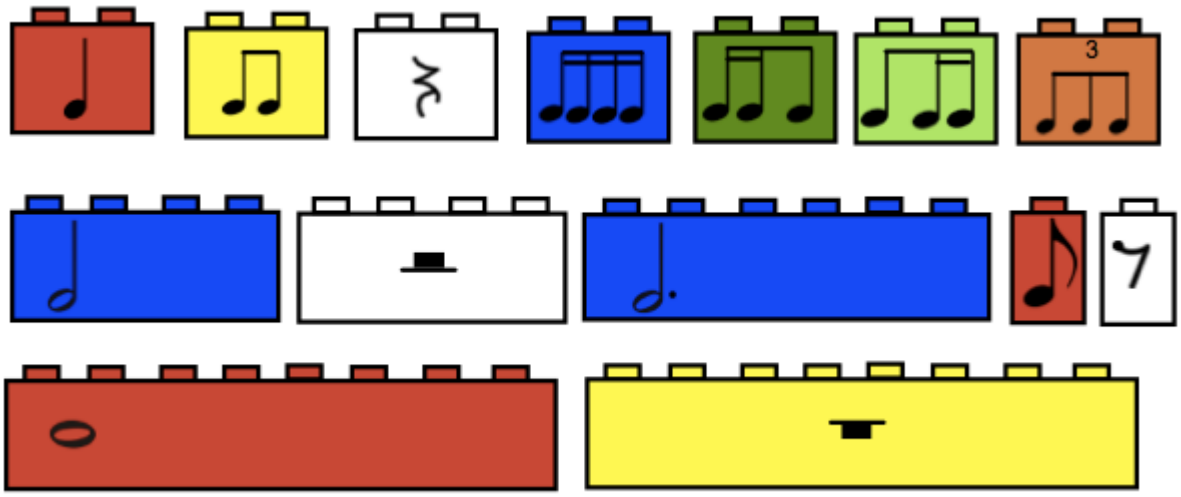
MIDIBlocks



A software and hardware defined modular synthesiser

Specification version 1.0
4 August 2015

Overview

Your team is commissioned with the design and production of one complete, working audio synthesiser system, to be known as the **MIDIBlocks** product.

The system will consist of a Windows software program which will allow the virtual construction of a synthesiser with prescribed effects using a graphical user interface. The blocks defined in this document will be connected and Musical Instrument Digital Interface (MIDI) sources will be used as inputs to the audio processing pathway your team's software has enabled in conjunction with the user's creative input.

The blocks within the software will have generic functions described in this specification, with parameters which are under the control of the user. When combined and fed with a MIDI input, the resulting output will be able to be saved to a file and passed to your **MIDIBlocks** hardware synthesiser via a USB interface.

The **MIDIBlocks** hardware synthesiser consists of modular, stand-alone circuit elements, which, when combined, allow the audio output to be heard through earbud style headphones. A power supply, processor and digital to analog converter (DAC) with mixer, voltage controlled filter and amplifier/buffer make up the basic specification modules. An optional feature of a live spectral display can be implemented if desired.

The modular concept is extended to the PCBs which will use an interconnect system which is a prescribed, matching plug and socket with defined pin allocations for power, audio-signal and digital interfacing.

The remainder of this document outlines the performance requirements in detail as well as the physical requirements and component sourcing details.

This document will change as clarification is needed and/or component availability changes. Notification to all teams of revisions will occur.

This document should be read in conjunction with the product marking criteria.

Glossary of Musical Terms

Due to the nature of this project, a minimal understanding of musical terminology is required. An explanation of some of these terms is given below. You are not expected to know any music theory in depth to succeed in this project. If you have any questions about this, please direct them to the discussion board on Blackboard.

Flat/Sharp Note: A note which contains a ‘b’ (flat) and a ‘#’ (sharp). On a piano, these would correspond to black keys, where all other notes are white.

Mode: The mode (when used to describe a scale) refers to the mathematical pattern used to derive the scale from the list of valid notes. Examples include major, minor, lydian, dorian, etc. A list of scales along with their modes can be found on Blackboard.

