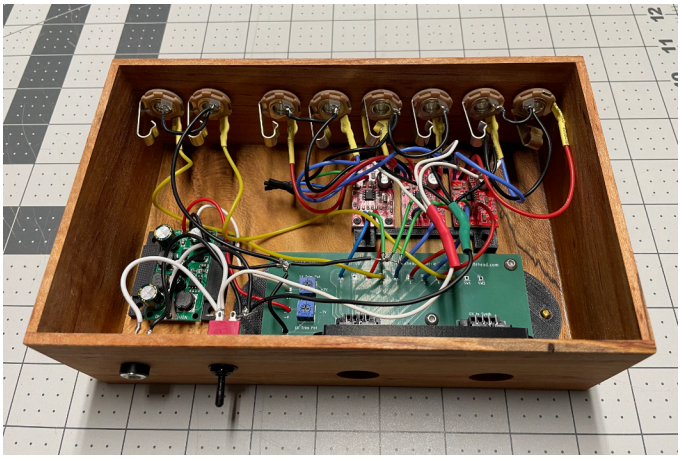
System Overview

The AMT(P) application is a polyphonic, audio to MIDI translator designed for instruments that use divided pickups. A divided pickup is one that has a separate pickup for each string. The application is a companion to the audio breakout boards designed by me, Jeff Whitehead, and published on my site, CoolLutherie.com, for you to build. To use the system, the 13-pin DIN instrument cable that would normally plug into a hardware synthesizer is, instead, plugged into the breakout board. Then, from the breakout board, an audio connection is made for each string to the computer's audio interface. The AMT(P) application reads the audio signal from each string and translates the notes to MIDI values. Additionally, AMT(P) can transmit pitch bend and modulation MIDI messages. For instruments that do not have divided pickups, AMT(P) can operate in mono mode, reading a single audio input. However, this mode is not polyphonic.



Here is an interior view of an audio breakout board that supports a guitar with a divided pickup such as the Roland GK-3 or the Graph Tech Ghost system. There is an option to connect to a hardware synthesizer in parallel or to use it without the synthesizer by powering the board with a 9 volt pedal board power supply.

Audio from each string is sent to a separate jack. Audio from the guitar's normal pickups is sent to its own jack. The board shown here supports an additional audio signal that can be achieved by hacking the GK's OEM pickup system, taking advantage of an unused connection within the 13-pin DIN cable.

Here is a breakout board connected to a Behringer ADAT. The ADAT provides an amplified audio input for each string as well as the normal pickup and can connect to an audio interface via an optical connection. The breakout board shown here can optionally support up to three hardware synthesizers running in parallel with the audio translation. Using an ADAT is a convenient method for processing polyphonic audio without consuming input jacks on the audio interface.

The breakout boards and AMT(P) application can also support a violin with the Cantini ISSP2 Earphonic bridge.

Here is a Cantini-equipped fiddle connected to a breakout board and a Roland GR-55 synthesizer.

There are numerous options for configuring an audio to MIDI system that best suits your needs. I provide all of the information as open source so that artists have the freedom to create the most appropriate system. Go to my site, CoolLutherie.com, to learn more.

Software Overview

With the instrument connected to an audio interface, either polyphonically via the breakout board as multiple channels (one for each string, plus one for the normal pickups), or monophonically through a single channel, the AMT(P) application will be able to read the audio channels and translate the inbound audio streams to MIDI notes. It is important to set the audio preferences correctly and to tune each string's input to match your playing style. There are visual controls to assist you.

