

# Gizmo

## An Arduino MIDI Tool

By Sean Luke [sean@cs.gmu.edu](mailto:sean@cs.gmu.edu)

Gizmo is a small MIDI device intended to be inserted between the MIDI OUT of one device and the MIDI IN of one or more other devices. There are a number of tools like this, such as the esteemed MIDIPal<sup>1</sup> or the variety of tools from MIDISolutions<sup>2</sup>

Gizmo is the unholy union of an Arduino Uno or Mega 2560,<sup>3</sup> an Adafruit 16x8 I2C LED matrix<sup>4</sup>, and a SparkFun MIDI Shield<sup>5</sup> to perform a variety of helpful MIDI tasks. Gizmo has several built-in applications, and assuming there's space<sup>6</sup> you can develop additional applications for it.

This is a software project: the hardware is just three off-the-shelf boards and four wires. There's no enclosure, no custom board. If you'd like to do that, I'd welcome seeing it.

Gizmo comes built-in with the following applications and other capabilities:

### Applications

- **Arpeggiator.** Gizmo's arpeggiator has built-in up, down, up/down, repeated chord, and random arpeggios spanning one to three octaves. Additionally, you can define up to ten additional arpeggiator patterns, each up to 32 notes long, involving up to 15 different chord notes, plus rests. Arpeggios can be *latched* (or not), meaning that they may continue to play even after you have released the keys. You can specify the note value relative to the tempo (ranging from eighth triplets to double whole notes), the note length as a percentage of the note value (how legato or staccato a note is), the output MIDI channel, and the degree of swing (syncopation). Gizmo will show the arpeggio on-screen.
- **Step Sequencer.** Gizmo's step sequencer can be organized as 12 16-note tracks, 8 24-note tracks, or 6 32-note tracks. Each note in a track can have its own unique pitch and velocity, or be a rest, or continue (lengthen, tie) the previous note. You can also fix the velocity for an entire track, mute tracks, fade their volume, specify their independent output MIDI channels, specify the note value relative to the tempo (again ranging from eighth triplets to double whole notes), the note length as a percentage of the note value (how legato or staccato a note is), and the degree of swing (syncopation). The step sequencer lets you edit in two modes: either by triggering independent steps like a classic drum sequencer, or by playing a sequence of notes. The Arduino Uno can store up to two sequences in its slots (shared with the Recorder, discussed next). The Mega can store up to nine sequences. Gizmo will show and edit sequences on-screen.
- **Recorder.** The Arduino doesn't have a lot of memory, but we provide a note recorder which records and plays back up to 64 notes (pitch and velocity) played over 21 measures. The recorder has 16-voice polyphony. It's enough to record a very short ditty. You can also set the recorder to loop the recording while playing, and to provide a click track. The Arduino Uno can store up to two recordings in its slots (shared with the Step Sequencer). The Mega can store up to nine recordings.
- **MIDI Gauge.** The gauge will display all incoming MIDI information on one or all channels. Note on, note off, and polyphonic aftertouch are shown with pitch and velocity (or pressure). Channel

---

<sup>1</sup>MIDIPal is quite similar to Gizmo in functionality and goals, but I wasn't even aware of the existence of MIDIPal when I started building Gizmo. And besides, re-inventing the wheel is fun! MIDIPal's page is <http://mutable-instruments.net/midipal>. This device has been discontinued, but its open source software lives on in other hardware clones, such as the MIDIGal (<https://midisizer.com/midigal/>)

<sup>2</sup><http://www.midisolutions.com/>

<sup>3</sup><http://arduino.cc>

<sup>4</sup><https://www.adafruit.com/categories/326> I use the 1.2-inch 16x8 matrix with red round LEDs, which works well. This particular model can be found at <https://www.adafruit.com/products/2037>

<sup>5</sup><https://www.sparkfun.com/products/12898>

<sup>6</sup>On the Uno there's basically no extra space: we're close to the limit. So if you want to do a big application you really should use a Mega.

aftertouch is shown with the appropriate pitch. Program changes indicate the number. Pitch bends indicate the value in full 14-bit. Channel control, NRPN, and RPN messages display the number, value, and whether the value is being set, incremented, or decremented. Sysex, song position, song select, tune request, start, continue, stop, and system reset are simply noted. Rapid-fire MIDI signals such as MIDI clock, active sensing, and time code just turn on individual LEDs.

