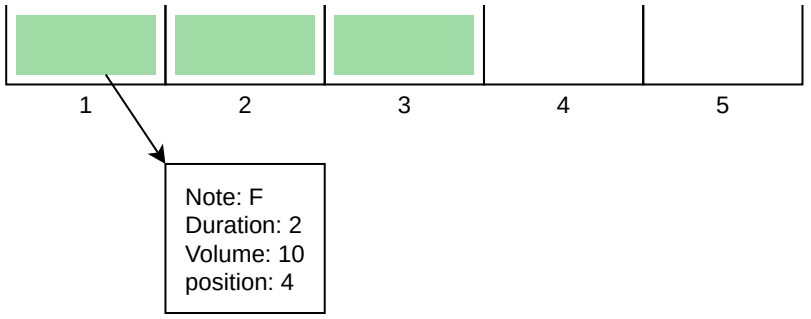
musical notation:



sequencer representation:

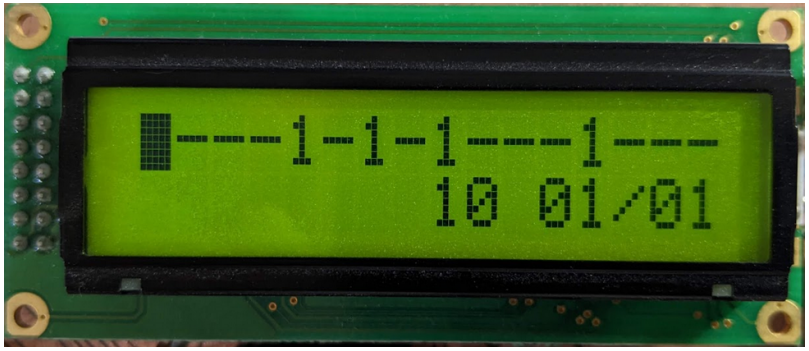


layout in memory:



sequencer - memory layout problem

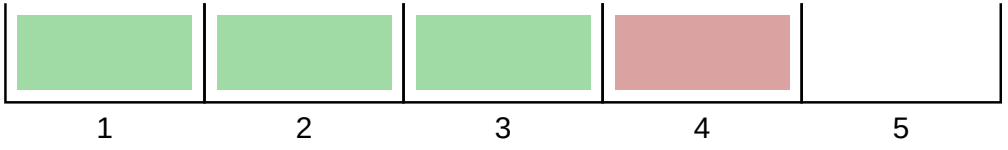
inserting a new note after the first one:



sequencer - memory layout problem

inserting a new note after the first one:

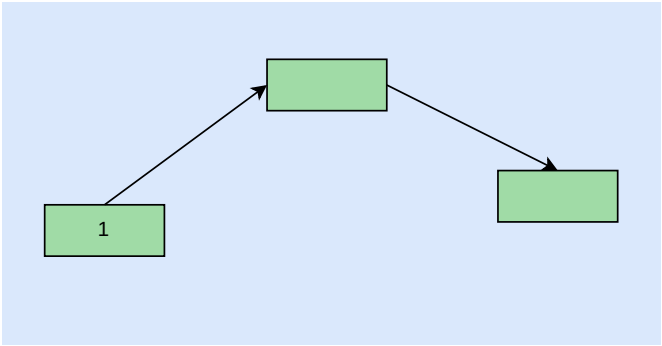
now elements in our array are out of order:



we can try to reorder old elements but depending on the number of notes this can take a lot of processor operations (cycles).

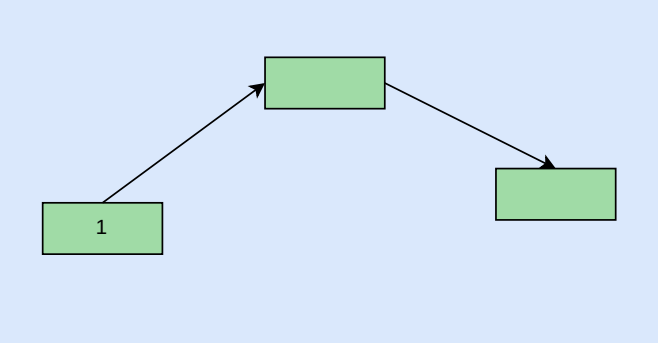
sequencer - optimal data-structure

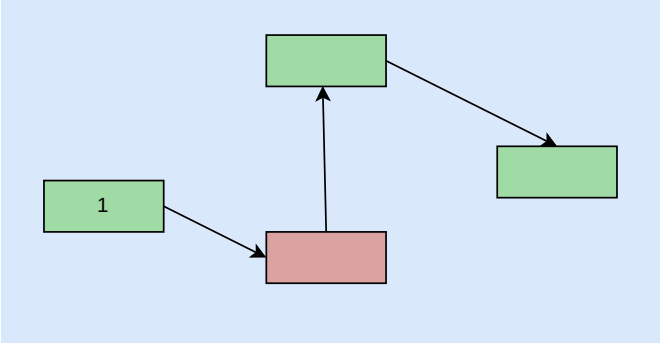
linked list:



sequencer - optimal data-structure

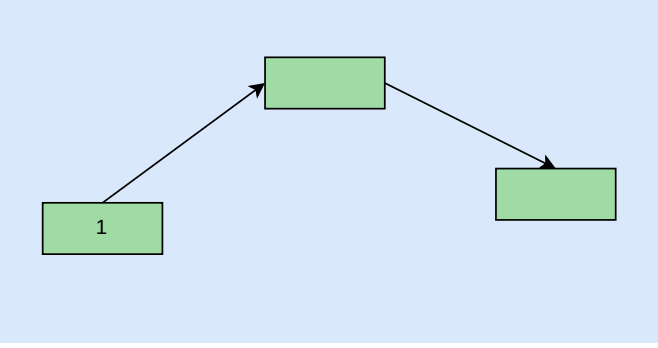
linked list:

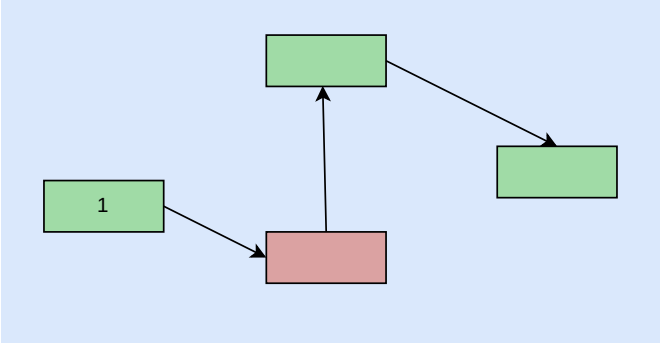




sequencer - optimal data-structure

linked list:

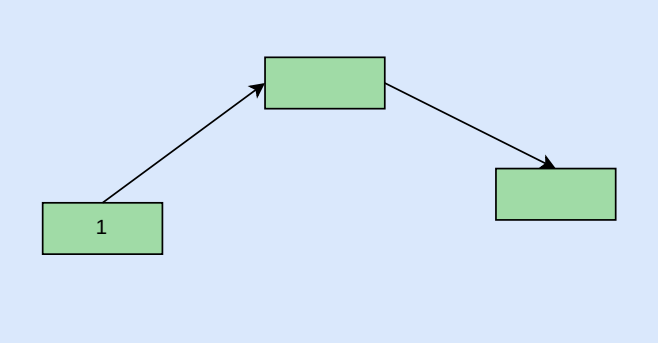


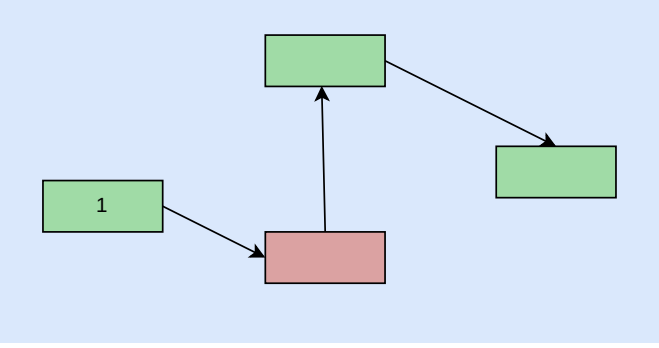


not available* on microprocessor which we're using.

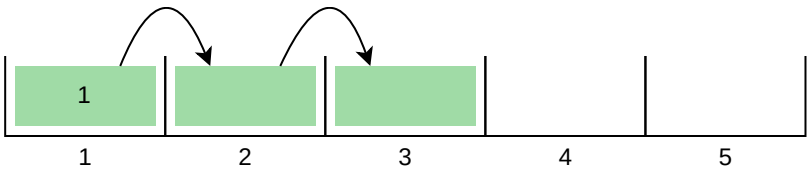
sequencer - optimal data-structure

linked list:



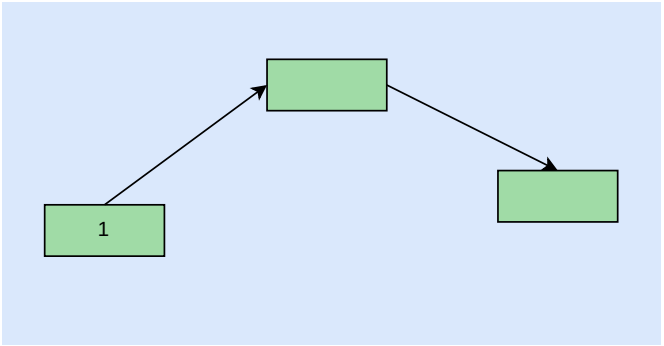


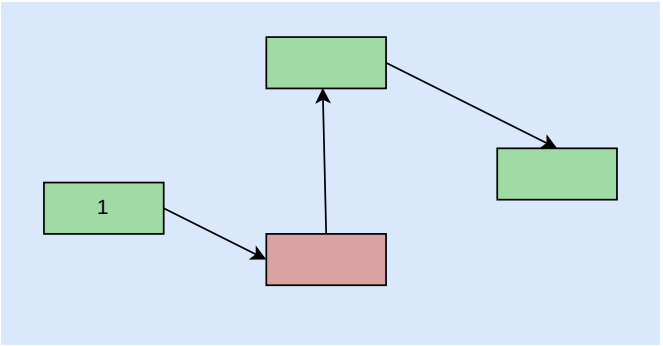
not available* on microprocessor which we're using. But we can emulate its features within an array:



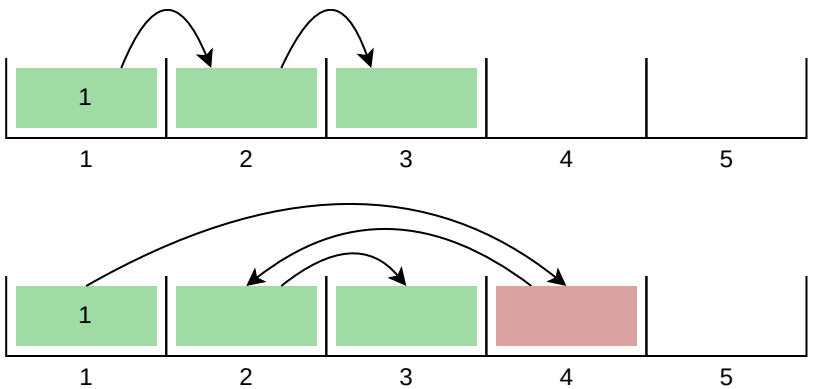
sequencer - optimal data-structure

linked list:



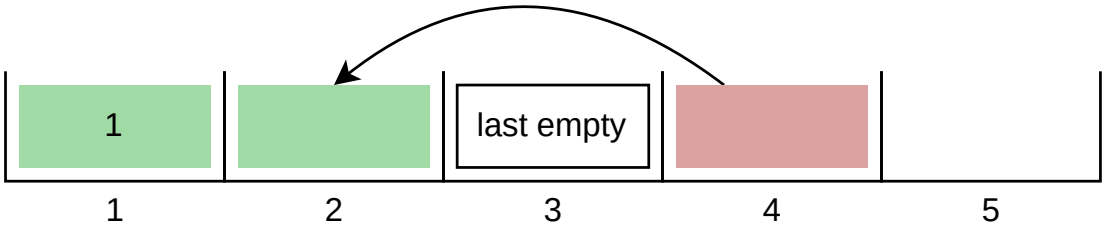


not available* on microprocessor which we're using. But we can emulate its features within an array:



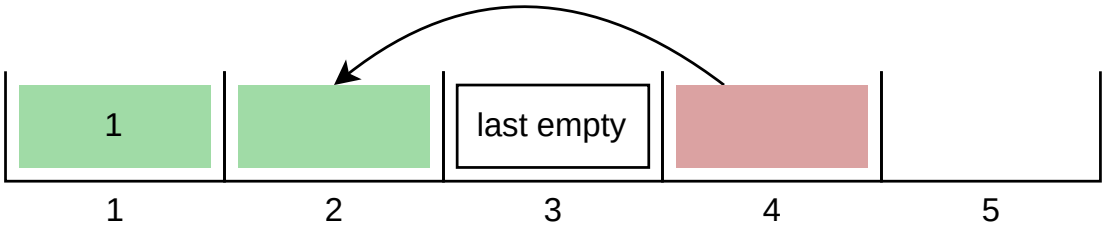
sequencer - optimal data-structure

when an element is removed, its place will be reused for next new element.

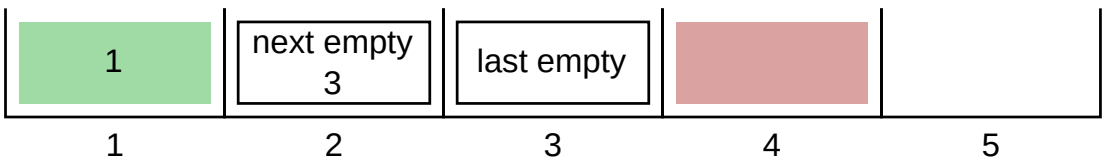


sequencer - optimal data-structure

when an element is removed, its place will be reused for next new element.



if multiple elements are removed, we add new elements in reverse order to their deletion.



this information is stored in the same array. In this way, we avoid fragmentation as well as overhead associated with other data-structures.

Conclusions:

With limited processing power I was able to achieve:

- multiple waveforms (sine, square, saw, triangle)
- 4 note polyphony
- MIDI support (12 octaves*, velocity, cc messages)
- sequencer
  - song timeline
  - chords
  - automations
- unnoticeable delay for external MIDI
- responsive graphics

Thank you for your time