Note: A fundamental unit of music, which is a single tone or pitch. The only possible valid notes (for the purposes of this exercise) are A, A#/Bb, B, C, C#/Db, D, D#/Eb, E, F, F#/Gb, G, G#/Ab. A link to a website showing all of the notes and their fundamental frequencies can be found in the Hardware Specification. Notes can be also be described using a number suffix, such as A1, C4, B7, where the number refers to how many times the frequency of vibration is doubled from the lowest possible frequency. For example, C0 (the lowest possible frequency for C) corresponds to a vibration frequency of 16.35Hz. Therefore, C1 would correspond to 32.7Hz, C2 to 65.4Hz, and so on.

Octave: An octave is a series of notes where the first note has a vibration frequency of exactly half of the last note. To move a note up one octave, simply add one to the number suffix of the note. For example, C4 is one octave below C5, and A7 is two octaves above A5.

As an example, the notes C, D, E, F, G, A, B, C make up an octave which describes the notes permissible in the C major scale. Note that the root note “C” is listed both at the start and the end of the list.

Root Note: The root note (when used to describe a scale) refers to the first note of the scale. For example, the C Major scale has a root note of C and the F harmonic minor scale has a root note of F. A list of scales along with their root notes can be found on Blackboard.

Scale: An ordered collection of notes which sound “good” together, usually formed by a mathematical pattern. A list of scales can be found on Blackboard. Scales are generally described by a mode and a root note, and using this information one can derive a listing of all of the possible valid notes in any octave.

Software Specification

You must provide a piece of software that meets the following specification.

Installation and software library requirements

While the software can be developed on any machine it must run on the Windows 7 computers in the ENGG2800 lab (47-209). If your solution requires the installation of other software resources such as libraries, these should be bundled into a single installer or executable. You should include some form of installer which creates a shortcut to your application on the desktop and within the start menu. As with the firmware development, any libraries which are not shipped as standard with the language you are using will need to be approved. To get a library approved please ask a question on the discussion board using the tag **softwareapproval.** A list of approved libraries will be posted to Blackboard.

Software Specification Summary

Your software must enable the user to connect to a MIDI source, process this MIDI input through a number of configurable processing blocks and save the output to a MIDI file and also feed the resulting data to your hardware device for synthesis. Figure 1 below shows this pipeline diagrammatically. Details of the required processing blocks are provided later in this document but as an example one block may shift the notes by a configurable amount based on a parameter `shift`.

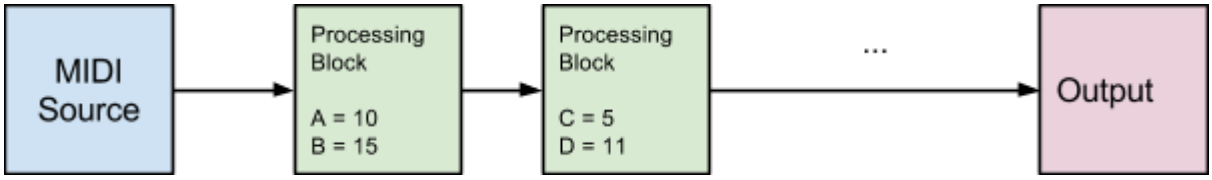


Fig. 1. PC Application processing diagram

Configurations, which include the input source as well as which processing blocks are used and their parameters should be able to be saved and loaded from files on the computer. There is no specified file format for this criteria but your file format should be documented as part of your submission process.

As users will be changing configurations often the program should keep a 15 action history of the processing chain allowing the user to undo at least 15 changes. No redo action is needed.