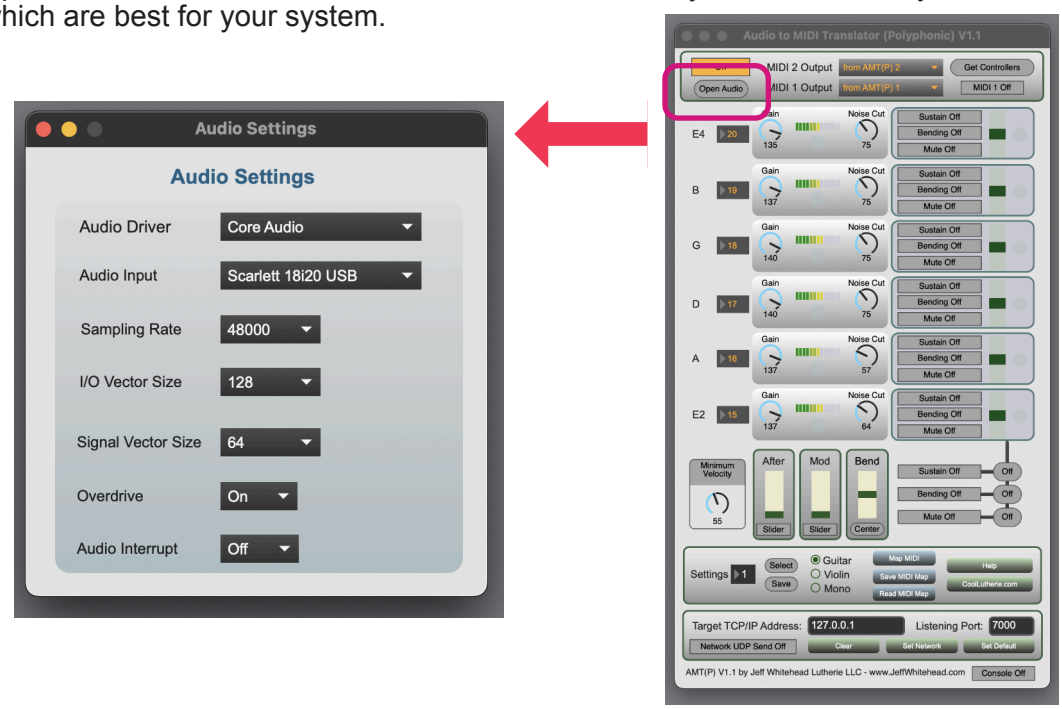
Translating stringed instrument audio to MIDI notes, accurately, is difficult. This is because the frequencies generated by a vibrating string are complex. The software implements a variety of technical solutions to achieve superior results, and it is helpful to understand some of the issues so that your playing can complement them. First, and foremost, your instrument must be in tune. Being in tune dramatically reduces the translation error rate. Secondly, while a string continues to ring, it is difficult to sense the attack of a new note. So, strumming chords is pretty messy, but an articulated arpeggio is not. Hammer-ons are easy to sense, pull-offs are not. The cleaner you play, the better the resulting translation.

For each string, there is a Gain setting and a Noise Cut setting. The higher the gain, the more sensitive the translation; however, that also introduces noise to the signal which can slow the processing. The noise is reduced by increasing the Noise Cut control. The blue LED will light when audio is being processed and the green LED will light when a MIDI note is being sent. Every guitar and audio interface is different. So, it is important to tune the application to each string for best results.