- **MIDI Control Surface.** Gizmo has two potentiometers and two buttons. Each can be configured to send a unique Program Change, Control Change, NRPN, or RPN commands, or optionally to output voltage values via either of two DACs. The potentiometers can only send MSB values.

**Other Cool Stuff** As if this weren't enough, Gizmo also comes with the following capabilities. All of these settings are automatically saved permanently in memory.

- **Tempo and the MIDI Clock.** Gizmo can sync to an external MIDI clock, can ignore it and use its own internal clock, and can emit the same as a MIDI clock. Gizmo can also pass the external MIDI clock through or block it. When using its own internal clock, Gizmo supports tempos ranging from 1 to 1024 BPM. Gizmo applications can also be set with a *note value* (or *note speed*) to play their notes relative to this clock, ranging from eighth triplets (1/24 beat) to double whole notes (8 beats). At any time, Gizmo pulses on-screen LEDs indicating the clock and note speed pulses. Finally, Gizmo can be configured to vary the degree of *swing* or *syncopation*.
- **In and Out MIDI Channels** Gizmo can be configured with input and output MIDI channels (some applications will use these: others will have multiple channels and will treat these as defaults). Both channels can be turned off, and the input channel can be OMNI. Gizmo pulses on-board LEDs indicating incoming and outgoing MIDI data.
- **Remote Control via MIDI NRPN.** Gizmo can be controlled remotely via NRPN rather than using its on-board controls. If you start using the controls, Gizmo's onboard potentiometers are locked out so their noise will not interfere with your remote control. You can unlock the potentiometers by pressing a button on-board Gizmo, or by sending Gizmo's *Release* NRPN message.
- **Transposition and Volume Control** You transpose all of Gizmo's MIDI output pitch by anywhere from -60...60 steps. Additionally you can multiply the MIDI output velocity (volume) by multiples of two ranging from 1/8 to 8.
- **Bypass.** You can quickly put Gizmo in Bypass mode, to pass through all MIDI signals and generate no new ones.
- **Screen Brightness and Menu Delay** You can change the screen brightness. Also, many of Gizmo's menus display the first few letters of a menu item, then pause for some interval of time before they start scrolling the full item. You can change this interval.

**Future Stuff (on the Arduino Mega)** I have written but not tested some additional code. It won't appear on the Uno version (not enough space) but may show up on the Mega version. This stuff includes:

- **Control Voltage** You can configure Gizmo to optionally output voltage values from 0-5V, through two DACs, in response to notes being played. One voltage output is in response to the note pitch, and the other is in response to the note velocity. This would allow you to control non-MIDI devices which respond instead to voltage.
- **Per-MIDI Device Options** There will be an area for specific kinds of MIDI devices. For example, I have written code to convert NRPN values into the unusual sysex messages required by the Kawai K4 Synthesizer so it can be manipulated easily via a standard MIDI controller.

- **Keyboard Splitting** You can split incoming MIDI notes by pitch and route them to different MIDI out channels.

Hopefully there'll be more than this. Message filtering? A MIDIPal-style CC LFO? Longer note recorder songs? And so on.

## 1 Building Gizmo

1. **Assemble the Hardware** Gizmo consists of an Arduino Uno or Arduino Mega, a SparkFun MIDI Shield<sup>7</sup> attached via four wires (I2C) to an Adafruit 16x8 LED Matrix.<sup>8</sup> The SparkFun MIDI Shield does not come with headers to plug into the Arduino, and you'll need to get those too: it's assumed you'll either use plain headers, or use stackable headers. I use stackable headers as it makes it easy for me to plug my four test wires into it.

Attach the four wires for SDA, SCL, 5V, and GND between the MIDI Shield and the LED Matrix.