Detailed Criteria

MIDI Source

The MIDI source must be selectable as either keyboard input, a file containing MIDI notes which can be loaded from disk (standard MIDI file type 0 format), or the system level MIDI driver on windows. To simulate having a MIDI controller attached to your computer, you can use Virtual MIDI Piano Keyboard (VMPK) (https://sourceforge.net/projects/vmpk/). The only MIDI messages which are required to be interpreted are **Note On** and **Note Off**. If the input is the user's QWERTY keyboard then the application should display a piano key layout on the screen in which piano keys display their corresponding keyboard keys. A rough mockup of this interface is shown below (Fig. 2). The virtual piano should comprise of a series of rectangles, each of which will play a note when pressed with the mouse, or when the corresponding key is pressed on the keyboard. It should only display notes in the current scale. Any notes which contain a # (A#/Bb, C#/Db, etc) should have a black background. All

other notes should have a white background. Some form of visual feedback should indicate when the key is pressed.

The virtual piano should accommodate at least 15 consecutive notes, and users should also be able to translate the piano view along the scale. This shifting should be in one of two step sizes. Users should be able to shift one note at a time, pushing all notes one spot left or right and adding a new note as needed. Users should also be able to shift the virtual piano by one full octave. In this case the notes should remain in place but should all move up one octave. Your virtual piano must allow for notes between C1 and B7 (inclusive). Any notes that happen to be shown outside this range should be unusable and greyed out to indicate that they are unusable. This allows for the octave shift to leave the notes in place when shifting the top or bottom beyond the limits of the note range. To make it clear which is the root note of the scale you should add some sort of visual indication to the virtual piano; for example, a dot or icon. More than one root note may be visible.

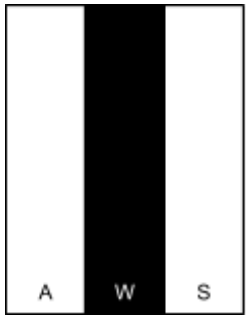


Fig. 2. Sample 'Keyboard Input' Display

Processing Blocks

Your software must allow for an unlimited number of processing blocks to be placed between the input and the output of the MIDI stream. When a new configuration is loaded there should be no processing blocks present and MIDI data should pass unchanged from the input to the output. The user of the software can then choose to add processing blocks one at a time between the input and the output. MIDI data will pass through each block in the chain before reaching the output. Input to a given block should be the output of the block before it. There will only ever be one block connected to the output of any other block (i.e there is a single, unbranched chain of blocks). Users should be able to rearrange processing blocks at any time either through a button on the blocks or by dragging to shift each block left or right along the chain. The block type (detailed below) and parameters of each block should be entered through an appropriate input item such as a text box, selection box or slider. If blocks are moved along the chain they should maintain their parameters. The user should be able to remove any processing blocks from the chain.

The software must also allow the user to set a tempo, in beats (or notes) per minute. When there is incoming input a clock should start within the software and notify all processing blocks each time a beat occurs, this allows processing blocks which rely on modifying timing to behave correctly. If the processing block is not constrained by the beat clock input should pass to the next block immediately. If the MIDI input is a MIDI file containing tempo information the tempo should be set to this value. When reading from a file the notes should be processed based on the file's timing information and not sent out as soon as they are read.

As well as the tempo, the user must be able to set a scale. This scale should be passed to all processing blocks and used to determine the notes present in the virtual keyboard. The list of available scales should come from a locally stored CSV file. The first column contains the mode of the key followed by all note starting with the root note of the key. The file will contain one full octave of

notes as well as a second root note of the next octave as the final entry. A sample csv file will be available on blackboard. The software must provide a mechanism for loading such a file and also be able to handle incorrectly formatted files.

The software must also maintain an audio and visual metronome. The metronome must change the colour of an on screen element and play a ticking noise. Every fourth tick must be a different sound (example). This metronome should be easily mutable within the application.

Processing Block Types

Pitch Shift Block

The pitch shift block shifts a MIDI note up or down by a certain number of notes. The block also constrains all notes to the globally set scale. Note that in the case of a 0 shift block, incoming notes should be shifted to the closest note in the set scale. When two notes are equidistant the note can be rounded either up or down but this behaviour should be consistent.

Parameters:
Shift - Integer - The number of notes (positive or negative) to shift the input by.