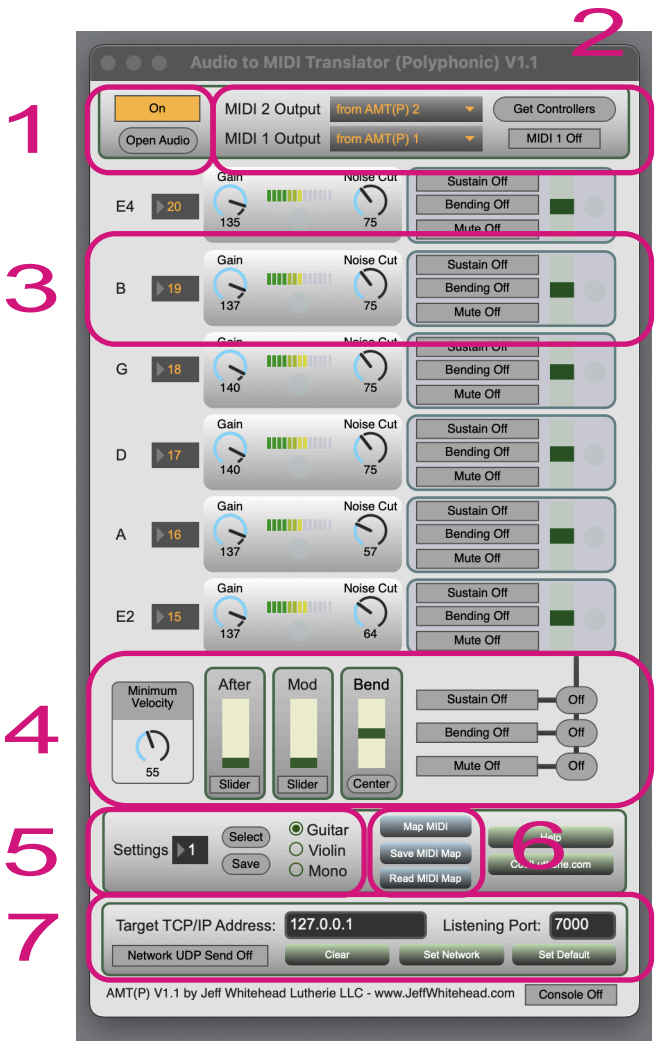
Translation occurs through digital sampling of the audio stream. Sampling rates of 44.1K and 48K are supported. Anything higher than that will not work. Since the process is one of MIDI translation, high sampling rates have no positive effect on the results anyway; therefore, you do not need an expensive audio interface to have excellent results.

The audio interface is configured through a popup panel. A Sampling Rate of 48000, I/O Vector Size of 128, and Signal Vector Size of 64 have worked well on Macs with Intel or M1 chip sets. It is important to have the Scheduler in Overdrive. The Audio Interrupt is normally set to off. These settings will optimize the audio to MIDI translation, reducing latency. To learn more about how Max runtime works, please refer to the documentation by Cycling '74. (cycling74.com)

Every computer and audio interface will be different. Therefore, you will need to try different settings to discover which are best for your system.



Description of the Input and Output Controls



1: Audio Interface. Here, you set the audio interface through which the guitar or violin audio signal is being sent to the computer. The On button activates the audio input and defaults the application to setting #1. The Open Audio button enables you to configure the audio interface.

2: MIDI Out Settings. There are two drop-down menus from which you can set the MIDI output controllers. The app provides two of its own MIDI output controllers, “from AMT(P) 1” and “from AMT(P) 2.” These are the recommended controllers to use. If these controllers are not visible in the menus, the Get Controllers button will refresh the controller list from your computer. MIDI 2 with MPE is sent through one controller, and MIDI version 1 through the other simultaneously. MIDI 2 with MPE will result in superior note triggering due to the modern protocol. MIDI 1 is available for software that does not support MIDI 2 and is toggled off by default.

3: String Audio Controls. The audio input channel is entered into the field. There are two adjustments available for each string. Gain sets the inbound audio level. Noise Cut sets a volume threshold for the system to ignore inbound audio in order to eliminate false note triggering. A string can have individual settings for sustain, bending, and muting. These can also be turned on and off through the global controls at the bottom.

4: Global settings. Minimum Velocity sets the minimum MIDI velocity for every note being translated. The Sustain, Bending, and Mute buttons affect all strings.

Global MIDI messages MIDI Bend, Modulation and Aftertouch are sent out when the sliders are moved. The Mod controller number used is the standard, CC #1.

5: Settings. Up to three settings can be saved. To switch settings, enter the number 1, 2 or 3 in the input field and click the Select button. To save the current settings for the app, click the Save button. The settings will be saved into the setting number currently displayed. The instrument toggle sets the app to guitar, violin, or mono mode. For either instrument, the strings are calibrated to standard tuning for audio to MIDI translation, which is the standard for hardware guitar synthesizers. In mono mode, only one audio interface is supported. This is to support instruments without a divided pickup.

