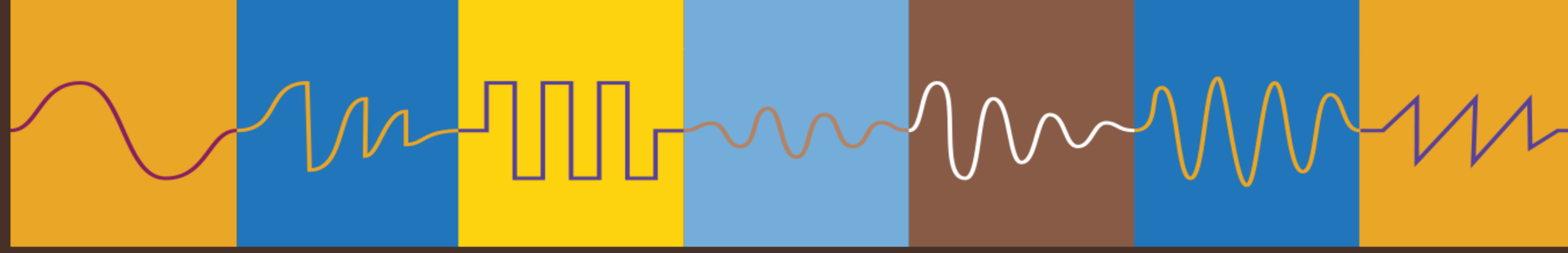
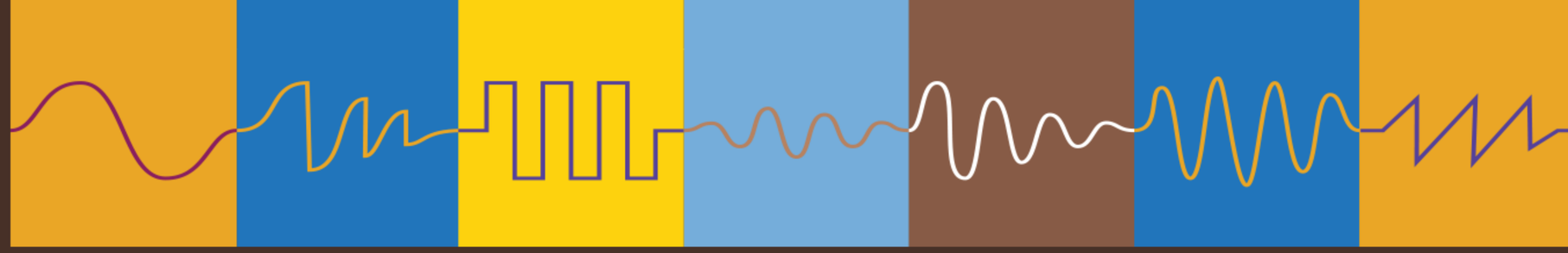
ADC22