Arpeggiator

An arpeggiator block takes all of the currently ‘on’ notes and outputs a sequence of short notes made up of these notes. The output sequence should contain only single beat notes (i.e notes are switched on at the start of a beat and switched off again at the start of the next beat), the order of output is one of four definable patterns. For example if the notes A B and C are currently on, the arpeggiator will output a sequence containing these three notes. If an additional note is turned on, or any of the notes turned off, the sequence continues using the new set of notes. The following patterns should be options for the arpeggiator pattern:

- Ascending Scale - Output notes should follow a selected scale in ascending order, when the highest note is reached the next note should wrap back to the lowest note.
- Descending scale - Output notes should follow a selected scale in descending order, when the lowest note in the scale is reached the next output should wrap to the highest note in the scale.
- Ping pong - In this mode notes ascend a scale in the same way as the Ascending Scale mode, the difference in this mode is that when the highest note is reached the output switches direction and descends to the lowest note. This up and down pattern is then repeated.
- Random - Notes will be output in a random pattern, i.e., the next note to be output is chosen at random from the set of currently ‘on’ notes (and should not be the same as the currently ‘on’ note - unless there is only one note on).

Parameters:
Pattern - Selection - One of the patterns defined above.

Monophonic

The monophonic block allows only one note to be on at once. If a new “note on” signal passes through this block then any other notes should be turned off before the next note begins. As such this block needs to keep track of the last note to pass through. When new input arrives one of several things may happen. If the input is an off signal for that note it simply passes through. If the note is an “on” signal for a new note the block outputs an “off” for the old note then an “on” for the new note. If the input is an “off” signal for any other note it is simply ignored.

Chordify

The chordify block takes every input note and outputs a chord based on that note. A chord is defined as note (the note played) plus the notes 2, and 4 higher within the globally set scale. If this note is above the highest note in the scale it should wrap to the next octave (or be ignored if this is outside the valid range of notes). As an example, in the natural minor scale with a root note of A (the first line in scales.csv), an A note would output an A, a C, and an E note. The chord should play as long as the note is on. This block has no parameters.

Gate [Advanced Feature]

The gate block is responsible for ensuring the MIDI stream maintains a constant tempo. The block can be set into one of three modes which determines how new input is handled while the gate is closed. In every cycle of the global clock the gate should open to allow a configurable number of notes to pass through the gate. The three gate modes are:

- Queue - Notes are queued within the block and passed in a first-in-first out fashion
- First Hold - Only the first note should be passed on the next tick, all additional notes are ignored.
- Last Hold - Only the last (most recent) note should be passed on the next tick, earlier notes are discarded.

If the note on and note off MIDI signals are received and placed in the same queue they should cancel and be dropped from the queue. The gate should also maintain a list of outputs so that any `note on` signal can be paired with a `note off` signal. This ensures that no notes are left on indefinitely. If the gate is in one of the two hold modes and any `note off` corresponding to a previously output `note on` is seen this should be released the next time the gate is opened as well.

Parameters:

Notes Per Tick - floating point number - The number of notes to pass through on each clock tick, if this number is less than 1 the gate should release a note every (1/notes per tick) ticks, rounded up to the next integer. e.g 0.25 would release a note every 4 ticks. Numbers higher than one should be rounded to the nearest integer and released notes should be evenly distributed.

Mode - Selection - One of the three modes described above.

MIDI Output

The resulting MIDI output should be able to be sent to the user's choice of one or both of

- the USB device (if connected), and/or
- an output file.

Users should be warned if there is incoming MIDI input without a selected output.

File output must be in the standard MIDI format to allow loading at a later stage as well as verification of processing block behaviour through unit tests. We will provide a limited set of unit tests during the semester. You are welcome to create and share additional tests amongst teams so long as these files contain only tests and do not reveal your team's implementation. We make no guarantees about the correctness of any tests other than the ones officially provided by staff.

Hardware Configuration

The software must also be able to send configuration information to the hardware (when connected).

Configuration information should be sent

- whenever the software starts up (with the hardware device already connected);
- whenever the hardware device is connected whilst the software is running;
- when the lowest root note visible on the virtual piano is changed (e.g. by shifting the virtual piano view; and
- whenever the scale is changed in the software.