6: Mapping MIDI controls. An external controller (such as a foot pedal) can be mapped to the app’s controls. Once mapped, press the Escape key to leave the mapping mode. Maps can be saved to, and recalled from, an external file.

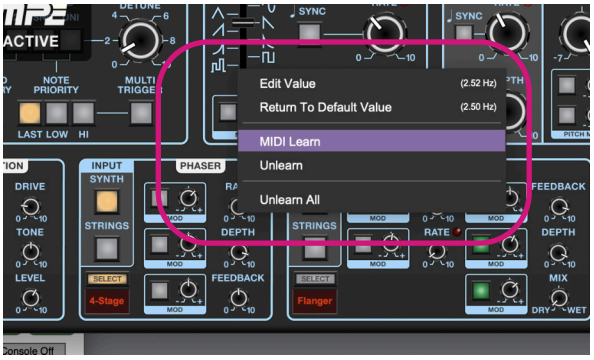
7: Network broadcast. Polyphonic data can be sent as JSON across the network via UDP for integration with other software.

Operating as a MIDI controller for software synthesizers

AMT(P).app becomes a MIDI controller in your environment. For this example, I am using Dreamsynth by Cherry Audio; (cherryaudio.com) a gorgeous, polyphonic synthesizer with multiple oscillators, analog strings, MPE support and a host of features that create a vast palette of tone.

Dreamsynth and other amazing synthesizers are available as a downloads with a free trial period from Cherry Audio. I encourage you to explore the many fabulous presets that they provide with each. The ability to play these software instruments polyphonically with a guitar or violin opens a completely new, transformative, sonic landscape.

Configuration is simple. With AMT(P) running, click the settings icon on the Dreamsynth menu bar, then the Audio/MIDI tab and activate the MIDI input that you selected for MIDI 2 output in AMT(P). Set the audio output in Dreamsynth to your preferred settings. Note that the larger the audio buffer size, the greater the latency in producing sound. So, the smallest setting without producing audible issues like pops and clicks is usually best.

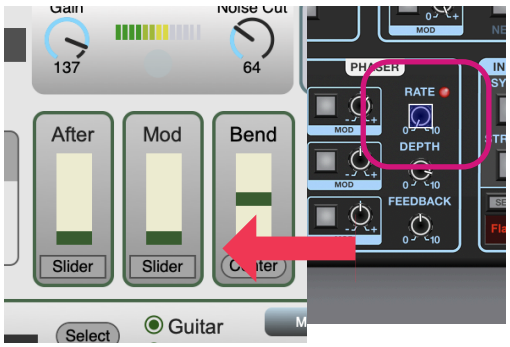


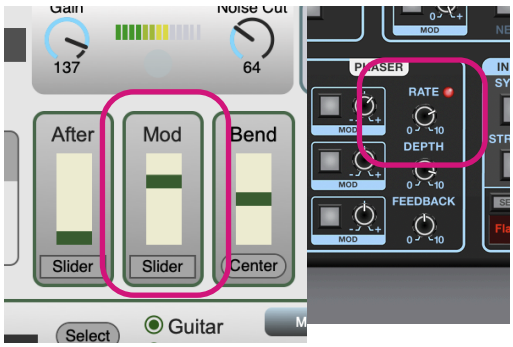
Your guitar or violin will now perform as a MIDI controller for Dreamsynth. In addition to playing notes, AMT(P) sends pitch bend and modulation MIDI messages, just like a keyboard controller. Therefore, the Bend and Mod sliders will interface with a software synthesizer automatically. AMT(P) sends the Mod message as CC #1, which is the standard MIDI value for a modulation wheel.

If you wish to map a slider to additional controls within Dreamsynth you can do so through a “MIDI Learn” process.