EVERY BEAT COUNTS - TEMPO SYNC 101

TAL AVIRAM

ADC22



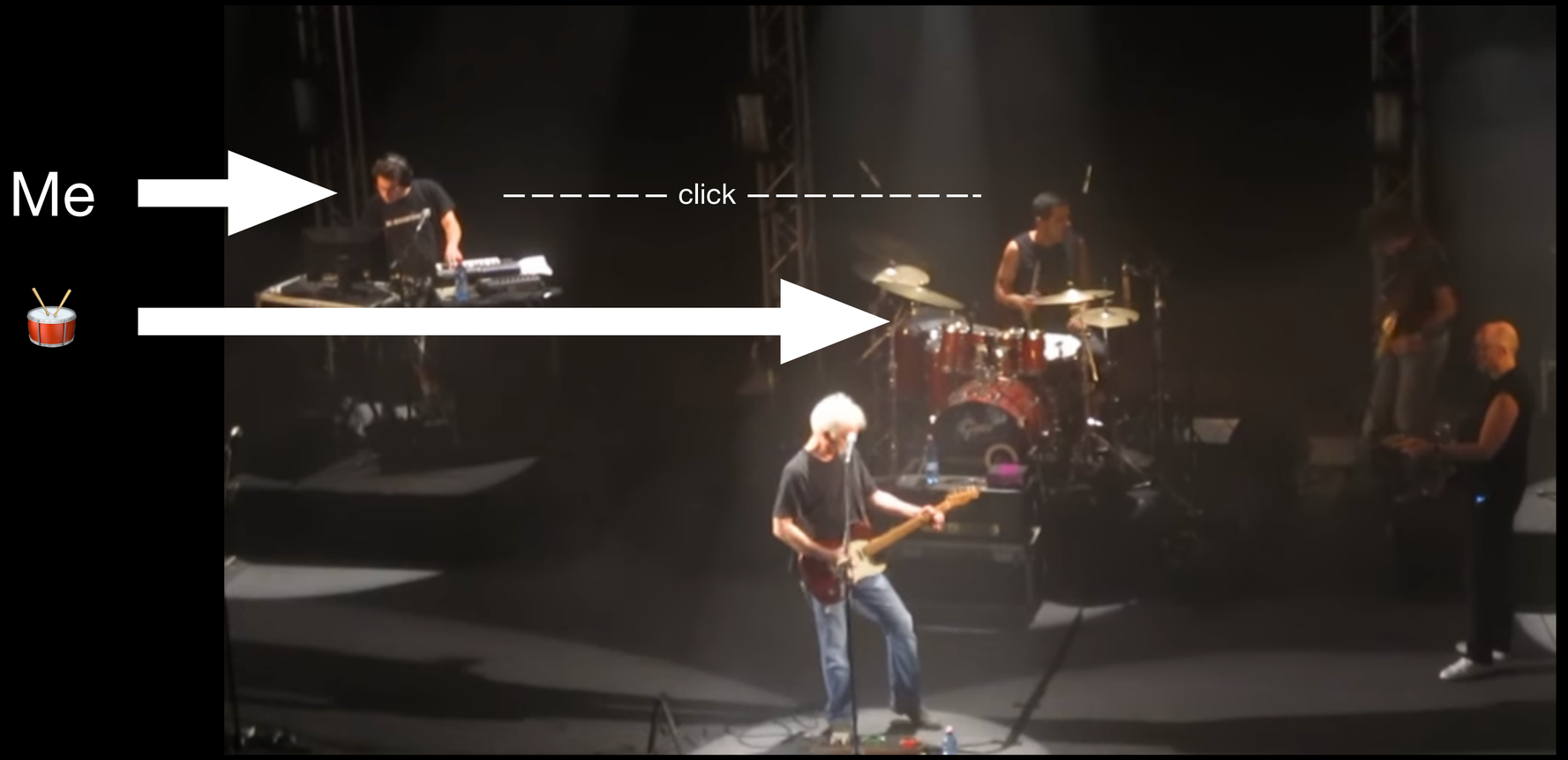
EVERY BEAT COUNTS - TEMPO SYNC 101

https://github.com/talaviram/every_beat_counts_adc2022

TAL AVIRAM

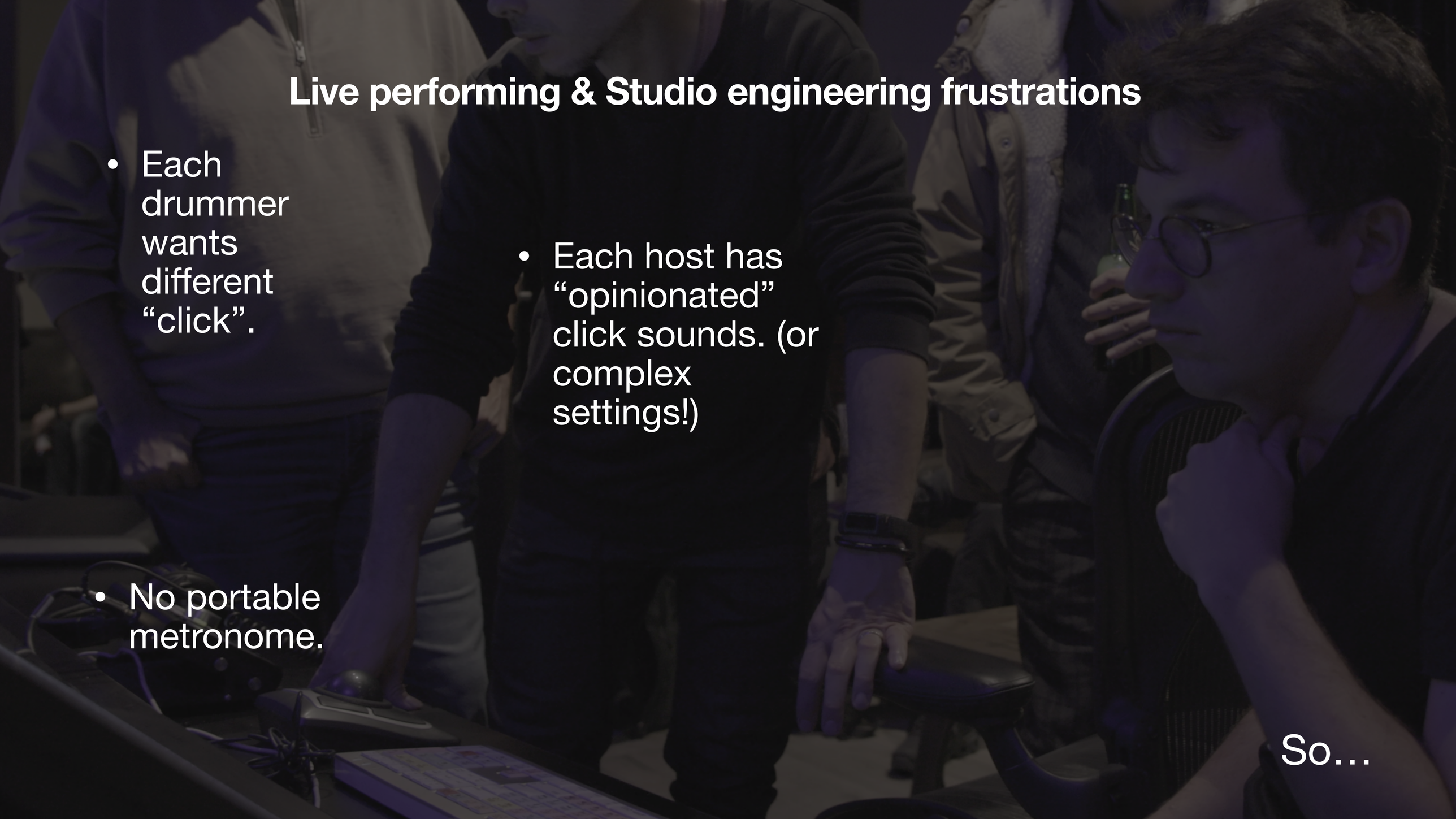
Motivation and Story

why talking about tempo sync?



Who am I?

- Started as a musician and a studio engineer.
- Audio developer by ☀️ @ SoundRadix
  - Check us out https://www.soundradix.com
- Still performing by 🌃.



Live performing & Studio engineering frustrations

- Each drummer wants different “click”.
- Each host has “opinionated” click sounds. (or complex settings!)
- No portable metronome.

So…

I’ve made a metronome

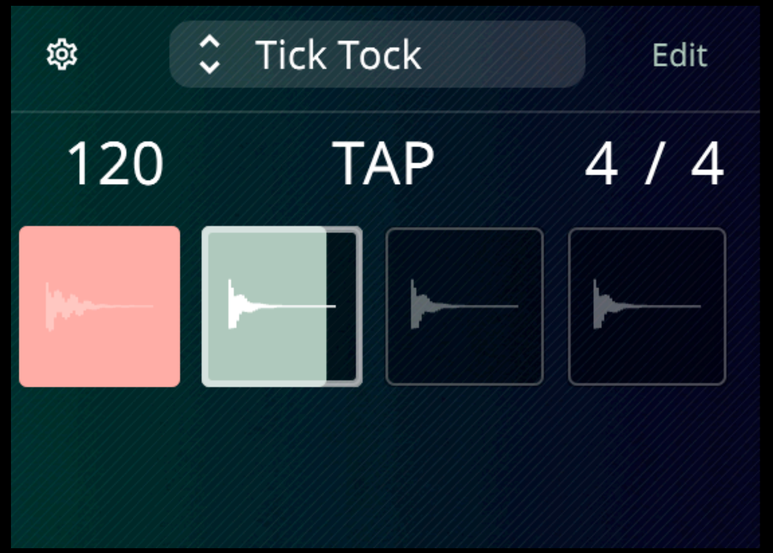
- Shameless Promotion - TICK Metronome
  - https://tick.talaviram.com
  - Free and Cross-Platform:

    macOS, Windows, Linux, *iOS, Android*
  - Open-Source:
    - https://www.github.com/talaviram/TICK

Simple ah?

Simple. Not Straightforward

- Lack of online material about ‘Tempo Syncing’.
- Let’s make a talk about it!



What’s Tempo Sync

Simplest Tempo Sync

```
double bpmToMillis (double newBpm)
{
    const auto beatPerSecond = newBpm / 60.0;
    // returns beat per millisecond...
    return beatPerSecond / 1000.0;
}

void process (float** audioBuffer, double currentHostBpm)
{
    constexpr auto BASE_QUARTER = 4;
    auto musicalDelayParam = 8; // eight note. 1/8

    double delayInMs =
                bpmToMillis (currentHostBpm) / (BASE_QUARTER / musicalDelayParam);

    // multiply by sample-rate...
    // use params in audio sample context
}
```

Simplest Tempo Sync

- That’s it! We got tempo-sync!
  - Well… kinda.

What’s Tempo Sync?

- Simple Tempo Sync:
  - Pros:
    - ✓ It syncs tempo value!
    - ✓ Easy to implement.
  - Cons:
    - No real “sync”
  - No broad (or better) term to describe it (as-far-as-I-know)
  - Tempo Sync (imho) is being mis-used.

Introduction

Why Tempo Sync?

- Sequencing
- Synth Arpeggiators
- LFOs
- Drum Machines
- External Devices

Time vs Musical Time

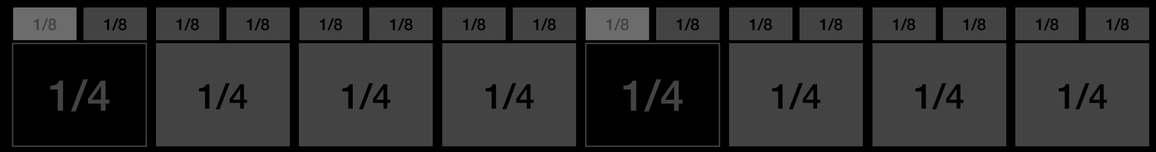
Wall-Time

- Wall-Time is described in absolute lengths:
  - Every second is exactly the same length as the previous one... (unless you’re accelerating in space)

  - The wall-time clock ticks at **constant** intervals. (hopefully…)

Musical-Time

- Musical time is described in absolute lengths (1/4, 1/8 …)
  - Each division describes the same amount of subdivisions.
  - Tempo (bpm) can fluctuate.



- The musical-time clock (bpm) can **vary** throughout the music.

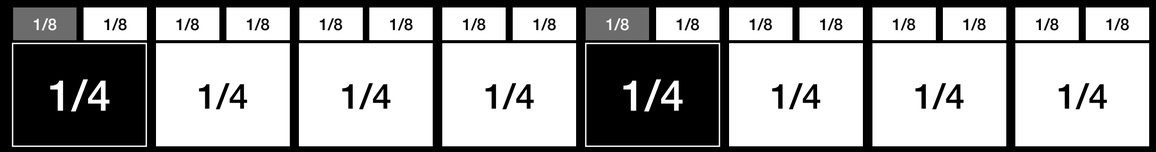
Wall-Time

- Wall-Time is described in absolute lengths:
  - Every second is exactly the same length as the previous one... (unless you’re accelerating in space)

  - The wall-time clock ticks at **constant** intervals. (hopefully…)

Musical-Time

- Musical time is described in absolute lengths (1/4, 1/8 …)
  - Each division describes the same amount of subdivisions.
  - Tempo (bpm) can fluctuate.



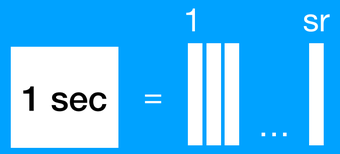
- The musical-time clock (bpm) can **vary** throughout the music.

Time vs Musical Time

In the Digital Domain

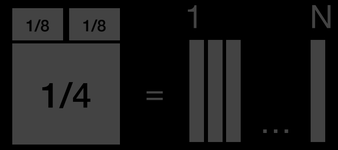
Digital (Audio) Sample

- Smallest unit of time in digitized (discrete) audio.
- Specific interval between two consecutive samples.
- We call this interval ‘sample rate’.
- Sample Rate is measured in Hz (events-per-second).
  - E.g. 44.1kHz - 44100 samples each one 1/44100 of a second.



Digital Music Unit

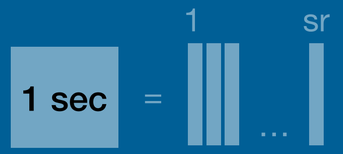
- Smallest unit of time used for sequencing events (in a musical context).
- Interval between 2 consecutive events:
  - PPQ(n) / TPQN - Pulses/Ticks Per Quarter Note



  - This resolution is a measure of time relative to **tempo** since the tempo defines the length of a quarter note and so the duration of each pulse.

