DrumKid manual (V1.2)

Introduction

Welcome to the DrumKid manual! DrumKid is a musical instrument that creates rhythms using random numbers. To get started with DrumKid, unscrew the six screws on the back of the unit (you should be able to do this by hand), and carefully remove the back plate. Insert three AA batteries (you can use rechargeable or single-use), replace the back plate, and tighten the screws. Slide the power switch to "on" - you should see the lights flash briefly. After this, DrumKid is ready to use. Plug some headphones or a 3.5mm stereo aux cable into the socket (mono won't work), and start playing. If you would prefer not to use batteries, you can also power DrumKid by connecting a USB mini cable to the socket recessed on the right hand side of the device.

Philosophy

DrumKid is designed around a very basic computer chip with limited memory and processing power. My idea was to squeeze as much cool sound out of this chip as possible, but it has its limits. Rather than try and hide from these limits, I decided to embrace them. This means that it is absolutely possible to make horrible, distorted, broken noises using DrumKid. Some of these horrible noises will sound cool, others will not. DrumKid is a playable instrument like any other - experiment with it and see what happens.

Demo videos and resources

You can find demo videos and other resources at https://www.mattbradshawdesign.com/projects/drumkid, or more technical info (source code, schematics, etc) at https://github.com/mattybrad/drumkid.

Basic functions

DrumKid has six buttons and four knobs. The basic functions are as follows:

- Press the start/stop button to start or stop the rhythm
- Press buttons A/B/C/D to select different knob functions (see table below)
- Turn any of the four knobs to alter the parameters in the current selected group (see table below)
- Tap the tap tempo button repeatedly to set the tempo
- Press buttons A+B together to load a session, then choose which session to load by pressing any of the buttons (or press multiple buttons together to cancel)
- Press buttons C+D together to save a session, then choose which slot to save your

session in by pressing any of the buttons (or press multiple buttons together to cancel)
- Press buttons B+C together to change the current active memory bank, then choose a specific memory bank by pressing any of the buttons

Other functions

DrumKid has a few extra hidden functions:

- Press buttons A+B+C together to reset your beat to default values. This will not affect any saved beats.
- Press buttons A+B+D together to generate a random beat. This will also not affect any saved beats.
- Press buttons B+C+D together to enter the MIDI settings menu (more details in the MIDI section of the manual)

Presets

In the aleatoric spirit of DrumKid, all 36 sessions stored when you first use the instrument are randomly generated when DrumKid is built. Try loading a few of them to get a feel for what DrumKid can do.

Parameters

There are 16 different parameters which can be adjusted on DrumKid, split into four groups, with each group's four parameters controlled by the four knobs. The parameters are grouped as follows:

Group A (randomness)

1. Chance
2. Zoom
3. Range
4. Midpoint

Group B (effects)

1. Pitch
2. Crush
3. Crop
4. Drop

Group C (drone)

1. Drone
2. Modulate
3. Tuning
4. Note

Group D (rhythm)

1. Beat
2. Beats/bar (time signature)
3. Swing
4. Tempo

You can try out each parameter by starting a rhythm (using the start/stop button), then selecting a group and turning the different knobs. For example, try selecting group B (by pressing the B button) then turning the first knob, which will now control "pitch". You should hear the sound change. Below are descriptions of exactly what each parameter does, and how it can be used.

Chance - The probability of extra drum hits being generated. At zero, the beat will be completely unchanged, while for higher values there will be lots of extra events, usually creating a busier, messier beat. This can be used in conjunction with zoom, midpoint, and range to create drum fills that transition organically from a simple beat.

Zoom - Chooses which subdivision of the beat should be affected by the randomly generated extra hits. When zoom is at zero, beats will only be generated for "whole notes" (i.e. the first beat of the bar). Higher values will affect half notes, quarter notes, eighth notes, and so on. A medium value is a good place to start for zoom, but turning it up can make a beat or fill sound more "urgent". Some of the smaller subdivisions will vary depending on the current "swing" setting.