2. **Modify your Arduino Software** You will need a pretty recent version of the Arduino software, as newer versions have better compilers which compile more compact code. I developed Gizmo on Arduino 1.6.12 for MacOS X. You will want to modify the source code files for the Arduino libraries in two spots:

**Add Nonblocking I2C Writes** This will dramatically speed up I2C for our purposes.

- (a) Locate your `Wire.cpp` and `Wire.h` files. On the Mac they're located in `Arduino.app/Contents/Java/hardware/arduino/avr/libraries/Wire/src/`
- (b) Add the following line inside the "public" method region in `Wire.h`:

```
uint8_t endTransmissionNonblocking();
```

- (c) Add the following method to `Wire.cpp`:

```
uint8_t TwoWire::endTransmissionNonblocking()
{
    uint8_t ret = twi_writeTo(txAddress, txBuffer, txBufferLength, 0, 1);
    return ret;
}
```

**Reduce the I2C Buffer Size from 32 to 20** This gives us some extra static RAM space.

- (a) Locate your `Wire.h` file. Again, on the Mac it's located in `Arduino.app/Contents/Java/hardware/arduino/avr/libraries/Wire/src/`
- (b) Change the `BUFFER_LENGTH` constant in the `Wire.h` file as follows:

```
#define BUFFER_LENGTH 20    // Was 32
```

- (c) Identify your `twi.h` file. On the Mac it's located in `Arduino.app/Contents/Java/hardware/arduino/avr/libraries/Wire/src/utility/`
- (d) Change the `TWIBUFFER_LENGTH` constant in the `twi.h` file as follows:

```
#define TWI_BUFFER_LENGTH 20    // Was 32
```

<sup>7</sup><https://www.sparkfun.com/products/12898>

<sup>8</sup>This matrix comes in various colors LED shapes (round, square), and matrix sizes (0.8 inch, 1.2 inch). I personally use a 1.2 inch round red matrix. The URL for my model is <https://www.adafruit.com/products/2037>

3. **Install the FortySevenEffects MIDI Library** But not straight off of its website.<sup>9</sup> The MIDI library is in a state of flux and is having some significant additions being made. The Gizmo code is extremely sensitive to code size (at least on the Uno). Thus for now, use the version I include in Gizmo's Github repository.
  4. **Switch the MIDI Shield to "Prog"**
  5. **Build and Upload Gizmo's Code to the Arduino** It should compile cleanly for either an Arduino Uno or an Arduino Mega 2560. On the Uno it'll complain of Low Memory.
  6. **Switch the MIDI Shield to "Run"** You can plug in MIDI cables now.
  7. **Reset Gizmo** Power up the Arduino. Then hold down all three pushbuttons on the MIDI Shield. While doing so, press and release the reset button on the MIDI Shield. Continue to hold down the three push buttons until the two LEDs on the MIDI Shield light up. Let go of the pushbuttons.
- This procedure initializes the memory, options, and files in the device. Now you're ready to go!

## 2 User Interface

Gizmo has a modest interface: a 16x8 LED display, two additional on-board LEDs (red and green), three buttons, and two potentiometers (dials): but it strives hard to make good use of these.

Gizmo's top-level interface layout is a menu hierarchy. You select menu items, which may take you deeper into the hierarchy, or you can go back up towards the top of the hierarchy. The top level menu chooses between different **applications**. Once you have entered an application, its **application number** is displayed in the **current application region** (see Figure ??).

The current application LEDs light up right-to-left in a certain pattern with increasing application numbers, as shown in the figure at right. Gizmo's interface can presently support up to eleven top-level applications.

Also shown in Figure ?? are the **Beat/Bypass** light and the **Note Pulse** light. The Beat/Bypass turns on and off every *beat* (quarter note). This shows the current **tempo**.<sup>10</sup> The Note Pulse light turns on and off every *note pulse*: this is the speed you have chosen for the arpeggiator, step sequencer, etc. Note pulses are defined in terms of note value: for example, if the note pulse is presently in sixteenth notes, then this light will turn on and off four times faster than the beat light, which is in quarter notes. The note pulse light is also affected by the degree of swing (syncopation) you have defined.

The reset of the screen is given over to the application. In many cases an application will display text and other data in the **Application Display** region, while lighting up certain status LEDs in the **Application Footer**.