The software must send the first 8 notes of the configured scale (wrapping into the next octave if necessary) to the hardware device, with the first note being the lowest root note currently visible on the virtual piano. The hardware should then assign these notes to the (physical) momentary-on buttons on the synthesizer block (described below). This configuration information should be remembered when the power to the system is switched off and restored when the power is switched on.

Hardware Specification

IMPORTANT: Any example parts presented in this specification are not necessarily the best part, or even appropriate for use in this design. They are only given to provide an illustration of the purpose and form factor of the part. Your final product may only use parts available from Digikey.

MIDIBlocks Connector

The MIDIBlocks connector pinout is as follows:

+5V (pin 1)	GND (pin 7)
+5V	GND
-5V	GND
-5V	GND
I2C_SDA	I2C_SCL
AUDIO (pin 6)	GND (pin 12)

Labelled images of the connectors can be seen below. Please note that while the pinout is the same for both the input and output MIDIBlocks connector, the numbering order is reversed between them. Larger versions of the images below can be found on Blackboard.

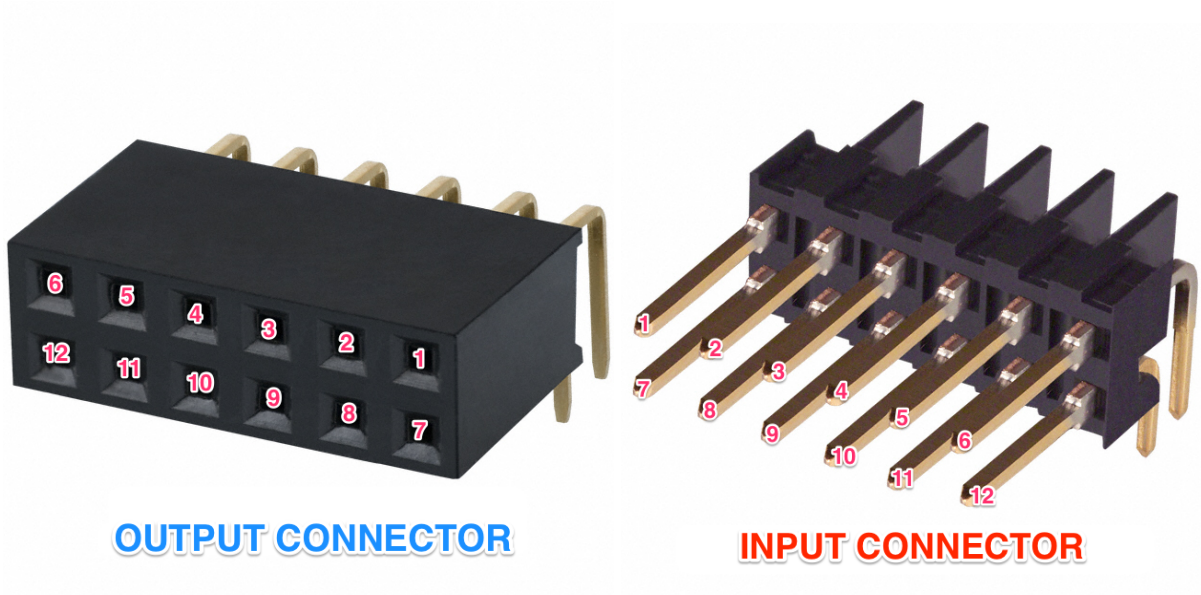


Fig. 3 Output Connector (left) and Input Connector (right)

Battery Connector

Positive voltage (Pin 1)
GND (Pin 2)
Negative voltage (Pin 3)

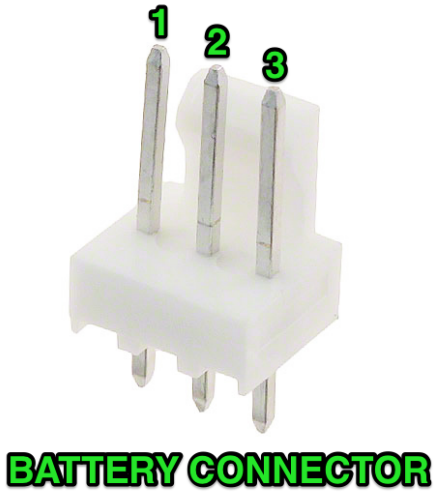


Fig. 4. Battery connector

Please see the Power Supply Block section for more details below. The connector shown is the one which is intended to be mounted on your PCB, not on the battery cable. The ETSG stock code of the connector is 01-34-01.