Range - This is the range of velocities which will be assigned to the random hits, and works in conjunction with "midpoint" (see below). For all the random hits to have the same velocity as each other, keep this value at zero. For a wide range of velocities, turn this up high.

Midpoint - This is the average velocity of the randomly generated hits, and works in conjunction with "range". Note that this control does allow negative values (below halfway). If you set the midpoint to maximum and have range set at zero, all the randomly generated hits will be at max velocity. Conversely, if you set it at zero, some hits will be muted. Setting it somewhere in the middle (with non-zero range) will add some random hits while reducing the volume of others. Try keeping midpoint at a three-quarters setting to add a bustling undercurrent to your beat, or turn it up briefly to create a drum fill.

Pitch - Alters the playback speed of the samples. Will play samples backwards if you turn the knob below halfway.

Crush - Reduces the number of bits used to calculate the audio output, creating a digital distortion effect. Higher values are clean, lower values are more distorted (fewer bits).

Crop - Crops the end of the samples, creating a staccato effect. Lower values are more cropped.

Drop - This control mutes some or all of the drum channels, allowing you to quickly "drop" everything except the hi-hat and snare, for example, or only retain the kick drum. Broadly, this control has "treble-y" channels at one end and "bass-y" channels at the other. The setting corresponding to all the channels being audible is somewhere in the middle.

Drone - DrumKid generates a drone which can be mixed with the drum sounds. The "drone" parameter controls the amount of raw drone signal added to the mix. Turning to the left adds a single tone, while turning to the right adds two tones, a fifth apart. A halfway value mutes the drone.

Modulation - The drone can also used to modulate the audio signal from the drums, creating a robotic effect. When "modulation" is set at halfway, no effect is heard. Turning the knob to the left modulates the drums with a single drone, while turning it to the right modulates the drums with two tones, a fifth apart. Turning the knob all the way left or right creates an extreme effect, while values closer to halfway will be more subtle.

Tuning - Controls the overall tuning of the drones (both the single and fifth tones).

Note - Alter the root note of the drones, in semitone steps over an octave range.

Beat - Chooses between a series of basic underlying drum patterns (see "preset beat list" below).

Beats/bar - Alters the time signature of the beat, i.e. the number of steps in the pattern. You can have between 1 and 13 beats in a bar, or if you turn the knob all the way to the right, the time signature will be randomised at the start of each bar.

Swing - Creates a swing feel to a beat by altering the timing of certain hits. There are three settings: straight, partial swing, and full (triplet) swing.

Tempo - Alters the tempo (BPM) of the beat. Will override any tempo previously set using the tap tempo function. Has a range of 10BPM to ~1000BPM, with the "sensible" tempos grouped in the middle of the range.

MIDI

DrumKid has MIDI in and out ports. These can be used to connect DrumKid to other MIDI equipment.

Synchronisation

DrumKid sends and receives MIDI clock signals. A clock signal will be sent through the "MIDI out" socket whenever DrumKid is playing. If DrumKid detects a clock signal through the "MIDI in" socket, it will synchronise itself to this signal. Once a clock signal has been received, the tempo controls will stop having any effect until a clock "stop" command is received.

Note output

By default, DrumKid outputs note data on channel 10 (the standard MIDI drum channel). The default note numbers are as follows:

- Kick - C1 (36)
- Click - C#1 (37)
- Snare - D1 (38)
- Closed hi-hat - F#1 (42)
- Tom - G1 (43)

CC/note input

DrumKid responds to MIDI CC (control change) messages on any channel. DrumKid responds to CC numbers between 16 and 31, in order, as follows:

- Chance (CC 16)
- Zoom (CC 17)
- Range (CC 18)
- Midpoint (CC 19)
- Pitch (CC 20)
- Crush (CC 21)
- Crop (CC 22)
- Drop (CC 23)
- Drone (CC 24)
- Modulate (CC 25)
- Tuning (CC 26)
- Note (CC 27) (almost certainly better to just control this with regular note commands - see below)
- Beat (CC 28)
- Beats/bar (time signature) (CC 29)
- Swing (CC 30)
- Tempo (CC 31) (disabled when using sync)

You can also control the drone's base note using MIDI note commands on any channel.