You wend your way through menus, choose applications, and manipulate them using Gizmo's three buttons and two potentiometers as follows:

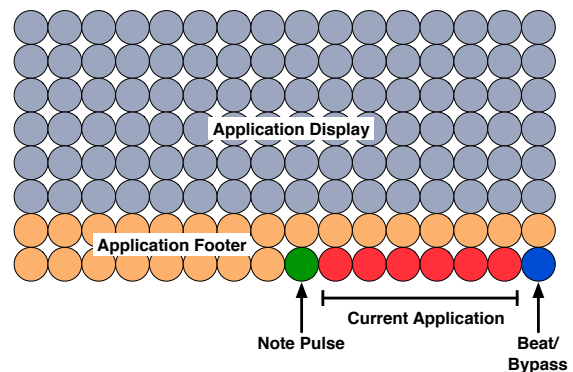


Figure 1: High-Level Gizmo Layout

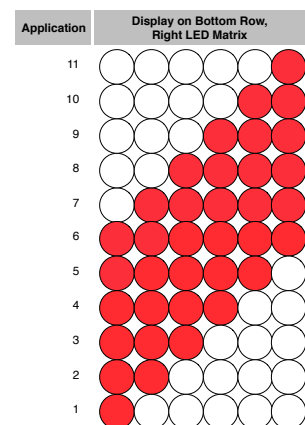


Figure 2: Application Values

<sup>9</sup><https://github.com/FortySevenEffects/arduino.midi.library>

<sup>10</sup>A beat occurs every 24 *pulses* of a MIDI Clock or Gizmo's internal clock.

**Buttons** The left button is called the **back button**. The right button is called the **select button**. The center button is called (unsurprisingly) the **middle button**. The buttons respond to being **pressed** and immediately released, and also respond to **long presses**: holding a button down for a long time (presently one-half second) and then releasing it.

Buttons do different things. Here are certain common items:

- **Pressing the Back Button** This is the “escape” or “cancel” button: it **always goes back up the menu hierarchy**. Thus you can’t ever get lost — just keep pressing the Back button and you’ll eventually find yourself at the root menu.
- **Long-Pressing the Back Button** This toggles **bypass mode**, where Gizmo tries hard to act as if it’s just a coupler between its input and output MIDI wires. In bypass mode, the **Beat/Bypass** light (Figure ??) will stop beating and instead will flash rapidly.
- **Pressing the Select Button** If you’re being presented with a menu of options, or a number to choose, or a glyph to choose, the Select button will select the currently-displayed option. In some applications (such as the Step Sequencer, Arpeggiator, and Recorder) the Select Button is assigned other tasks.

**Potentiometers** Gizmo’s potentiometers (or **pots**) do a number of tasks, including scrolling through menu options, moving a cursor horizontally or vertically, or choosing a number. The left pot does primary tasks, and the right pot assists when appropriate. Note that these are potentiometers, not encoders: if you start turning one, you may experience a big jump initially. The potentiometers have a limited range (0...1023), and a significant degree of noise, so realistically they can only choose numbers between 0...256 or so. If asked to select a *big* number, the **left pot** will act as a coarse-level selector, and the **right pot** will fine-tune the number.

**On-Board LEDs** There are two LEDs on-board the SparkFun MIDI Shield. The **red LED** is toggled on and off to reflect incoming MIDI data. The **green LED** is toggled on and off to reflect outgoing MIDI data.

### 3 Initializing Gizmo

Gizmo must be initialized before it can be used. You can also do a factory reset on Gizmo with this procedure:

1. Hold down all three buttons
2. Press the reset button
3. Continue to hold down all three buttons until both on-board LEDs have lit.
4. Let go of the buttons.

Gizmo will reset all of its flash RAM and reboot. This means that all options will be set to their default state, and all file storage (including arpeggios) will be emptied.

### 4 Starting Gizmo

If you power up Gizmo, or press its reset button, it will display its **boot screen**, which includes the version number (at right, the version number is “1”).

Gizmo will then show the **root menu**. Use the left potentiometer to scroll to any of the following applications, and use the select button to select an application. Available applications are:<sup>11</sup>

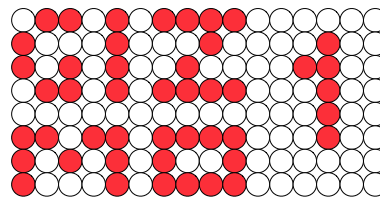


Figure 3: Boot Screen

<sup>11</sup>Later I’ll be adding a **Utility** application, where small MIDI utilities will be available.