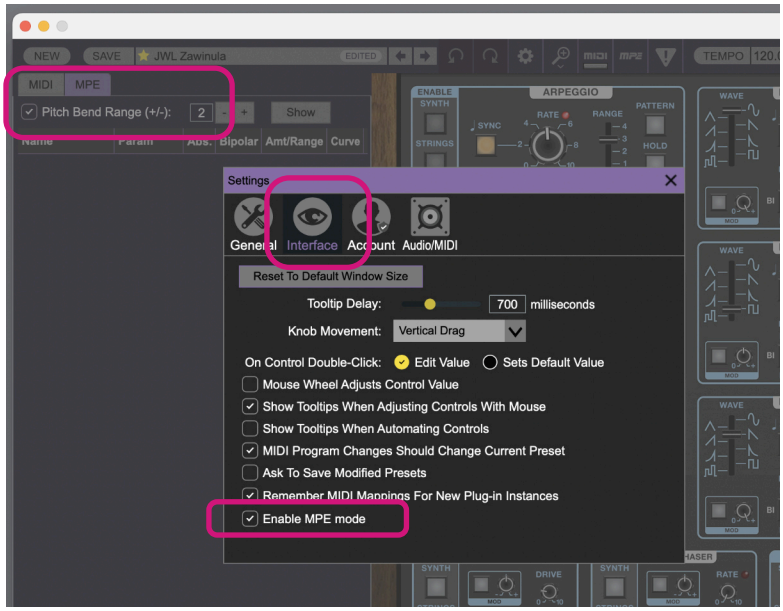
For example, to connect the Mod slider in AMT(P) to a knob in Dreamsynth, control-click the knob and select “MIDI Learn” from the pop-up menu. (This example will connect the phaser rate knob.)

The knob will be highlighted in a box, indicating that it is waiting for a MIDI signal. Then move the Mod slider in AMT(P) to complete the connection.





Once learned, when you move the slider, you should observe that it controls the knob setting. To disconnect the control, simply control-click the knob and select the option to “Unlearn.”



To enable MPE mode within Dreamsynth, from Settings, select the Interface tab, then check the box next to “Enable MPE mode.”

To adjust the pitch bend of the synthesizer to match the guitar string’s bending range, click MPE at the top menu to open the MPE side panel. Then, check the box next to “Pitch Bend Range” and set the value to 2.

String bending will now be possible for individual notes being played by the polyphonic synthesizer as well as globally from the Bend slider control from AMT(P).

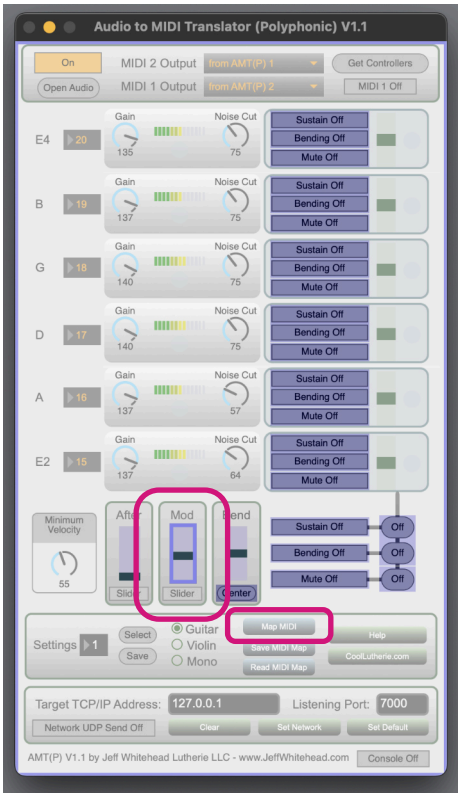
Global bending will affect all notes equally and simultaneously.

Operating an AMT(P) control from another controller

Just like utilizing MIDI Learn within Dreamsynth, it can also be used within AMT(P) to allow control from another MIDI device. For example, you might want to have a foot pedal from another controller move the Mod slider that you just mapped to the Dreamsynth’s Phaser Rate knob.

Doing so is similar to the operations within Dreamsynth.

