

```

1 class SleepyDetector {
2     static constexpr double TimerLengthMs = 20.;
3     static constexpr double Eps = 1e-5;
4
5     class Mono {
6         struct Sample {
7             Sample()
8                 : val(0.) { }
9
10            void prepare() noexcept { val = 0.; }
11
12            bool operator()(double x) noexcept {
13                auto const xAbs = std::abs(x);
14                auto const dist = val - xAbs;
15                bool e = dist > Eps;
16                val = xAbs;
17                return e;
18            }
19
20            double val;
21        };
22
23    public:
24        Mono()
25            : sample()
26              , timerIndex(0)
27              , timerLength(0)
28              , ringing(false) { }
29
30        void prepare(int _timerLength) noexcept {
31            sample.prepare();
32            timerLength = _timerLength;
33            timerIndex = 0;
34            ringing = false;
35        };
36
37        void triggerNoteOn() noexcept {
38            timerIndex = 0;
39            ringing = true;
40        }
41
42        void operator()(double* smpls, int start, int end) noexcept {
43            if (!ringing)
44                return;
45            for (auto s = start; s < end; ++s) {
46                auto const y = smpls[s];
47                if (sample(y)) {
48                    timerIndex = 0;
49                    return;
50                }
51            }
52            timerIndex += end - start;
53            if (timerIndex < timerLength)
54                return;
55            timerIndex = 0;
56            ringing = false;
57        }
58
59        void operator()(double* smpls, int numSamples) noexcept {
60            operator()(smpls, 0, numSamples);
61        }
62
63        Sample sample;
64        int timerIndex, timerLength;
65        bool ringing;
66    };
67
68 public:

```

```

69     SleepyDetector()
70         : detectors()
71         , noteOn(false) { }
72
73     void prepare(double sampleRate) noexcept {
74         auto const timerLength = static_cast<int>(math::msToSamples(TimerLengthMs,
sampleRate));
75         for (auto& d : detectors)
76             d.prepare(timerLength);
77         noteOn = false;
78     };
79
80     void panic(String& log) {
81         log += "\nRinging: " + String(isRinging() ? "1" : "0");
82         log += "\nNoteOn: " + String(noteOn ? "1" : "0");
83     }
84
85     void triggerNoteOn() noexcept {
86         noteOn = true;
87         for (auto& d : detectors)
88             d.triggerNoteOn();
89     }
90
91     void triggerNoteOff() noexcept { noteOn = false; }
92
93     void operator()(double** samples, int numChannels, int start, int end) noexcept {
94         if (noteOn)
95             return;
96         for (auto ch = 0; ch < numChannels; ++ch) {
97             auto const smpls = samples[ch];
98             auto& detector = detectors[ch];
99             detector(smpls, start, end);
100         }
101     }
102
103     void operator()(double** samples, int numChannels, int numSamples) noexcept {
104         operator()(samples, numChannels, 0, numSamples);
105     }
106
107     bool const isRinging() const noexcept {
108         for (auto const& d : detectors)
109             if (d.ringing)
110                 return true;
111         return false;
112     }
113
114     bool const isSleepy() const noexcept { return !isRinging(); }
115
116     bool const isNoteOn() const noexcept { return noteOn; }
117
118 private:
119     std::array<Mono, 2> detectors {};
120     bool noteOn;
121 };

```