## 5 The Arpeggiator

The arpeggiator allows you to play different kinds of arpeggios, and also to create and save up to ten of them. The arpeggiator's main menu lets you choose to play five different kinds of pre-set arpeggios, or to select an arpeggio from user-created arpeggio slots 1–10, or finally to create an arpeggio.

An arpeggio is a pattern for repeatedly playing the notes in a chord. When you hold down a chord on the keyboard, the arpeggiator uses this pattern to play the notes, typically one at a time, according to the pattern selected. Patterns do not specify exact notes, but rather note orderings. For example, an arpeggio might be defined as 1, 2, 2, 3, 4, 2, 3: if 1 is set to be the root (lowest) note of your chord, then if you play a C7 chord (C E G B $\flat$ ), then the arpeggiator will repeat the sequence C E E G B $\flat$  E G. This pattern will repeat as long as you hold down the keys in the chord.

Alternatively if you toggle the **latch**, then the playing will continue after you have released the keys, and will only shift to a new chord when you hold down a new chord. Latch mode is toggled by pressing the Middle button while playing an arpeggio. When Latch mode is engaged, an LED will light as shown in Figure ??.

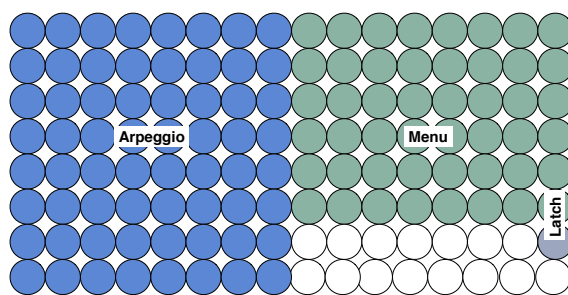


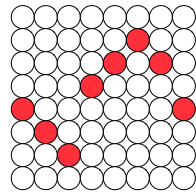
Figure 4: The Arpeggiator Display

**Playing Preset Arpeggios** There are five preset arpeggios: *up*, *down*, *up-down*, *chord*, and *random*.

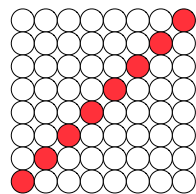
## 6 The Step Sequencer

## 7 The Recorder

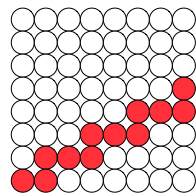
## 8 The MIDI Gauge



Example Arpeggio



Ascending Notes  
in Sparse Mode



Ascending Notes  
in Dense Mode

**Figure 5: General Gauge Display and Frequent MIDI Display** These are simple displays which take up the entire screen. This occurs if the MIDI IN channel is something other than NO CHANNEL (See **Options**→ **In MIDI**, in Section ??).

MIDI messages are displayed as follows: Some MIDI messages are extremely frequent and essentially impossible to display usefully.

**General Gauge Display** These are simple displays which take up the entire screen.

- **Pitch Bend** The 14-bit value is displayed
- **Program Change** “PC” is displayed, followed by the value.
- **Channel Control** Text is scrolled indicating CC, the parameter, and the value.
- **NRPN** Text is scrolled indicating NRPN, the parameter, and the value (both MSB-only, and MSB+LSB)

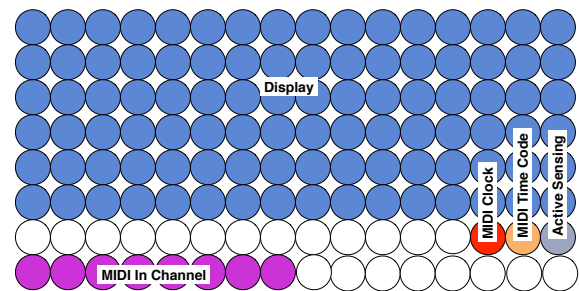


Figure 6: General Gauge Display and Frequent MIDI Display

**RPN** Text is scrolled indicating NRPN, the parameter, and the value (both MSB-only, and MSB+LSB values), plus whether the value was set, incremented, or decremented.