Digital (Audio) Sample

- Smallest unit of time in digitized (discrete) audio.
- Specific interval between two consecutive samples.
- We call this interval ‘sample rate’.
- Sample Rate is measured in Hz (events-per-second).
  - E.g. 44.1kHz - 44100 samples each one 1/44100 of a second.

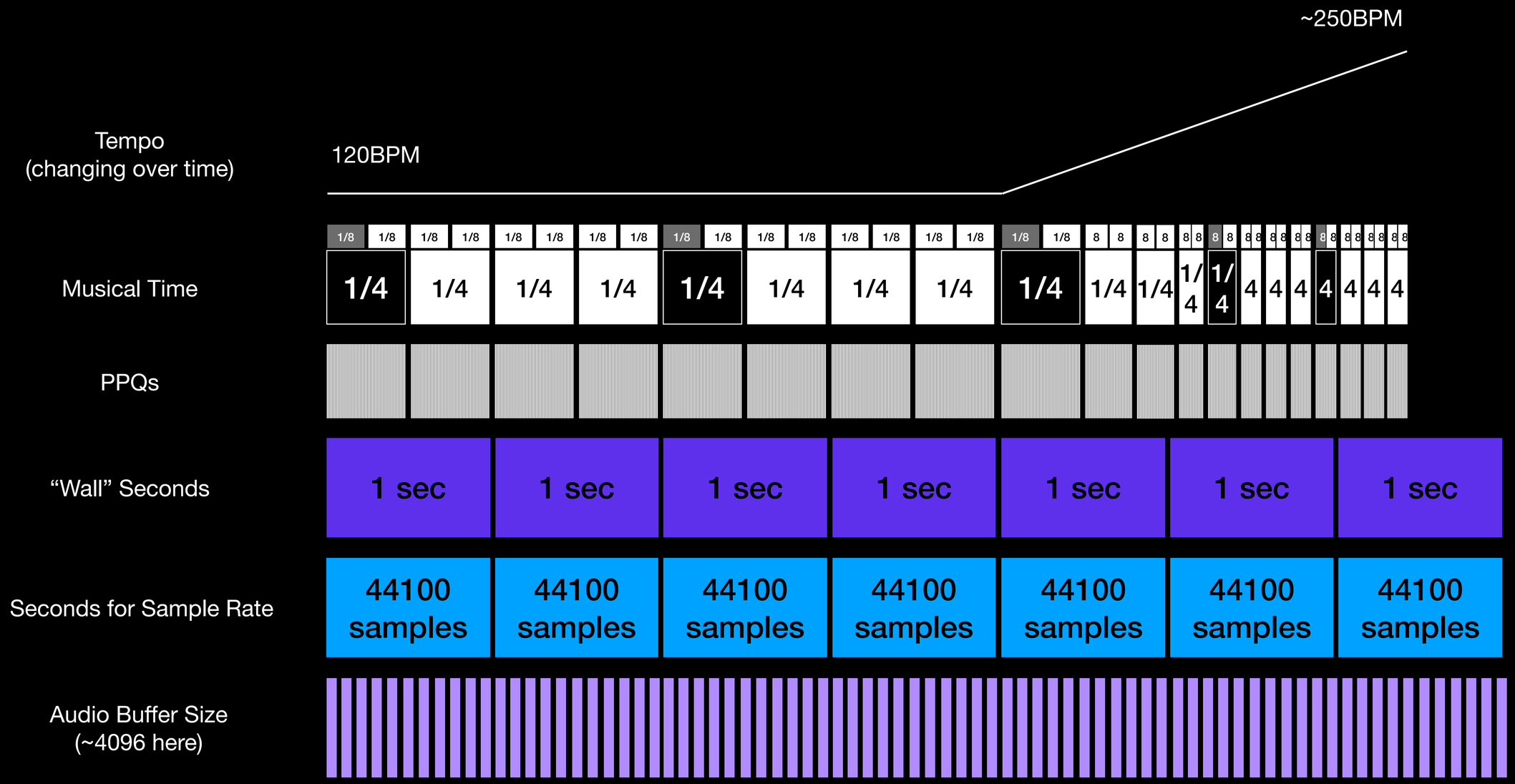


Digital Music Unit

- Smallest unit of time used for sequencing events (in a musical context).
- Interval between 2 consecutive events:
  - PPQ(n) / TPQN - Pulses/Ticks Per Quarter Note

  - This resolution is a measure of time relative to **tempo** since the tempo defines the length of a quarter note and so the duration of each pulse.

Summary


~250BPM
Tempo
(changing over time)
120BPM
Musical Time
1/8
1/4
PPQs
“Wall” Seconds
1 sec
Seconds for Sample Rate
44100
samples
Audio Buffer Size
(~4096 here)

Summary

Musical-Time

Min:Sec Upper Ruler

Wall-Time

Bar:Beats Upper Ruler

Let’s sync a metronome…

Let’s sync a metronome

Ingredients (provided by the host)

- ppq position (transport location in ppq’s)
  - Usually we’ll have this in a normalized float (total_ppqs/ppqn).

    eg: if 1200 *ppq* and 960 *ppqn* then 1200/960 = 1.25
- bpm / tempo
  - how fast are we pulsing?
- time signature
  - numerator & denominator
- More… (we’ll evaluate them later)

Let’s build a metronome

c6b09ca

TempoUtils - class for doing all the tempo-sync calculations for us.

- Basic JUCE Template Plug-In (Projucer & CMake)
- TempoUtils Class (Calculations from host’s playhead)

```
int TempoUtils::getLastWholeQuarter() const
{
    return static_cast<int> (floor (ppq));
}
```

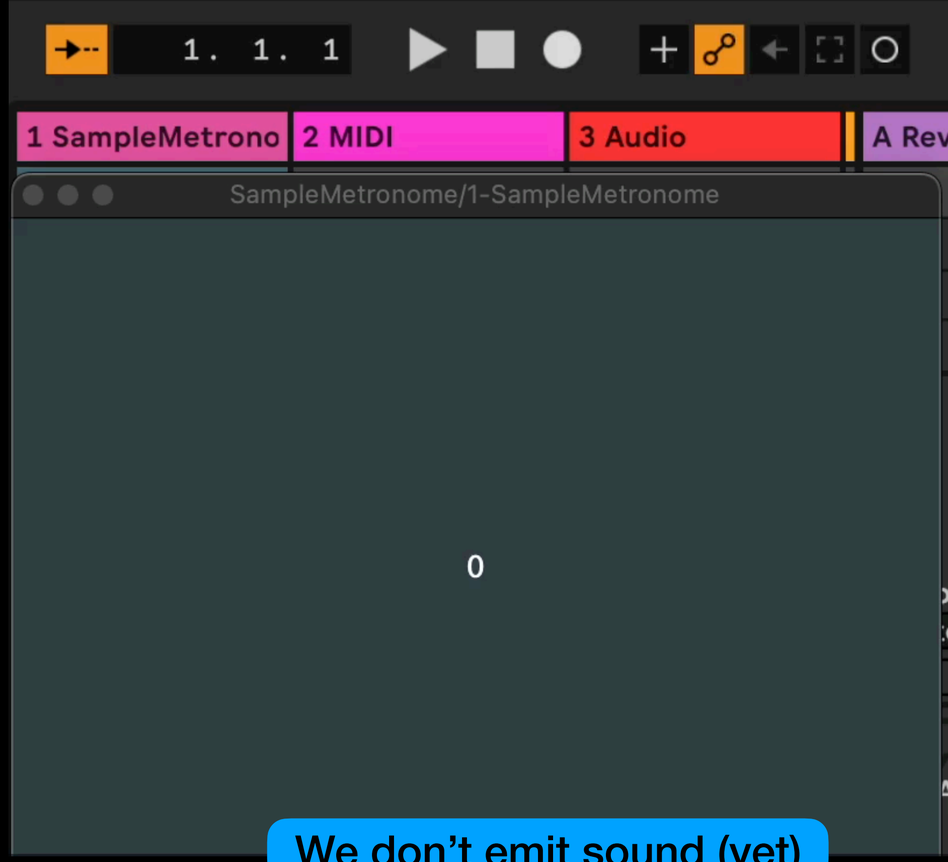
processBlock()

```
if (auto* playhead = getPlayHead())
{
    const auto pos = playhead->getPosition();
    if (pos.hasValue())
        tempoUtils.calculatePosition (*pos);
}
```