MIDI settings menu

If you press buttons B+C+D together, you can access the MIDI settings menu, which allows you to change the note and MIDI channel for each of the five drum channels used by DrumKid. Once you enter the menu, you can:

- Press any of the first five buttons (start/stop, A, B, C or D) to choose one of the five drum channels, or press the tap tempo button to exit the settings menu
- Once you have selected a drum, you can edit it as follows:
- Use the first two buttons (start/stop and A) to increase or decrease the MIDI note number
- Use the middle two buttons (B and C) to increase of decrease the MIDI channel
- Use button D to test the current note number and channel (i.e. send a MIDI note command with the new settings)
- Use the tap tempo button to confirm your selection and save it to DrumKid's memory. You will then have to enter the settings menu again using B+C+D if you want to change another drum channel
- At any time when in the MIDI settings menu, you can press all six buttons together to reset all MIDI settings to the defaults (see "note output" section above) - this will also exit the settings menu

Other info

Because DrumKid uses a lo-fi method (known as pulse width modulation) to generate its audio signal, there is a trace, high-frequency noise present in the output. While this signal should be above the human range of hearing, you may want to filter it out if you are recording DrumKid in a studio, especially if you intend to pitch-shift the recording downwards (since this could bring the noise within human hearing range). To remove the noise, use a band-stop or low-pass filter (before any pitch-shifting) - the offending frequency should be at 32768Hz.

Preset beat list (X denotes LED)

1. You can leave your hat on (◯●●●●)
2. Johnny Two-Hats (-X---)
3. Drum lesson (--X--)
4. Half-time rock (---X-)
5. There there (----X)
6. Metal (X----)
7. Four to the floor (-X---)
8. "Dance" (--X--)
9. Blue Monday (---X-)
10. Chime (----X)
11. Funk Soul Brother (X----)
12. Amen I (-X---)
13. Amen II (--X--)

14. Superstition (---X-)
15. Derribar el muro (----X)
16. Hips don't lie (X----)
17. Videotape (-X---)
18. Videotape (Bonnaroo) (--X--)
19. Military (---X-)
20. Ballad (----X)
21. Waltz (X----)
22. Take Five (-X---)
23. Unsquare (--X--)
24. Nihil (---X-)

Firmware versions

This manual refers to firmware version 1.2, which is now available and is included with newer DrumKids. Please note that updating to the new firmware from version 1.0 may alter some settings in your saved beats, particularly the time signature, tempo, and "drop" settings.

Updating firmware

DrumKid's "brain" is a tiny computer called an Arduino Nano clone - this is the board inside DrumKid with a mini USB port on it. To upload new firmware, you will need a computer running the latest Arduino IDE software (download from https://www.arduino.cc/) and a USB "mini-B" cable to connect the computer to the Arduino Nano.

Before you upload anything to the Arduino, unplug any MIDI cables and make sure the switch on the underside of the PCB is set to "Arduino" rather than "MIDI" (and remember to switch it back to "MIDI" when you're done). You should also check that the "board", "processor" and "port" settings are correct in the Arduino software. Click on the "Tools" menu to check this:

- "Board" should be "Arduino Nano"
- "Processor" should be "ATmega328P (Old Bootloader)" - please note that it needs to be the old bootloader specifically!
- "Port" will depend on your setup - try changing this if you're having trouble

To test that you're able to upload code, try opening the "Blink" example sketch (File > Examples > Basics > Blink) then uploading it to DrumKid (Sketch > Upload). If successful, one of the LEDs should start flashing on and off. Next, try uploading the latest DrumKid code from https://github.com/mattybrad/drumkid - start by downloading the whole project (repository) as a ZIP file, then unzip it to a folder on your computer, open the arduino/1.2/drumkid/drumkid.ino file (or a later version if it exists), and upload this sketch to the Arduino.