General block specifications

Unless specified otherwise, all **input connectors** should be placed on the left side of the board when viewed from the component side. Similarly, all **output connectors** should be placed on the right side of the board.

All boards are required to have at least 4 plastic standoffs with a thread size of M3 and corresponding standard plastic nuts. The mounting holes should be distributed toward the four corners of the board. No other stand-offs or 'feet' are to be used. All boards should be raised to the same height by the standoffs, and all components protruding on the underside of the board should be shorter than the height of the standoffs. The standoffs will be available at ETSG under the stock codes XXXXXX and XXXXXX (coming soon)

Hardware Blocks

Power Supply

Input connectors:

- Battery connector (see diagram above)

Output connectors:

- MIDIBlocks connector

Controls:

- A labelled power switch which disconnects the battery/batteries from the power supply circuitry.

Specification:

The power supply block is intended to provide the +5V and -5V voltage rails for the MIDIBlocks system. It should accept one or two battery voltages into the battery connector. You should also

provide any necessary battery packs with non-rechargeable C, D, AA or AAA size battery cells, and all conductors should all be appropriately fastened into a single row, female, KK, 3 pin connector (ETSG stock code: 01-26-01). Please note that un-keyed pin headers (example) are **not** acceptable.

If you choose to only have a single input voltage, you must leave the unused pin disconnected. For example, if you wanted to use a single 1.5V battery as your power supply, you should connect the positive terminal to pin 1, the negative terminal to pin 2, and leave pin 3 electrically unconnected to any circuits. Even if it is unconnected, it should be soldered to the PCB.

The power supply is expected to be efficient. Over 80% of the power from the battery should be supplied to the remainder of the circuit in all modes of operation (i.e. both power-saving mode and operational mode when playing notes at any volume).

Synthesiser

Input connectors:

- MIDIBlocks connector
- FTDI header (placed on the top edge of the board)
- It is expected that each team will source an appropriate USB cable which will be included in the product budget.

Output connectors:

- MIDIBlocks connector.

Controls:

- Eight momentary-on pushbutton switches (example, example)
- A multipole switch with at least 4 discrete positions (example, example). This will be stocked in ETSG as part XXXXXX [coming soon]
- A potentiometer (example, example)

Indicators:

- Three LEDs

Specification:

The synthesiser block should implement three 8-bit numerically-controlled oscillators (NCO) inside an AVR microcontroller. The numerical outputs should be sent to three separate digital to analog converters, and then merged using an audio mixing circuit and output via the AUDIO pin of the MIDIBlocks connector.

Each individual oscillator should be capable of oscillating at the standard musical frequencies for the equal tempered scale, tuned to A4=440Hz for the notes in octave 1 through 7 inclusive (a total frequency range of C1=32.70Hz to B7=3951.07Hz). A list of the exact possible frequencies and the corresponding note names can be seen at http://www.phy.mtu.edu/~suits/notefreqs.html .

The multiple-position switch is used to select which waveform is being used by all of the oscillators. The four possible waveforms (in order) are sinusoid, square, triangle and sawtooth. Labels (which could be symbols) must indicate which position is which.

The block should be able to play notes using two different mechanisms; either by using the eight buttons on the device, or by receiving messages over UART (via an FTDI USB to serial converter, ETSG stock code XXXXXX [coming soon]). The format of the UART messages is up to you, but it must take into account both note-on and note-off events. Both note-playing methods should function

simultaneously. The block must be capable of playing three different tones at once, and and if more than three notes are pressed the block shall always play the latest three notes that it has received “on” events for with no corresponding “off” event. It should only play notes while a key is pressed, or while there is a note-on message without a corresponding note-off message.