Simple PPQ Counter

1c9fe31 Show quarters in UI.

```
void SimpleMetronomeAudioProcessorEditor::paint
(Graphics& g)
{
    auto& tempoState = audioProcessor.getTempoState();
    const auto quarter =
                    tempoState.getLastWholeQuarter();

    g.fillAll (getLookAndFeel().
               findColour
               (ResizableWindow::backgroundColourId));
    g.setColour (Colours::white);
    g.setFont (15.0f);

    g.drawFittedText (
         String (quarter), getLocalBounds(),
         Justification::centred, 1);
}
```



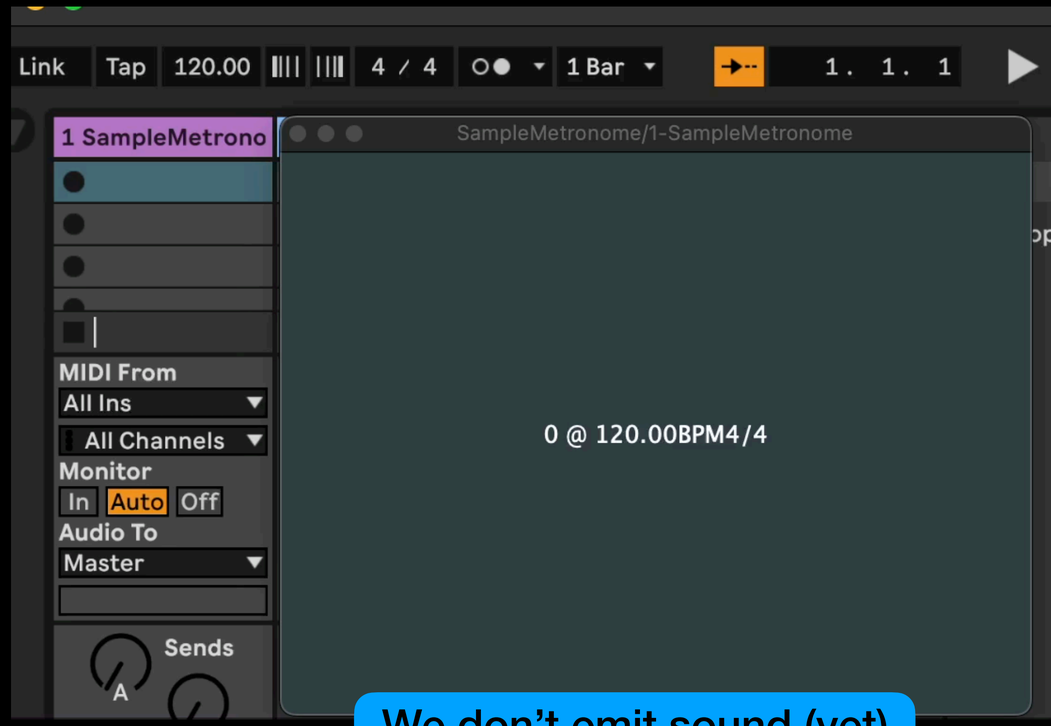
We don’t emit sound (yet)

Tempo & Time Signature

4737e2e

Get more tempo syncing data to our util class.

```
void SimpleMetronomeAudioProcessorEditor::paint (Graphics& g)
{
    auto& tempoState = audioProcessor.getTempoState();
    const auto quarter = tempoState.getLastWholeQuarter();
    g.fillAll (getLookAndFeel().findColour
               (ResizableWindow::backgroundColourId));

    g.setColour (tempoState.isPlaying() ?
                 Colours::lightgreen : Colours::white);
    g.setFont (15.0f);
    g.drawFittedText (String (quarter) + " @ " +
                      String (tempoState.getTempo(), 2) + "BPM" +
                      String (tempoState.getTimeSignature().numerator) +
                      "/" +
                      String (tempoState.getTimeSignature().denominator),
                      getLocalBounds(), Justification::centred, 1);
}
```



We don’t emit sound (yet)

Prepare for pulse ticking

3cb1a89 Quarter in seconds.

```
double TempoUtils::getQuarterInSeconds() const
{
    // beat per second
    return bpm / 60.0;
}
```

* sample-rate uses hertz (which is equivalent to second)

cc11da4 Add sub quarter getter.

```
double TempoUtils::getSubQuarterDivision() const
{
    return ppq - getLastWholeQurater();
}
```

* the ratio of current unfinished ppq

We ‘tick’ at quarter lengths!

5af2802 Impulse in quarter length samples.

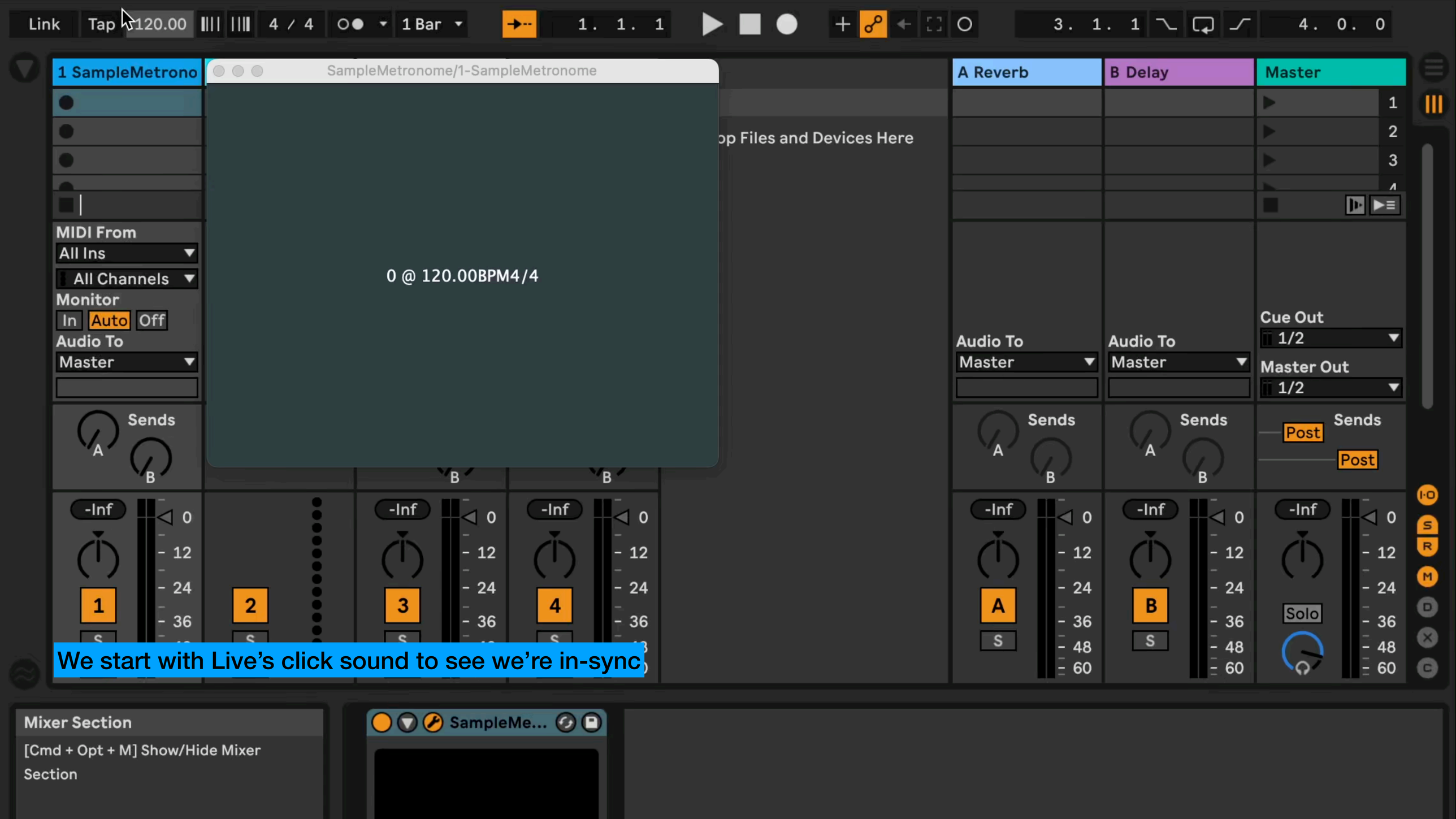
```
if (auto* playhead = getPlayHead())
{
    const auto pos = playhead->getPosition();
    if (pos.hasValue())
        tempoUtils.calculatePosition (*pos);
}

const auto quarterInSamples = getSampleRate() / tempoUtils.getQuarterInSeconds();
ppqSamplesCounter += buffer.getNumSamples();
std::optional<int> tickAt;

if (ppqSamplesCounter > quarterInSamples)
{
    ppqSamplesCounter %= roundToInt (quarterInSamples);
    tickAt = buffer.getNumSamples() - ppqSamplesCounter;
    // missed deadline. don't tick
    if (*tickAt < 0)
        tickAt = {};
}

if (tickAt.has_value())
{
    for (int channel = 0; channel < totalNumOutputChannels; ++channel)
        buffer.setSample (channel, *tickAt, 0.5f);
}
```