**System Exclusive** "SYSX" is displayed.

**Song Position** "SPOS" is displayed.

**Song Select** "SSEL" is displayed.

**Tune Request** "TREQ" is displayed.

**Start** "STRT" is displayed.

**Continue** "CONT" is displayed.

**Stop** "STOP" is displayed.

**System Reset** "RSET" is displayed.

**Channel**      **Display on Second to Bottom Row, Left Side**

16  
15  
14  
13  
12  
11  
10  
9  
8  
7  
6  
5  
4  
3  
2  
1

All (OMNI)

OFF

Figure 7: MIDI Channel Values

- **Active Sensing**
- **MIDI Time Code**
- **MIDI Clock**
- **Channel Aftertouch.** This one is special. If you press the **middle button**, you can toggle whether aftertouch is displayed as a simple toggled LED, or if the display should say “AT”, followed by the aftertouch value.

Figure 8: Note Gauge Layout

8



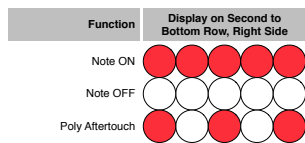


Figure 9: Note Function. The note is displayed by showing the note (such as B $\flat$ ), with the octave number directly below it. MIDI has eleven different octaves (0–10): Middle C is the bottom note of octave 5. The octave number is displayed as shown in Figure ?? . As usual, the MIDI In Channel is shown below that.

## 9 The Controller

The Controller allows you to assign to each of two buttons and two potentiometers any NRPN, RPN, Control Change (CC), or Program Change (PC) value. These assignments are stored in Flash memory and survive reboots and power cycling.

### Go

This enters the controller proper. Turning the left or right knobs, or pressing the Select or Middle buttons, will cause them to issue MIDI messages as assigned (below). If you press the Back button, you can exit the controller.

When you turn a knob or press a button, and the knob/button has had its control type set to something other than OFF, then the appropriate MIDI message is sent, and the value of the message is displayed as a number on-screen. Messages are only sent if the MIDI OUT channel is something other than NO CHANNEL (See **Options** → **Out MIDI**, in Section ??).

Note that for the time being the controller will only send (MSB) values between 0...127. It cannot send 14-bit NRPN and RPN values. This is largely a limitation of the resolution of the potentiometers.

### Left Knob

This submenu lets you select your control type and parameter number for the left potentiometer.

You are first given the option of what kind of **control type** (OFF, CC, NRPN, RPN, PC) you would like to manipulate with the left knob. If OFF, then turning the knob will not do anything.

After you have selected a control type, if you have selected CC, NRPN, or RPN, you will be then asked to select a **controller parameter number**. (Otherwise, for PC and OFF, you will go back to the top-level Controller Menu). CC parameter numbers may range 0 ... 119. NRPN and RPN parameter numbers may range 0 ... 16383. When you have completed this, you will be sent back to the top-level Controller menu.

### Right Knob

This submenu lets you select your control type and parameter number for the right potentiometer.

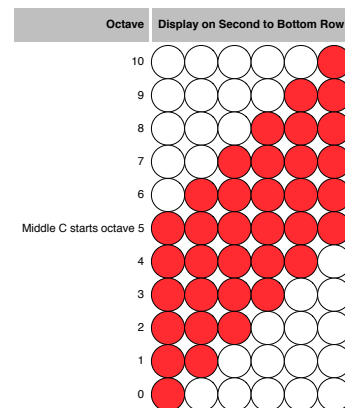


Figure 10: Octave Values

You are first given the option of what kind of **control type** (OFF, CC, NRPN, RPN, PC) you would like to manipulate with the right knob. If OFF, then turning the knob will not do anything.

After you have selected a control type, if you have selected CC, NRPN, or RPN, you will be then asked to select a **controller parameter number**. (Otherwise, for PC and OFF, you will go back to the top-level Controller Menu). CC parameter numbers may range 0 ... 119. NRPN and RPN parameter numbers may range 0 ... 16383. When you have completed this, you will be sent back to the top-level Controller menu.

### Middle Button

This submenu lets you select your control type, parameter number, and on/off values for the middle button.

You are first given the option of what kind of **control type** (OFF, CC, NRPN, RPN, PC) you would like to manipulate with the middle button. If OFF, then pressing the button will not do anything.