Example: If on-events are received for notes 1, 2, 3, 4 and 5 in that order then notes 3, 4 and 5 should be played. If an off-event is then received for note 2 there is no change to the notes played. If this is followed by an off event for note 4 then note 4 will be replaced by note 1 in the output.

The potentiometer position must control the cutoff frequency of the Voltage Controlled Filter block via the I2C bus in the MIDIBlocks connector.

One LED (labelled “POWER”) must indicate whether the synthesiser is powered-on or not. It should be on constantly when the system is in operational mode. It should blink briefly once every five seconds (5% duty cycle) when the system is in power-saving mode. (Modes of operation are described later in the document.)

One LED (labelled “SOUND”) should be on whenever a sound is being generated and off otherwise.

One LED (labelled “RX”) should blink whenever data is received via the USB UART connection and should be off otherwise.

A series of reference waveforms and oscilloscope configurations will be supplied via Blackboard to compare against the output waveforms of the circuit. You can load these waveforms and settings via a USB stick onto an oscilloscope in the lab, and you can then adjust the oscilloscope to try to match your output waveform with the reference waveform. After loading the settings file, the reference must “match” (see the marksheet for specific definition of this) the output of the circuit without adjusting the timebase scale (seconds per division) of the oscilloscope. Detailed instructions for performing this test will be provided shortly.

Voltage Controlled Filter (VCF)

Input connectors:

- MIDIBlocks connector

Output connectors:

- MIDIBlocks connector

Controls:

- A non-momentary switch with two positions (example).

Indicators:

- An LED

Specification:

The voltage controlled filter should implement a low pass filter which can have its cutoff adjusted by a control input voltage. The cutoff must be able to move between (at least) 100Hz and 10kHz. The filter must have a voltage gain around 1 (within the range 0.90 to 1.10) in the passband, and a rolloff of at least -12dB/octave.

The switch should allow the filter to be bypassed (ie, the audio input is connected directly to the audio output). When the filter is bypassed, the LED should be off. Otherwise the LED should be on.

Amplifier/Mixer

Input connectors:

- MIDIBlocks connector

Output connectors:

- 3.5mm stereo audio jack (example) (placed on the right edge of the board)

Controls:

- A potentiometer (example, example)

Indicators:

- An LED

Specification:
A minimum single channel 500mV output (RMS) output into 32 ohm stereo headphones is required. The same signal should be supplied to both left and right channels. The potentiometer should control the volume of the amplifier - no signal should be output at the lowest volume. Integrated circuit (IC) amplifiers are permitted as are discrete BJT or FET designs. The LED is only present to indicate if the circuit is receiving power. The amplifier should have "flat" passband gain over 20Hz - 20kHz.

The module should have low distortion characteristics. Quantitative distortion characteristics will be provided soon.

4-Band Spectrogram [ADVANCED FEATURE]

Input connectors:

- MIDIBlocks connector

Output connectors:

- MIDIBlocks connector

Indicators:

- four 10-way LED bar graphs (example) or 40 discrete LEDs arranged in 4 bar graphs

Specification:
The 4-band spectrogram block should convert four audio bands into a corresponding "volume" level for that band, and this should be displayed on the LED bar graphs. The four audio bands should be centred at 100Hz, 500Hz, 1kHz and 2kHz, and the bandpass rolloff should be at least -20dB/decade (1st order filter). Each individual LED must always be fully on or fully off; they must not be driven by an analog voltage so that they are partially on. It is expected, but not essential, that you will use a microcontroller to implement this block.

This block must not modify the audio signal - the audio input should be connected directly to the audio output.

Operating Modes and Power Requirements

The system will support two modes of operation.