No visual indication.
“free-sync” pulse at quarter lengths at BPM

We ‘tick’ at host

a5be997 "kinda" synced quarter impulse

```
      const auto quarterInSamples = getSampleRate() / tempoUtils.getQuarterInSeconds();
-     ppqSamplesCounter += buffer.getNumSamples();
-     std::optional<int> tickAt;
+     const auto nextQuarterInSamples = roundToInt ((1.0 - tempoUtils.getSubQuarterDivision()) * quarterInSamples);

-     if (ppqSamplesCounter > quarterInSamples)
-     {
-         ppqSamplesCounter %= roundToInt (quarterInSamples);
-         tickAt = buffer.getNumSamples() - ppqSamplesCounter;
+     std::optional<int> tickAt;

-         // missed deadline. don't tick
-         if (*tickAt < 0)
-             tickAt = {};
-     }
+     if (nextQuarterInSamples < buffer.getNumSamples())
+         tickAt = nextQuarterInSamples;
```


Link
Tap
120.00
4 / 4
1 Bar
1 . 1 . 1
3 . 1 . 1
4 . 0 . 0
1 SampleMetrono
SampleMetronome/1-SampleMetronome
0 @ 120.00BPM4/4
A Reverb
B Delay
Master
MIDI From
All Ins
All Channels
Monitor
In
Auto
Off
Audio To
Master
Sends
Cue Out
1/2
Master Out
Post
-Inf
0
12
24
36
48
60
1
2
3
4
A
B
S
Solo
We start with Live's click sound to see we're in-sync
Mixer Section
[Cmd + Opt + M] Show/Hide Mixer Section
SampleMe...

What’s missing?

a5be997 "kinda" synced quarter impulse

```
      const auto quarterInSamples = getSampleRate() / tempoUtils.getQuarterInSeconds();
-     ppqSamplesCounter += buffer.getNumSamples();
-     std::optional<int> tickAt;
+     const auto nextQuarterInSamples = roundToInt ((1.0 - tempoUtils.getSubQuarterDivision()) * quarterInSamples);

-     if (ppqSamplesCounter > quarterInSamples)
-     {
-         ppqSamplesCounter %= roundToInt (quarterInSamples);
-         tickAt = buffer.getNumSamples() - ppqSamplesCounter;
+     std::optional<int> tickAt;

-         // missed deadline. don't tick
-         if (*tickAt < 0)
-             tickAt = {};
-     }
+     if (nextQuarterInSamples < buffer.getNumSamples())
+         tickAt = nextQuarterInSamples;
```

What’s missing?

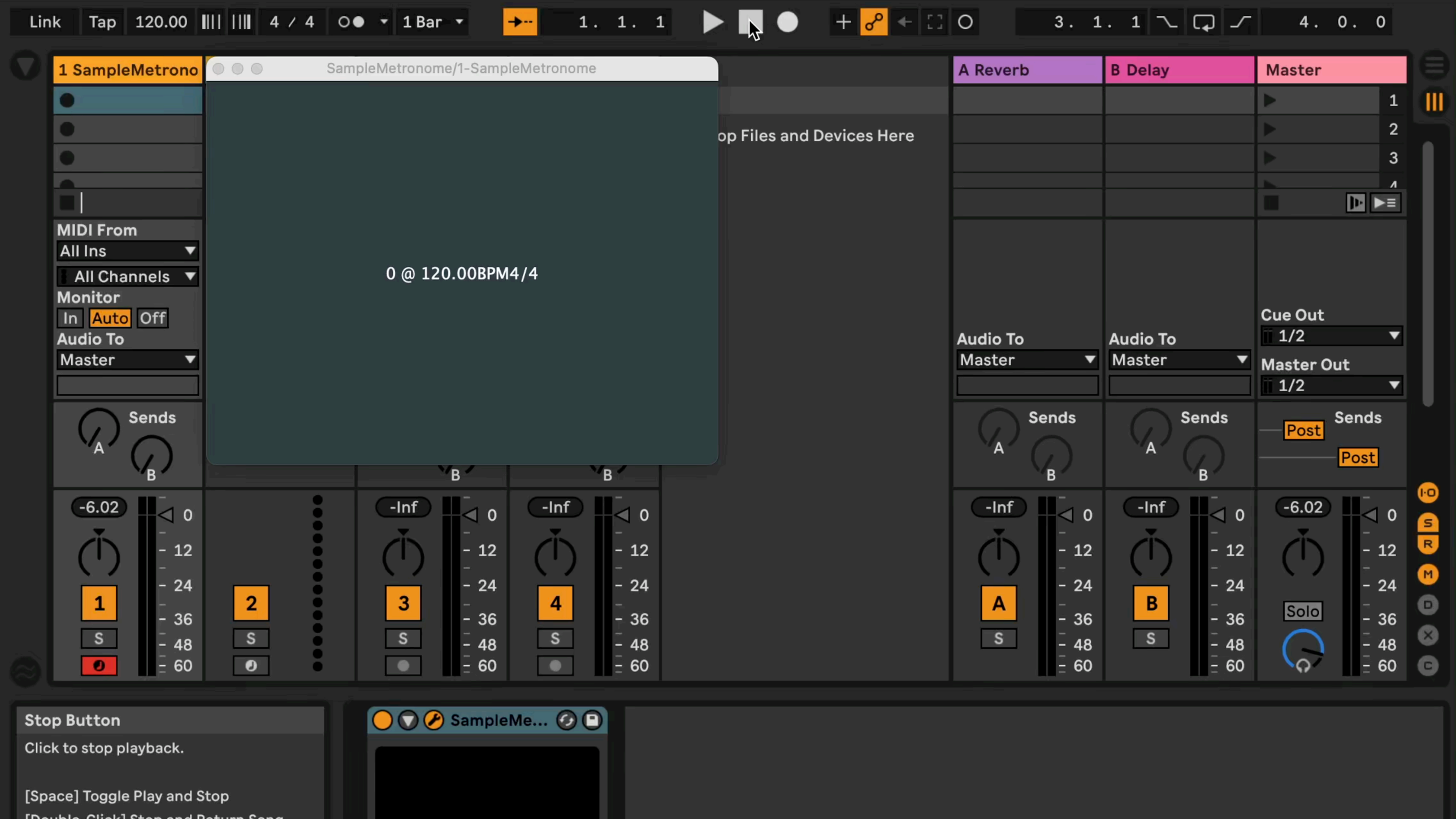
a5be997 "kinda" synced quarter impulse

The First Pulse!

Fix when quarter is exactly on start!

e339696 Handle corner cases of first beat / avoid output when no playback.

```
+     if (! tempoUtils.isPlaying())
+         return;
+
      const auto quarterInSamples = getSampleRate() / tempoUtils.getQuarterInSeconds();
      const auto nextQuarterInSamples = roundToInt ((1.0 - tempoUtils.getSubQuarterDivision()) * quarterInSamples);

      std::optional<int> tickAt;

-     if (nextQuarterInSamples < buffer.getNumSamples())
+     if (tempoUtils.getSubQuarterDivision() == 0.0)
+         tickAt = 0;
+     else if (nextQuarterInSamples < buffer.getNumSamples())
          tickAt = nextQuarterInSamples;
```


Link
Tap
120.00
4 / 4
1 Bar
1. 1. 1
3. 1. 1
4. 0. 0
1 SampleMetrono
SampleMetronome/1-SampleMetronome
0 @ 120.00BPM4/4
op Files and Devices Here
A Reverb
B Delay
Master
MIDI From
All Ins
All Channels
Monitor
In
Auto
Off
Audio To
Master
Sends
A
B
Post
Cue Out
1/2
Master Out
-6.02
-Inf
0
12
24
36
48
60
1
2
3
4
A
B
S
Solo
Stop Button
Click to stop playback.
[Space] Toggle Play and Stop
SampleMe...

Any other “missing” quarters cases?