After you have selected a control type, if you have selected CC, NRPN, or RPN, you will be then asked to select a **controller parameter number**. (Otherwise, for PC and OFF, you will go back to the top-level Controller Menu). CC parameter numbers may range 0 ... 119. NRPN and RPN parameter numbers may range 0 ... 16383.

When you have completed this, you will be then asked to enter the value sent when the button is **pressed**. This value must be 0...127 (the Controller does not send 14-bit values). Afterwards, you will be similarly asked to enter the value sent when the button is **pressed again** (the buttons act as toggles). When you have completed this, you will be sent back to the top-level Controller menu.

### Select Button

This submenu lets you select your control type, parameter number, and on/off values for the select button.

You are first given the option of what kind of **control type** (OFF, CC, NRPN, RPN, PC) you would like to manipulate with the select button. If OFF, then pressing the button will not do anything.

After you have selected a control type, if you have selected CC, NRPN, or RPN, you will be then asked to select a **controller parameter number**. (Otherwise, for PC and OFF, you will go back to the top-level Controller Menu). CC parameter numbers may range 0 ... 119. NRPN and RPN parameter numbers may range 0 ... 16383.

When you have completed this, you will be then asked to enter the value sent when the button is **pressed**. This value must be 0...127 (the Controller does not send 14-bit values). Afterwards, you will be similarly asked to enter the value sent when the button is **pressed again** (the buttons act as toggles). When you have completed this, you will be sent back to the top-level Controller menu.

## 10 Options

Options sets global parameters for the device. Once set, these parameters are stored in Flash memory and so will survive a power cycle. Some options can be also accessed from certain other applications as a convenience.

### Tempo

If Gizmo is following its own internal clock rather than relying on an external MIDI clock, this specifies how fast a quarter note is. If Gizmo is following an external MIDI clock, this value is ignored, and the MIDI clock's specification of a quarter note is used instead. (See **Options**→**MIDI Clock**). When not in **Bypass Mode**, the tempo is shown by the pulsing on/off of the **Beat / Bypass Light** (see Figure ??).

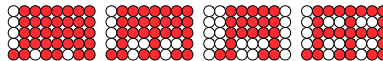


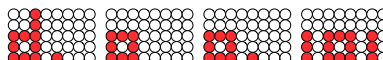
**Choose:** 1 ... 999 Beats Per Minute. Note that this is a large number, and so may require you to choose it with the left potentiometer, then fine-tune it with the right potentiometer.<sup>12</sup>

### Note Speed

Various applications (arpeggiators, step sequencers) produce notes at a certain rate relative to the tempo. For example, though the tempo may specify that a quarter note is set to 120 Beats Per Minute, the arpeggiator might be generating eighth notes and so is producing notes at twice that speed. You specify the note speed here.

Note speed is shown by pulsing the **Note Pulse Light** (Figure ??).

**Choose:**

Eighth Triplet	1/24	Beat	
Quarter Triplet	1/12	Beat	
Thirty-Second Note	1/8	Beat	
Half Triplet	1/6	Beat	
Sixteenth Note	1/4	Beat	
Triplet	1/3	Beat	
Eighth Note	1/2	Beat	
Quarter Note	1	Beat (duh)	
Quarter Note Tied to Triplet	1 1/3	Beats	
Dotted Quarter Note	1 1/2	Beats	
Half Note	2	Beats	
Half Note Tied to Two Triplets	2 2/3	Beats	
Dotted Half Note	3	Beats	
Whole Note	4	Beats	
Dotted Whole Note	6	Beats	
Double Whole Note	8	Beats	

### Swing

Swing, or **syncopation**, is the degree to which every even eighth note is delayed. 0% means no swing at all. 100% means so much swing that the even note plays at the same time as the next even eighth note. That's a lot.

**Choose:** 0% ... 100% (larger percentages are more swing, that is, longer delay every other note)

<sup>12</sup>Gizmo can go lots faster than 999: in theory it could go clear to 31200 or so. But then your synthesizer would explode and we wouldn't want that.

### Transpose

Here you can state that any notes generated by Gizmo be **transposed** up or down by as much as 60 notes. If a note is transposed to the point that it exceeds the MIDI range, it is not played. When appropriate, some applications (such as the MIDI Gauge) ignore this option.