If you encounter any problems uploading code to the Arduino, check the following:

- The switch on the underside of the PCB should be set to "Arduino"
- Try (carefully) removing the Arduino Nano from DrumKid and uploading the "Blink" example sketch
- Try a different USB cable
- Double-check the "board" and "processor" settings (see above)
- Try a different port

Using your own samples

You can upload your own samples using the online tool on the DrumKid web page: https://mattbradshawdesign.com/projects/drumkid/ - once you have downloaded the samples, copy them into your firmware folder and re-upload the firmware using the Arduino software. If you want to restore the default samples, you can either generate a new set of default samples using the web tool, or you can find a backup of them in the "originalsamples" folder.

Hacking DrumKid (this section is a work in progress!)

DrumKid is an open source project, based on the Arduino Nano, and is designed in such a way that it can be modified and repaired. The source files for DrumKid are available from https://github.com/mattybrad/drumkid/ - you will find the schematics, CAD files, parts list, source code and more. The rest of this manual is aimed at advanced users of DrumKid who are already familiar with the instrument's basic features and would like to customise DrumKid (or for anyone who is just interested in how DrumKid works). Most ideas described in this part of the manual will require some skills in programming and/or electronics, but don't let that put you off - even if you don't currently have those skills, this might be a good way to learn!

Disclaimer

The remainder of this manual will be a little less structured and a little more stream-of-consciousness. Also, I'll be referring to various files found in the GitHub repo, which is an ever-changing resource. If you can't find a particular file in the exact location specified, or if something is a bit different to what is described here, it probably just means that I've reorganised the repo or made some improvements to one of the files. You may occasionally need to use your imagination and/or ingenuity.

What's possible?

Here are a few examples of things that you could do with DrumKid (beyond what's described in the vanilla manual above):

- Add your own preset beats
- Change the drone waveform
- Replace one of the parameters with an effect that you program yourself (e.g. delay or

chorus)
- Add a light sensor to control certain parameters
- Add Eurorack CV control
- Make your own case in your favourite colour
- Mount DrumKid to a pedalboard or similar
- Build an enclosed wooden case for DrumKid
- 3D-print a custom case

What's inside DrumKid?

The front panel of DrumKid is a printed circuit board (PCB), with some components mounted on top (buttons, knobs, LEDs, power switch, headphone output) and some hidden underneath. The stuff underneath is what we're mainly concerned with in this section of the manual. Besides the circuit board and electronic components, DrumKid also contains laser-cut plastic parts and metal stand-offs, which form a rudimentary (open-sided) case.

If you're only modifying DrumKid's software/code, you won't even need to do any disassembly, but it's useful to see and understand what you're modifying before you get stuck in. Unscrew the six thumb-screws on the back of DrumKid, and remove the rear cover (as if you were replacing the batteries). You should now be able to see all the circuitry, including the Arduino Nano board which controls everything.

The board inside DrumKid is actually technically not an official Arduino Nano, but an unofficial clone, so I will just call it a Nano from this point onwards. "Arduino" refers to a specific brand of boards, although the design is open source, which is how unofficial clones are able to exist legally. It was too costly to put an official Arduino board inside DrumKid, but I would heartily recommend buying an official Arduino to play with if you haven't used one before, or donating to the Arduino project.

Anyway, the Nano is a microcontroller - basically a small, low-powered computer. This computer runs a single "sketch", which is the name for a piece of software running on an Arduino (or compatible) board. This sketch monitors DrumKid's buttons and knobs, controls its LEDs, sends and receives data via the MIDI ports, and (most importantly) generates an audio signal.

Each of the Nano's pins (the metal bits along the edges) corresponds to a different signal either going into or out of the board, and there is a spare set of "breakout" holes that you can use to solder extra components which will connect to any of the pins. To access the other side of the board for soldering, simply unscrew the four machine screws on the front of DrumKid which hold the plastic cover in place.