Bonus question

```
void processBlock (AudioBuffer<float>& buffer, MidiBuffer&)
{
...
    const auto quarterInSamples = getSampleRate() / tempoUtils.getQuarterInSeconds();
    const auto nextQuarterInSamples = roundToInt ((1.0 - tempoUtils.getSubQuarterDivision()) *
                                                  quarterInSamples);

    std::optional<int> tickAt;

    if (tempoUtils.getSubQuarterDivision() == 0.0)
        tickAt = 0;
    else if (nextQuarterInSamples < buffer.getNumSamples())
        tickAt = nextQuarterInSamples;

    if (tickAt.has_value())
    {
        for (int channel = 0; channel < totalNumOutputChannels; ++channel)
            buffer.setSample (channel, *tickAt, 0.5f);
    }
...
}
```

Handle large buffers

dc26df9 Handle corner case(s) if multiple ticks in a single buffer.

```
      const auto quarterInSamples = getSampleRate() / tempoUtils.getQuarterInSeconds();
-     const auto nextQuarterInSamples = roundToInt ((1.0 - tempoUtils.getSubQuarterDivision()) * quarterInSamples);
-
-     std::optional<int> tickAt;
+     auto nextQuarterInSamples = roundToInt ((1.0 - tempoUtils.getSubQuarterDivision()) * quarterInSamples);

      if (tempoUtils.getSubQuarterDivision() == 0.0)
-         tickAt = 0;
-     else if (nextQuarterInSamples < buffer.getNumSamples())
-         tickAt = nextQuarterInSamples;
+         nextQuarterInSamples = 0;

-     if (tickAt.has_value())
+     while (nextQuarterInSamples < buffer.getNumSamples())
      {
-         for (int channel = 0; channel < totalNumOutputChannels; ++channel)
-             buffer.setSample (channel, *tickAt, 0.5f);
+         std::optional<int> tickAt;
+
+         if (nextQuarterInSamples < buffer.getNumSamples())
+             tickAt = nextQuarterInSamples;
+
+         if (tickAt.has_value())
+         {
+             for (int channel = 0; channel < totalNumOutputChannels; ++channel)
+                 buffer.setSample (channel, *tickAt, 0.5f);
+
+             // move to next tick
+             nextQuarterInSamples = roundToInt (nextQuarterInSamples + quarterInSamples);
+         }
```

Time Signature

Support time signatures rather than quarters…

Get ready for beats…

07a7713 Refactored quarterInSamples to beatInSamples

```
-     const auto quarterInSamples = getSampleRate() / tempoUtils.getQuarterInSeconds();
-     auto nextQuarterInSamples = roundToInt ((1.0 - tempoUtils.getSubQuarterDivision()) * quarterInSamples);
+     const auto beatInSamples = getSampleRate() / tempoUtils.getQuarterInSeconds();
+     auto nextBeatInSamples = roundToInt ((1.0 - tempoUtils.getSubQuarterDivision()) * beatInSamples);

      if (tempoUtils.getSubQuarterDivision() == 0.0)
-         nextQuarterInSamples = 0;
+         nextBeatInSamples = 0;

-     while (nextQuarterInSamples < buffer.getNumSamples())
+     while (nextBeatInSamples < buffer.getNumSamples())
      {
          std::optional<int> tickAt;

-         if (nextQuarterInSamples < buffer.getNumSamples())
-             tickAt = nextQuarterInSamples;
+         if (nextBeatInSamples < buffer.getNumSamples())
+             tickAt = nextBeatInSamples;

          if (tickAt.has_value())
          {
              for (int channel = 0; channel < totalNumOutputChannels; ++channel)
                  buffer.setSample (channel, *tickAt, 0.5f);

              // move to next tick
-             nextQuarterInSamples = roundToInt (nextQuarterInSamples + quarterInSamples);
+             nextBeatInSamples = roundToInt (nextBeatInSamples + beatInSamples);
```

Get ready for beats…

5c83772 Denominator ratio to quarters.

- Calculate the ratio between quarters to our time-signature’s denominator.

```
double TempoUtils::getDenominatorRatioToQuarters() const
{
    return 4.0 / static_cast<double> (getTimeSignature().denominator);
}
```

- For example:

4/8 = 1/2 (0.5). So our ‘ticks’ will be every half a quarter…

Get “pulses” per time signature denominator

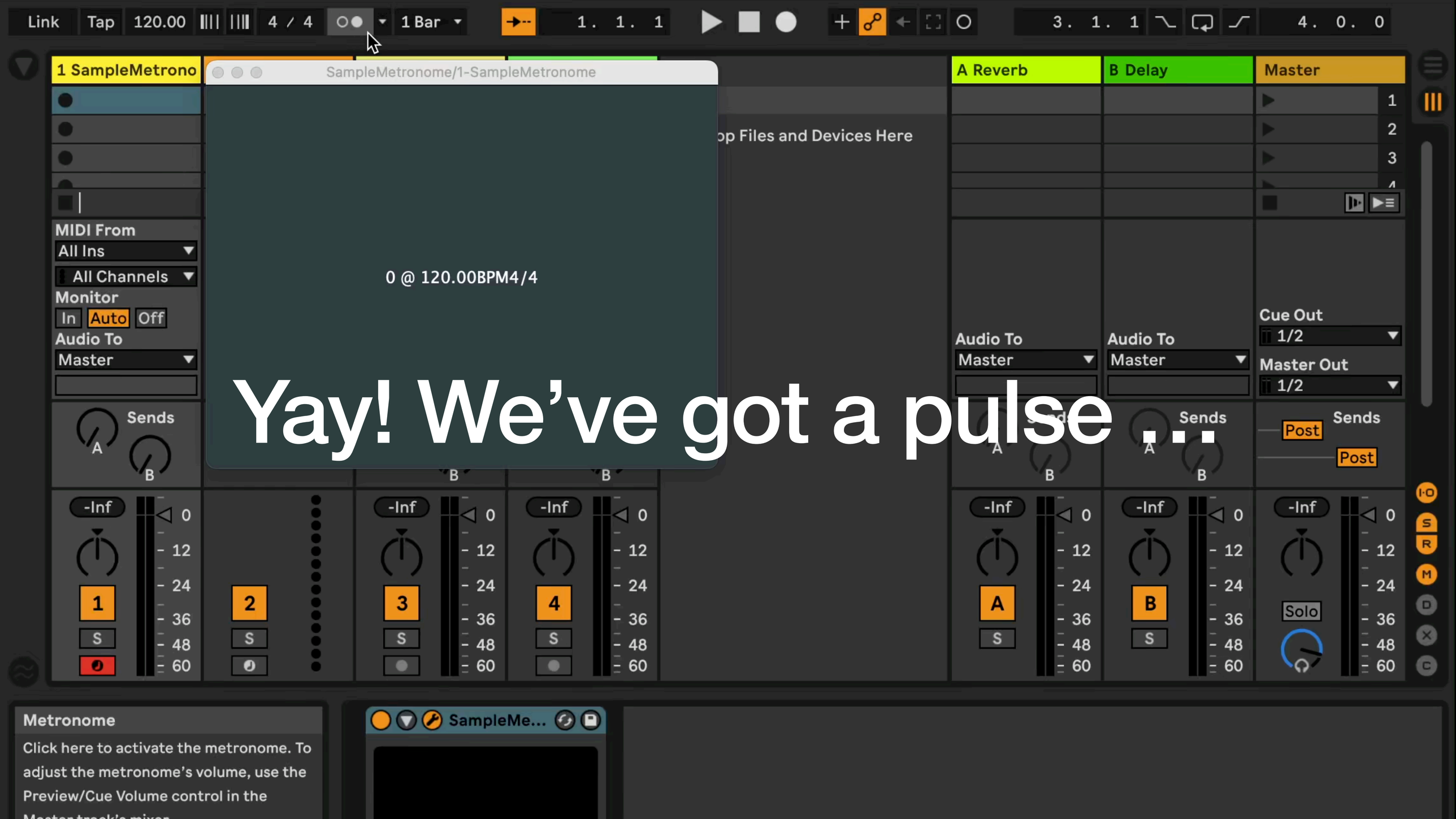
4945f69 Getter for beats based on denominator instead of quarters.

```
int TempoUtils::getLastWholeAsDenominator() const
{
    return static_cast<int> (floor (ppq / getDenominatorRatioToQuarters()));
}

double TempoUtils::getSubDivisionForDenominator() const
{
    return (ppq / getDenominatorRatioToQuarters()) - getLastWholeAsDenominator();
}
```

Glue everything…

4eabe8d Support time-signature ticks instead of quarters.

```
-     const auto beatInSamples = getSampleRate() / tempoUtils.getQuarterInSeconds();
-     auto nextBeatInSamples = roundToInt ((1.0 - tempoUtils.getSubQuarterDivision()) * beatInSamples);
+     const auto denominatorRatio = tempoUtils.getDenominatorRatioToQuarters();
+     const auto beatInSamples = (getSampleRate() / tempoUtils.getQuarterInSeconds()) * denominatorRatio;
+     auto nextBeatInSamples = roundToInt ((1.0 - tempoUtils.getSubDivisionForDenominator()) * beatInSamples);

-     if (tempoUtils.getSubQuarterDivision() == 0.0)
+     if (tempoUtils.getSubDivisionForDenominator() == 0.0)
```


Link
Tap
120.00
4 / 4
1 Bar
1. 1. 1
3. 1. 1
4. 0. 0
1 SampleMetrono
SampleMetronome/1-SampleMetronome
0 @ 120.00BPM4/4
A Reverb
B Delay
Master
op Files and Devices Here
MIDI From
All Ins
All Channels
Monitor
In
Auto
Off
Audio To
Master
Sends
Cue Out
1/2
Master Out
1/2
Post
Yay! We've got a pulse ...
-Inf
0
12
24
36
48
60
1
2
3
4
A
B
S
Solo
Metronome
Click here to activate the metronome. To adjust the metronome's volume, use the Preview/Cue Volume control in the
SampleMe...