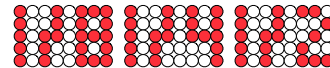
*Choose:* -60 ... 60

### Volume

Like Transpose, here you can state that any notes generated by Gizmo have their volume changed by multiplying their MIDI velocity by some value. If a resulting note velocity exceeds the maximum (127), it is bounded to 127.

*Choose:*

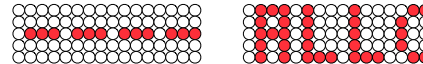
1/8  
1/4  
1/2  
1(default)  
2  
4  
8



### In MIDI<sup>13</sup>

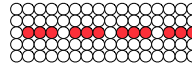
Many applications expect notes and other controls to come in via a specific input MIDI channel. You specify it here.

*Choose:* No Channel, Channels 1...16, or ALL Channels



**Out MIDI** Many applications emit notes etc. via a specific output MIDI channel. You specify it here. Other applications can emit notes on several different channels, in which case this value determines the default channel used.

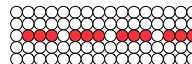
*Choose:* No Channel, or Channels 1...16



### Control MIDI

The arduino can be controlled via NRPN messages. You specify the channel here on which it will listen for them

*Choose:* No Channel, or Channels 1...16



The NRPN Messages are:

---

<sup>13</sup>Why aren't these called MIDI In and MIDI Out? Because then they'd be indistinguishable on the menu screen before the text started scrolling.

<i>NRPN Parameter</i>	<i>Min Value</i>	<i>Max Value</i>	<i>MSB/LSB</i>	<i>Description</i>
0	0 (released)	1 (pressed)	MSB	Press/Release the Back Button
1	0 (released)	1 (pressed)	MSB	Press/Release the Middle Button
2	0 (released)	1 (pressed)	MSB	Press/Release the Select Button
3	0	1023	MSB + LSB	Turn the Left Potentiometer
4	0	1023	MSB + LSB	Turn the Right Potentiometer
5	0 (un-bypass)	1 (bypass)	MSB	Toggle Bypass
6	Any Value	Any Value	MSB	Unlock Potentiometers. When an NRPN message is received, normally the on-board potentiometers are locked so that turning them has no effect. Pressing an on-board button unlocks them: so does sending this NRPN message.
7	Any Value	Any Value	MSB	Start Clock. If Gizmo's clock is stopped, resets and starts it. When appropriate, sends a MIDI START message.
8	Any Value	Any Value	MSB	Stop Clock. If Gizmo's clock is playing, stops it. When appropriate, sends a MIDI STOP message.
9	Any Value	Any Value	MSB	Continue Clock. If Gizmo's clock is stopped, continues it. When appropriate, sends a MIDI CONTINUE message.

## MIDI Clock

Gizmo can respond to a MIDI clock, ignore it, or emit its own MIDI clock. The use of the **Options**→**Tempo** setting will depend on the setting chosen here.

### Choose:

Ignore	Use an internal clock but let any external MIDI clock pass through.
Use	Use an external MIDI clock and also let it pass through.
Consume	Use an external MIDI clock but don't let it pass through.
Generate	Use an internal clock and emit it as a MIDI clock. Don't let any external MIDI clock pass through.
Block	Use an internal clock but don't emit as a MIDI clock. Don't let any external MIDI clock pass through.

## Screen Brightness

This changes the brightness of the LED matrix.

**Choose:** 1...16 (higher values are brighter)


## Menu Delay

Gizmo often will display text (such as "SWING") by filling the screen with the first few letters (in this case, "SWI"), then pausing for a certain delay, then scrolling through the whole word horizontally. You can specify how long that pause is here.

### Choose:

0	Seconds	(scroll immediately)
1/8	Seconds	
1/4	Seconds	
1/3	Seconds	
1/2	Seconds	



1	Second	(default)	
2	Seconds		
3	Seconds		
4	Seconds		
8	Seconds		
$\infty$	Seconds	(never scroll)	

### **Voltage (or No Voltage**

Toggles whether Gizmo will send voltage information out via the optional DACs in response to notes played or to Controller potentiometers being turned.

### **The About Screen**

Presently says: GIZMO V1 (C) 2016 BY SEAN LUKE