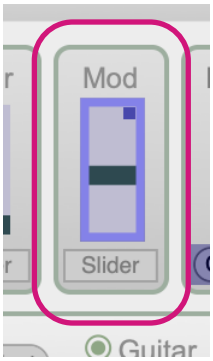
Click the “Map MIDI” button. The controls that can accept a MIDI mapping will be visible while everything else will be disabled. Select the control to be mapped. In this example, that is the Mod slider. It will be highlighted in a box to indicate it has been selected for MIDI learning.



Then, move the foot pedal (or whatever MIDI control you wish to map) and you will see the slider respond and a small block will appear on the slider indicating that it has been assigned to the MIDI control.

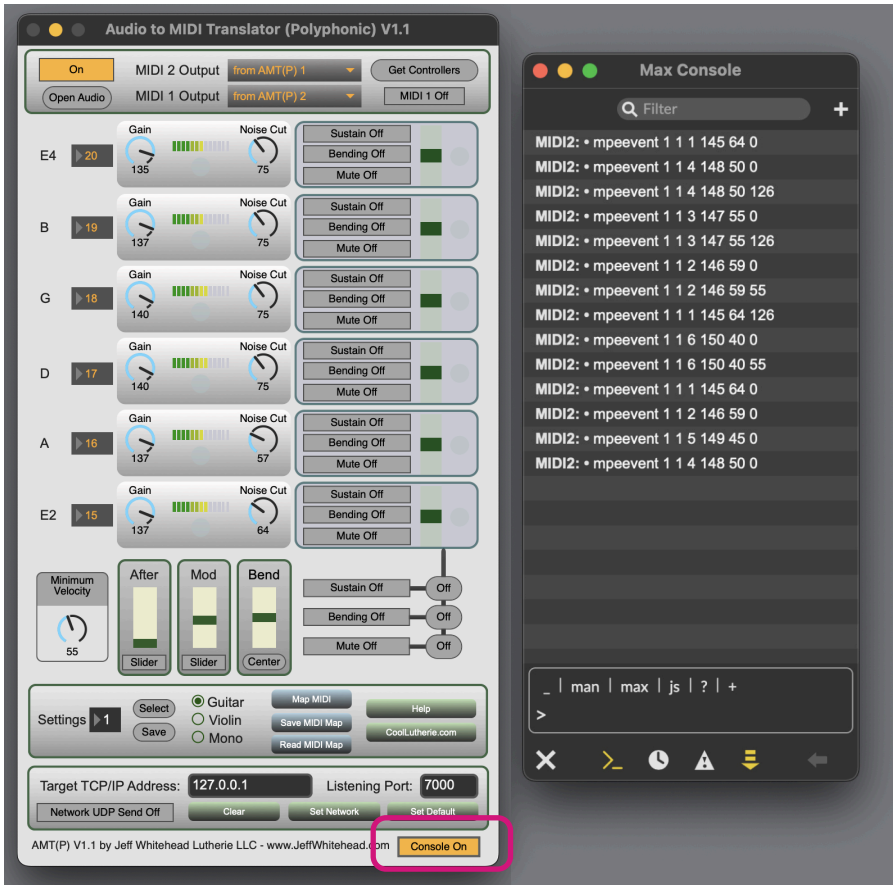
Undoing the mapping is simple; with Map MIDI active, control-click the slider and select “Delete Mapping.” You will see the small block disappear and the mapping will be dropped.

To leave the MIDI mapping mode, press the Escape key.



Using the Max Console to view MIDI output

Being a Max application, AMT(P) can take advantage of the real time signal processing and data display that the Max environment provides. One of the most important utilities from Max is the Console. The Max Console displays information in real time that the developer wishes to post. For AMT(P), that information is the MIDI 2 messages that are being sent from the application through the selected controller output.



These messages include notes on and off with velocity and channel information. Pitch bend and Mod messages are also displayed.

If you are having difficulty integrating with another MIDI application, it can be helpful to see what is being sent out from AMT(P) in real time.