1..2..3..4

“One” more thing?

- Where’s our one?
- Why is the ‘one’ is important?
  - Beat location is crucial part of music.
  - Synced sequences might need bar context!
  - Harmonic Rhythm & Chord Progression

Prepare for accent

86d6b2f Refactor impulse gain to a const towards accents.

```
+           const auto beatGain = 0.5f;
            std::optional<int> tickAt;

            if (nextBeatInSamples < buffer.getNumSamples())
  @@ -148,7 +149,7 @@ void SimpleMetronomeAudioProcessor::processBlock (AudioBuffer<float>& buffer, Mi
            if (tickAt.has_value())
            {
                for (int channel = 0; channel < totalNumOutputChannels; ++channel)
-                   buffer.setSample (channel, *tickAt, 0.5f);
+                   buffer.setSample (channel, *tickAt, beatGain);
```

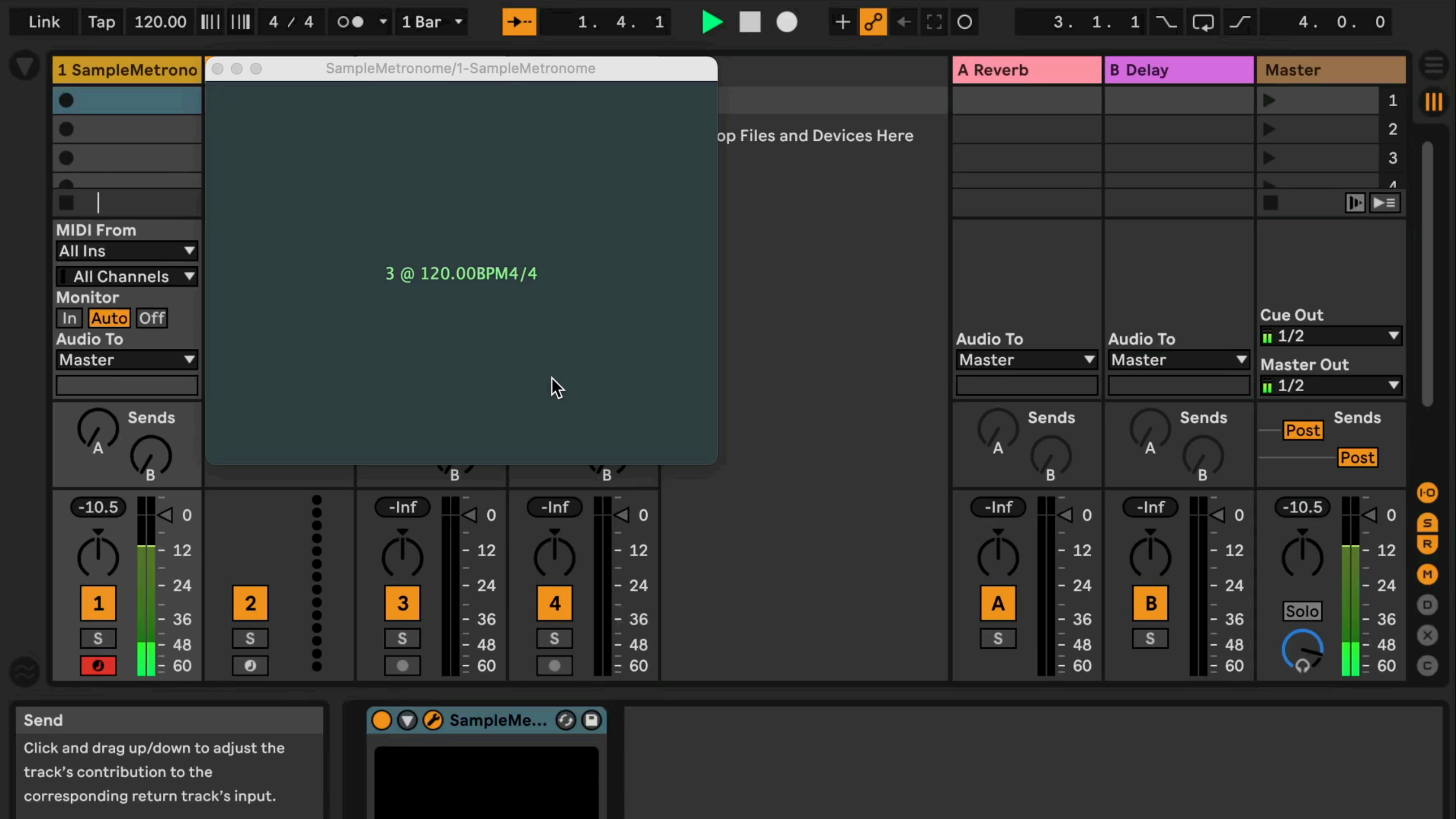
eefb9bc Get current beat for signature.

```
int TempoUtils::getCurrentBeat() const
{
    return static_cast<int> (std::floor (ppq / getDenominatorRatioToQuarters())) % getTimeSignature().numerator + 1;
}
```

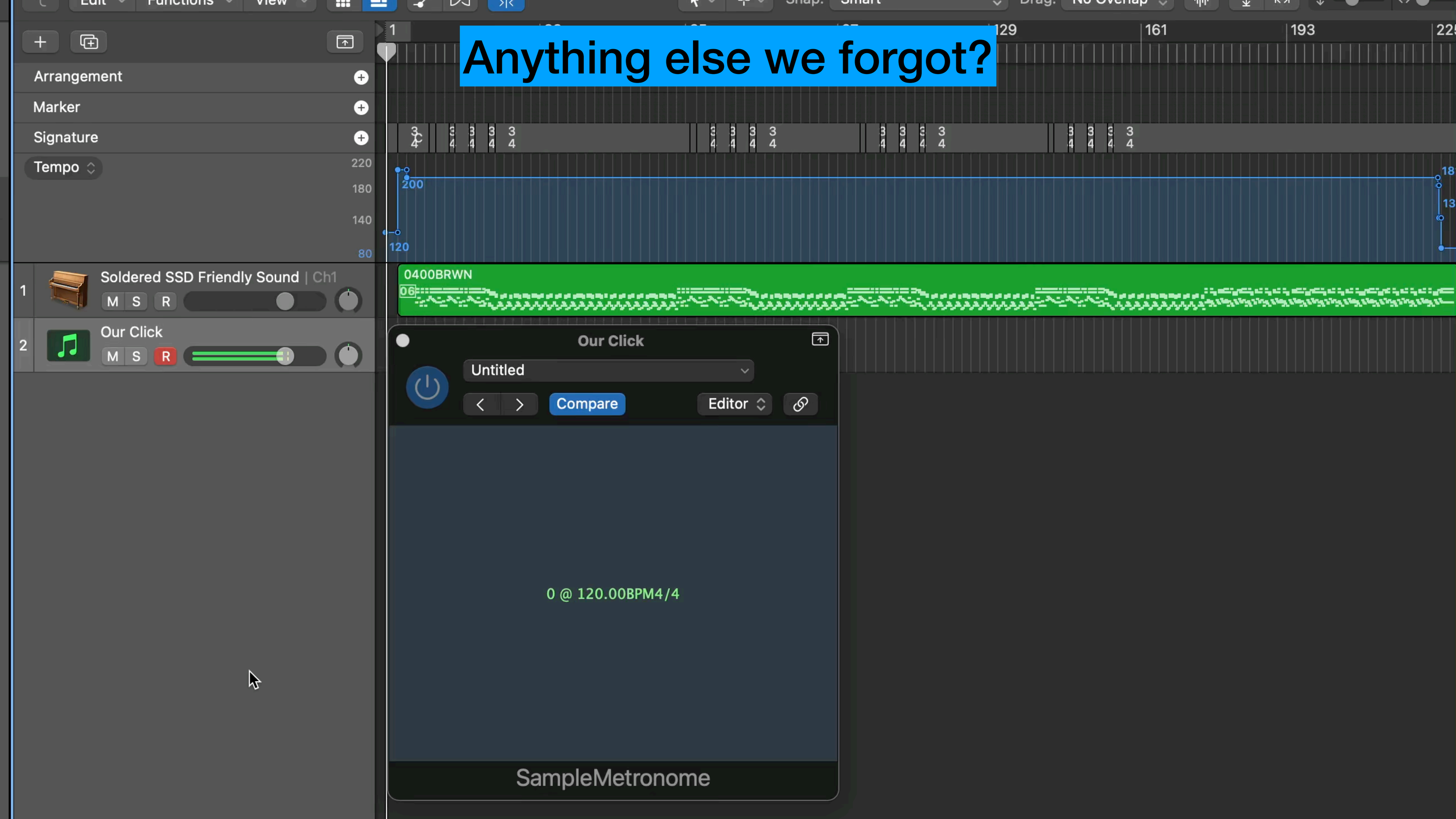
Impulse with accent on “ones”.

