**Topic title**

Implementation of a polyphonic, MIDI-controlled, musical, wavetable synthesiser on an STM32 microcontroller

**Topic description**

The synthesiser must take a MIDI input (a communication protocol conveying note information that runs on UART) and produce a waveform based on the MIDI data. This must be polyphonic, i.e., multiple notes/voices (up to 16) can be played simultaneously.

The waveforms must be produced using wavetable synthesis. This uses stored samples of predefined waveforms, which can be blended and interpolated to form a single voice. Voices are added together to produce the output.

Voice parameters can be manipulated with user-controlled inputs (via pots or a digital interface). Parameters are and not limited to: ADSR (attack; decay; sustain; release) envelopes modulating voice volume; additional effects such as distortion; filtering that includes resonance and a cut-off frequency; wave-shaping (functions that fold and manipulate wave tables); choice of base waveform; frequency “gliding”.

It must be possible to modulate some parameters using modulators such as ADSR envelopes and LFOs (low-frequency oscillators).

The output samples must be produced by the DAC at a fixed frequency and buffered to drive a line-level output.

DSP techniques such as anti-aliasing and filtering will be required.

The device must be low latency, such that it is functional in a real-time environment.

**Why this topic is suggested**

I am a part-time musician and became an engineering student with the intent of focusing on audio processing, both digitally via DSP and through analog electronics. I am heavily invested in created a product that is musically useable and useful for my purposes through the means of combining engineering and many years of musical experience.

Commercial synthesisers are very expensive to obtain and require a lot of musical and engineering knowledge to design. This topic is therefore a golden opportunity to demonstrate what I have learnt over the past 4 years of my BEng E/E degree.

**ECSA outcome 1 (problem solving)**

Many problems are required to be solved using a variety of mathematical techniques that must also be implemented in software running on an STM32 microcontroller.

Some of the few expected problems to be solved are: converting wavetable samples to a frequency; filtering in the discreet time domain (Z-domain analysis required); design choices to limit latency; writing efficient code; robust data communication and conversion; efficient software architecture design; time-based waveform manipulation; implementing wave-folding and distortion functions; dealing with aliasing.

**ECSA outcome 2 (Application of scientific and engineering knowledge)**

As mentioned in outcome 1, a variety of mathematical tools and engineering tools and knowledge must be used in combination with musical knowledge.

This topic requires in-depth understanding of frequency synthesis, data manipulation, coding, discreet time-domain operations (such as filtering), digital communication and microcontroller operation.

**ECSA outcome 3 (Engineering design)**

A modular system must be designed and implemented on a microcontroller. Additional analog systems might also be required to work in conjunction with the microcontroller.

The design must adhere to a set of specifications and limitations (see topic descriptions) and must be musically useful.

PCB design might also be required.

Nearly all parts in this system relies heavily on design, whether it be high-level modular system design, mathematical models or software implementation.

**ECSA outcome 4 (investigations, experiments and data analysis)**

Given a certain MIDI input, an expected waveform output can be measured. Modules can also be individually measured to ensure proper functionality. This includes measuring latency, filter responses, output waveforms (frequency analysis required) etc.

Output data will need to be debugged and analysed in frequency and time domains via communication with an external measurement device (such as a computer). Measurement tools must be constructed in software to adequately measure system operation.

**ECSA outcome 5 (engineering methods, skills and tools)**

This topic will test programming and hardware knowledge and skills, along with implementing theoretical models in software.

A variety of tools are needed, which includes simulation packages such as MATLAB, IDEs such as STM32CubeIDE, UART terminals (for debug purposes) and MIDI generators.

Engineering methods such as frequency analysis and transfer function modelling will be required.