If you are a visual artist, using the MIDI messages to control live performances, the console is a valuable tool to view the actual data.

To launch the Max Console, from the AMT(P) main menu, select “Window” then “Max Console.”

To turn on the message output to the console, click the Console On/Off button.

About MIDI 2.0 and why it is important

Everything that you need to know about MIDI can be found at the association’s website midi.org. If you have not yet joined the association, I encourage you to do so. It is free to join as an individual member, and there is a wealth of information about the MIDI standard and the companies, products, and projects that are using it.

The reason that AMT(P) utilizes MIDI 2, apart from its being the latest specification, is that it offers the ability to efficiently send detailed information about the note being played on a stringed instrument. Since it is possible to play the same note on multiple strings, this information can be expressed as a sub-channel. In the screen shot above, you can see a main channel of 1, followed by a sub-channel value for the string, then the control information, note, and velocity within each message. With the older MIDI 1 specification, polyphony required a separate main channel for each string which could confuse synthesizers while processing duplicate notes coming from multiple channels. This problem of duplicate note processing would often lead to dropped notes or, worse, notes that would never be turned off.

Better yet, MIDI 2 includes a new standard, MIDI Polyphonic Expression (MPE). This new messaging enables the transmission of expressive controls for individual notes. For a guitar, that means that a pitch bend can be associated with a particular note on a particular string in addition to pitch bending being associated with all notes within the main channel. This is a fundamentally new capability.

There is a lot more to MIDI 2 and MPE. The electronic music industry is rapidly transforming because of it.

Recommended Practices for Optimal Performance

For accurate audio to MIDI translation, there are techniques for setting up the guitar or violin, the audio interface, and the AMT(P) software to achieve optimal performance.

Before any other activity, ensure that your instrument is in tune. The AMT(P) app is sampling audio wave forms from a stringed instrument that are very complex. A properly tuned instrument will contribute to accurate translation with minimal latency.

The divided pickup, (a Roland GK-3 or a Graph Tech Ghost Hexpander for the guitar or the Cantini ISSP2 for the violin), produces a relatively weak audio signal. ***Therefore, if you are not using a breakout board with microphone preamps installed, it is important to boost the audio output of the divided pickup coming through the breakout board in order to achieve enough of an audio signal to perform accurate audio to MIDI translation.*** An audio interface with a separate volume control for each input will suffice, increasing the volume of each string to a point just below clipping. Such inputs are usually microphone inputs on the audio interface, ADAT, or mixing board. Some audio interfaces provide preamp controls with volume adjustment for line inputs. These inputs will also work because they can boost the audio signal to a usable level. A line input without amplification for the output of the divided pickup will not provide an adequate audio signal for accurate MIDI translation.

Within the AMT(P) app, increasing the Gain control until the blue LED flashes *while a note is not being played*, indicates the maximum acceptable input volume for the string. At that point, the app is reading only noise from the audio signal and attempting to perform audio to MIDI note translation. Backing off the gain until the LED goes off brings the signal below the noise threshold and to its most sensitive sampling state.

Playing a note on the selected string should trigger the blue LED. Depending upon your playing style, such as finger picking versus using a plectrum or bowing, you may adjust the gain until the LED flashes at the moment of attack. The illuminated LED indicates that there is audio signal available for translation. When the note is actually being translated, the green LED will be illuminated.

After one string is set to the desired Gain level, move on to the next and continue until all strings indicate audio to MIDI translation capability by way of the illuminated LED's.

If a note on a string is being triggered while a different string is being played, then the Noise Cut control on the string that is not being played may be increased to filter out the ambient noise. Between the adjustment of the Gain and Noise Cut controls for each string, an optimal setting can be achieved for each string in accordance with your playing style.

Sensing a duplicate note from a stringed instrument without creating false duplicate notes is a complicated matter. Related in complexity is the ability to distinguish a sustained note from a re-triggered note. If you wish to rapidly re-play a note, it is best to dampen the note when played so as to reduce the sustained vibrations.