bbc6eb5 Our sample got accents! That's it?

```
    const auto denominatorRatio = tempoUtils.getDenominatorRatioToQuarters();
    const auto beatInSamples = (getSampleRate() / tempoUtils.getQuarterInSeconds()) * denominatorRatio;
    auto nextBeatInSamples = roundToInt ((1.0 - tempoUtils.getSubDivisionForDenominator()) * beatInSamples);
+   auto nextBeat = tempoUtils.getCurrentBeat() + 1;

    if (tempoUtils.getSubDivisionForDenominator() == 0.0)
        nextBeatInSamples = 0;

    while (nextBeatInSamples < buffer.getNumSamples())
    {
-       const auto beatGain = 0.5f;
+       // beats starts at 1; no modulo
+       if (nextBeat > tempoUtils.getTimeSignature().numerator)
+           nextBeat = 1;
+
+       const auto beatGain = nextBeat == 1 ? 0.8f : 0.3f;
        std::optional<int> tickAt;

        if (nextBeatInSamples < buffer.getNumSamples())
@@ -153,6 +158,7 @@ void SimpleMetronomeAudioProcessor::processBlock (AudioBuffer<float>& buffer, Mi

            // move to next tick
            nextBeatInSamples = roundToInt (nextBeatInSamples + beatInSamples);
+           nextBeat++;
```


Link
Tap
120.00
4 / 4
1 Bar
1 . 4 . 1
3 . 1 . 1
4 . 0 . 0
1 SampleMetrono
SampleMetronome/1-SampleMetronome
3 @ 120.00BPM4/4
op Files and Devices Here
A Reverb
B Delay
Master
MIDI From
All Ins
All Channels
Monitor
In
Auto
Off
Audio To
Master
Sends
A
B
Cue Out
1/2
Master Out
1/2
Post
-10.5
-Inf
0
12
24
36
48
60
1
2
3
4
A
B
S
Solo
Send
Click and drag up/down to adjust the track's contribution to the corresponding return track's input.
SampleMe...

Anything else we forgot?


Anything else we forgot?
Arrangement
Marker
Signature
Tempo
220
180
140
80
200
120
1
129
161
193
Soldered SSD Friendly Sound | Ch1
M S R
0400BRWN
06
2
Our Click
M S R
Our Click
Untitled
Compare
Editor
0 @ 120.00BPM4/4
SampleMetronome

Let’s build a metronome

bbc6eb5 Our sample got accents! That's it?

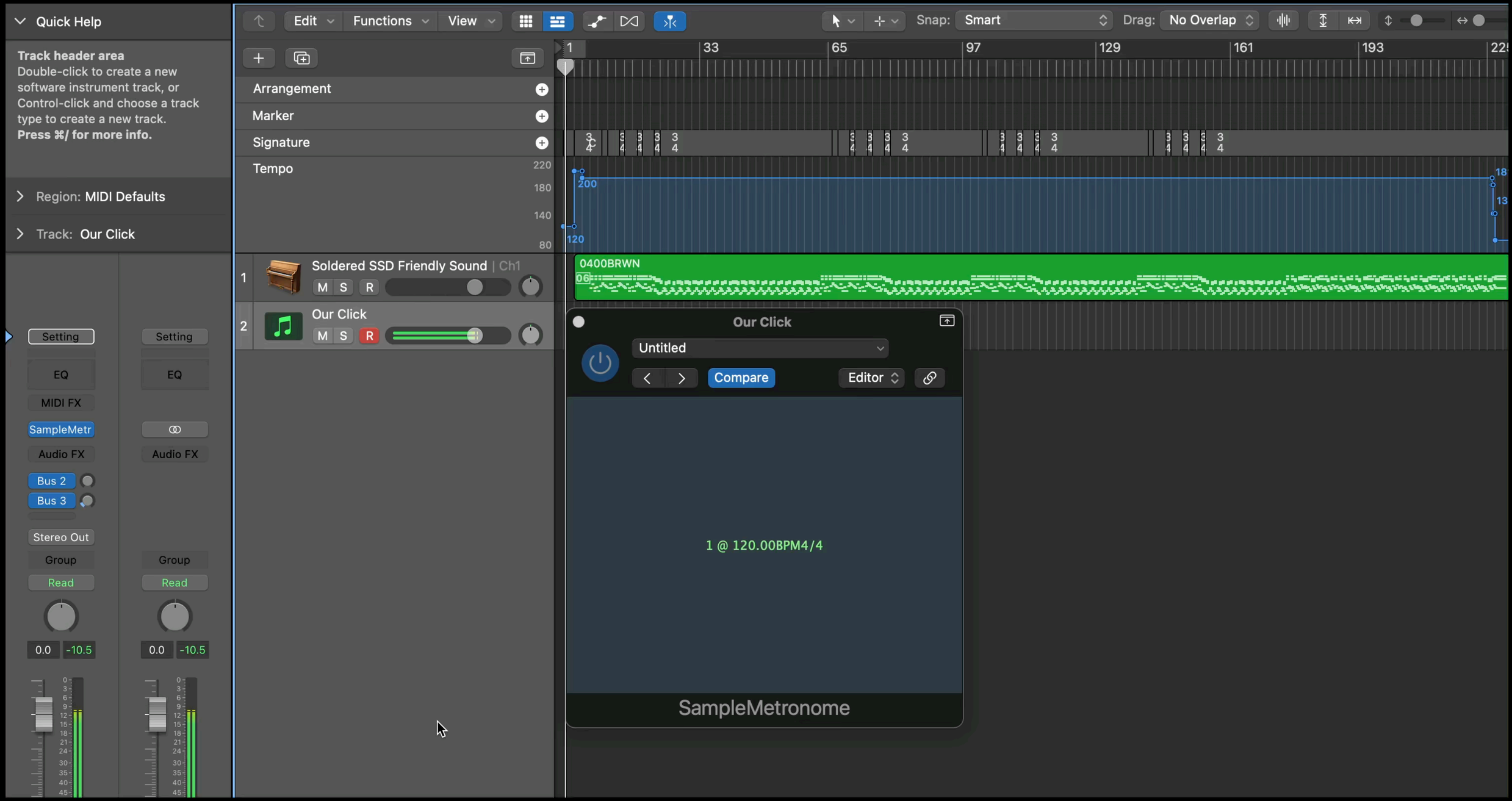
- Q: So what’s the problem?
  - A: Time signature changes mess with our accent.

- Can we solve it?
  - Last bar position to the rescue!*

```
28  28    int TempoUtils::getCurrentBeat() const
29  29    {
30      -     return static_cast<int> (std::floor (ppq / getDenominatorRatioToQuarters())) % getTimeSignature().numerator + 1;
    30  +     return static_cast<int> (std::floor ((ppq - lastBarPpq) / getDenominatorRatioToQuarters())) % getTimeSignature().numerator + 1;
31  31    }
```

* Limited availability. Might not be supported by all hosts/formats

76594ba Support changing signatures.



Summary

Conclusions | Gotchas | Ideas

- Impulse Sample.
  Actual products are more complex - Audio Samples, Synthesis, SRC, Multi-Layer, “cycle” smoothing.
- Time and Musical Time domains won’t always play nice with floats.
  - Keep musical events in musical time;Keep audio events in clock-time.
  - See Ardour 7 release notes ;)
  - Per-Block not Per-Sample

Summary

Conclusions | Gotchas | Ideas

- Time-Signature requires some orientation (eg last bar):
  - JUCE (currently) lacks proper bar position for AAX/Pro Tools
  - Some hosts start counting bars different!
- We only know last bar position; but not what bar it is.
  - AAX, CLAP and LV2 do provide actual bar number.
  - ARA adds greater MusicalContext.

Thank You

https://github.com/talaviram/every_beat_counts_adc2022

Questions

Questions

@talaviram