```
class WavetableSynthesizer
{
public:
    void audioCallback()
    {
        if (std::unique_lock<spin_lock> tryLock (mutex, std::try_to_lock); tryLock.owns_lock())
        {
            // Do something with wavetable
        }
        else
        {
            // Do something else as wavetable is not available
        }
    }

    void updateWavetable (/* args */)
    {
        // Create new Wavetable
        auto newWavetable = std::make_unique<Wavetable> (/* args */);

        {
            std::lock_guard<spin_lock> lock (mutex);
            std::swap (wavetable, newWavetable);
        }

        // Delete old wavetable here to lock for least time possible
    }

private:
    spin_lock mutex;
    std::unique_ptr<Wavetable> wavetable;
};
```

Basic spin_lock

```
class spin_lock
{
public:
    void lock() noexcept        { while (flag.test_and_set()); }
    void unlock() noexcept      { flag.clear(); }
    bool try_lock() noexcept    { return ! flag.test_and_set(); }

private:
    std::atomic_flag flag = ATOMIC_FLAG_INIT;
};
```