Hacking DrumKid's code

The basic process for hacking DrumKid's code starts the same way as installing a firmware update, so you should start by reading the "Updating Firmware" section above. You will see that I suggest uploading the "Blink" example sketch to check that the upload process is working properly. The next stage towards hacking DrumKid is to try editing that sketch. Try making the LED blink faster or slower, or try lighting a different LED. Have a look at some other example sketches. Maybe try and make certain LEDs light up when you press certain buttons. Below is a list of what each of the Nano's pins is used for in the circuit, to help you get started.

Nano pins

The Nano board has 14 digital pins (which can either send or receive binary on/off signals) and 6 analogue pins (which can read a range of voltages between 0V and 5V). In DrumKid's circuit, the analogue pins are used to read the potentiometers, and the digital pins are used for everything else. Certain digital pins can also output a special kind of signal (PWM - pulse width modulation) which is useful for generating audio signals. Here is the list of pins used (and not used) by DrumKid:
D0 - MIDI input
D1 - MIDI output
D2 - LED 1
D3 - LED 2
D4 - Start/stop button
D5 - Button A
D6 - Button B
D7 - Button C
D8 - Button D
D9 - Audio output
D10 - Tap tempo button
D11 - LED 3
D12 - LED 4
D13 - LED 5
A0 - Potentiometer 1
A1 - Potentiometer 2
A2 - Potentiometer 3
A3 - Potentiometer 4
Please note there is also an LED (the left one) permanently connected to the 5V power supply - this is just to show you that DrumKid is receiving power, and cannot be controlled by the Nano.

How DrumKid makes sound

To generate audio, DrumKid makes use of a library called Mozzi, which you can download from https://sensorium.github.io/Mozzi/

Mozzi makes efficient use of a microcontroller's limited resources, and is one of the few ways I have found of allowing an Arduino-style board to generate interesting sounds (beyond just square-wave bleeps). I would recommend trying the Mozzi example sketches once you've installed the library. Start with the basic sine wave example and check that you can hear a sound through the output, then try editing Mozzi sketches or writing your own to get familiar with how the library works.

A simple first "hack" - editing the default rhythms

Once you're happy using the Arduino software to edit sketches and upload them to DrumKid, you can start editing the DrumKid source code. A good first task is to replace some of the default rhythms. Open the drumkid.ino file in your firmware folder, and you should see various tabs at the top of the window, representing other files besides the main code. Open the tab marked "beats.h".

You should see a lot of zeros and ones, organised into lines and blocks. Each line corresponds to a channel (kick, hi-hat, snare, rim, or tom, although obviously you can choose any samples you want). A line contains 32 ones or zeros. If you changed every 0 on the first line into a 1, the kick drum would be triggered 32 times over two bars. Your ones and zeros must be organised into groups of eight, like they are to start with (each group of eight is called a "byte" - computers like things to be in well-organised bytes). Each byte begins with the letter B, so that the software knows what to expect.

Try resetting an entire beat to just zeros, but then add a one at the start of the very first line (so the first byte is B10000000). Upload the code, use the "beat" control to find the beat you have created, and you should hear a very boring rhythm consisting of just one kick drum at the start of the bar.

Now make some more interesting beats. Remember you can always re-download the original beats from GitHub if you want them (or just make a copy of the beats.h file before you start editing).

Interfacing DrumKid with other gear (Eurorack etc)

By default, DrumKid has an audio output, a MIDI output, and a MIDI input. This is already enough to connectivity to allow DrumKid to talk to a computer, and you may find that this is enough functionality for you, particularly if you are adept at using your DAW to re-map incoming MIDI signals. However, if you want to make DrumKid trigger a modular synth without using MIDI, for example... well, this topic is going to need some more in-depth thought and experimentation from me so I'm going to stop here for now. I did say it was a work in progress. More coming soon!