1. Power saving mode. When the power switch is on and there is no USB connection to the synthesiser block and no notes are being played, the system should enter power saving mode within 10 seconds of the last note being played. In this mode, all LEDs may be switched off. (The POWER LED on the synthesiser module should behave as described above.) The power

supply must be able to keep the system (without the optional spectrogram module) in this mode for at least two weeks (336 hours).

2. Operational mode. The system is in this mode when the power supply module is connected and is supplying power to the system and either (or both)
    a. notes are being played (or have recently been played) or
    b. when there is a USB connection present.

The system should not draw power from the USB supply. If the power switch is off and USB is connected, the system must not operate.

Budget

Your target budget for the complete product is $110 (US dollars). This includes the printed circuit boards, batteries, USB cable and all components attached to all printed circuit boards - including mechanical components such as stand-offs.

Approved Suppliers

Other than PCBs, your product bill of materials must only include parts obtainable from the US Digikey (http://www.digikey.com) website. All budgeting should be done using prices on this website, in US dollars. You must budget for enough components to build your entire prototype, and you must take into account any minimum order quantities (MOQ). For example, if your product needs eleven 12kOhm resistors, but the MOQ is 20 units, your product budget must contain all 20, regardless of how many you use.

You may obtain parts from other suppliers (and may be reimbursed, subject to the reimbursement policy) but the parts used in your product must be obtainable from Digikey. For parts which do not have a visible printed manufacturer part number, you must provide evidence (e.g. purchase receipt) that the part you have used on your board is the part you list in your bill of materials. Your final submission must include the manufacturer datasheets for all parts used (electronic copies only). Some useful alternative suppliers are:

- ITEE ETSG - http://www.itee.uq.edu.au/etsg/search-stock
- http://au.element14.com
- http://australia.rs-online.com
- http://www.digikey.com.au/ (note: this has prices in Australian dollars)
- http://au.mouser.com/
- http://www.hobbyking.com/
- http://www.jaycar.com
- http://littlebirdelectronics.com/
- https://www.adafruit.com/
- http://www.seeedstudio.com/

All components must be RoHS compliant and evidence of this must be provided (e.g. if not stated in the datasheet, a manufacturer/reseller statement must be included). If necessary, for non-electrical components (i.e. parts that do not need to be shown in a circuit schematic such as standoffs and cable ties etc.), we will accept RoHS evidence for parts similar to that used, e.g., evidence for another manufacturer's part of the same composition. For ease of construction it is not required that solder be lead free. Small quantities of glue, heatshrink and wire (e.g. fly wire) may be used without being listed on the bill of materials. You may not use 3D printed parts.

Printed Circuit Boards (PCBs)

Printed circuit boards are to be costed at US$2.00 per square inch (or part thereof). PCBs may be produced using the internal ENGG/METR PCB order system (please see Blackboard for further information) or by external means. No leniency will be given for externally ordered PCBs which arrive late.

Veroboard/protoboard (if required) should be costed at the price of the minimum size board that you would need to purchase to build your circuit.

Further information will be provided on the required bill-of-materials format near the end of semester.

Software and Firmware Requirements

Your client requires that you maintain all source files (firmware, PC software, etc.) in a supplied Git repository. The team must demonstrate active use of the repository across the project duration. The team must nominate a style guide to be followed for code development. (One style guide per programming language is permitted.)

It is possible that that some functionality is easier through the use of third party software libraries. In this course the use of any library which is not included as standard with your choice of programming language will need to be approved before use. In order to request approval of a library you should ask a question on the course discussion forum using the tag *softwareapproval* (this will be a course tag in red). A list of approved software libraries will be published during the semester as the requests come in.

Your final product submission must contain

- a list of approved third-party libraries used, and
- a link to and a summary of the license under which each third-party library is used, and
- a statement about the implications of the use of each library on the commercialisation of the product (e.g. whether source code must be made available or not).

You may not use any third-party software library which would prevent the commercial sale of this product. Any software library which requires a license to be purchased before commercial use must have its cost included in the bill-of-materials. Inclusion of a software library on the list of approved software libraries for the course does not imply that the library is suitable for use in a commercial product, only that the library does not contain prohibited functionality.