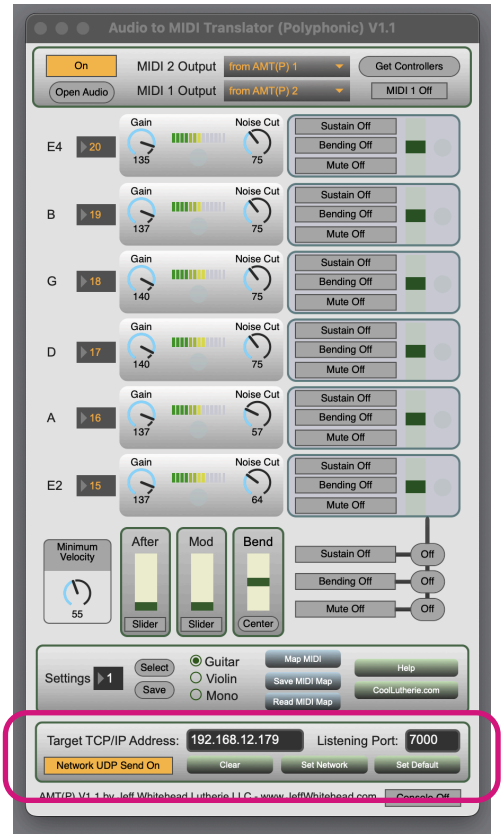
Lastly, the global Minimum Velocity control sets the minimum MIDI note velocity for all notes being played. The numeric value of the Minimum Velocity control is a MIDI note velocity, ranging from 0 to 127. Setting a high value will result in stronger note attack by the MIDI instrument when the note is played. Some synthesizer software is more responsive when the note velocity is higher.

After all settings have been made, it is important to save them. The input field for Settings accepts three different settings. Click on the numeric field to enter the desired number, then drag your mouse up and down over the displayed number, or type in the value to change it. While the Settings number within which you chose to store your settings is displayed, click the Save button to save them to that Settings number. All of your global and string settings will be saved within the app. To recall a Setting, enter the number to be retrieved and click the Select button.

Note: If the MIDI controllers that are active on your computer change, the MIDI 1 and 2 controllers that you saved may need to be re-selected when the app is re-launched. This is because the index value of the saved controllers will no longer match the name of the controller you wish to be using. Click the Get Controllers button to refresh the list, then reselect the controller that you wish to use.

Network broadcasting of polyphonic data

TouchDesigner is a real time graphics design environment and display engine that is available to download from derivative.ca. For integration with TouchDesigner, AMT(P) converts the polyphonic MIDI information to JSON and transmits it over the network as a UDP broadcast. Within TouchDesigner, this JSON can be read in real time and translated to data that manipulates visual performances. TouchDesigner can be running on a Mac or Windows computer within the same network.

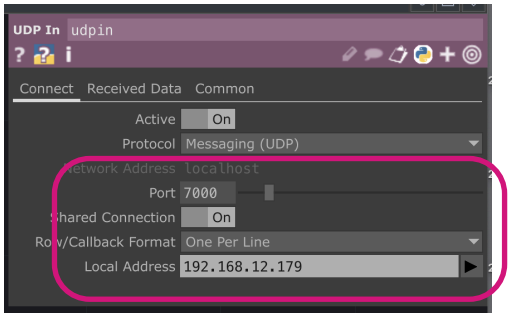


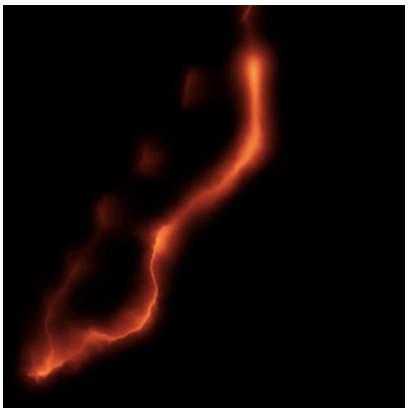
If TouchDesigner and AMT(P) are running the same computer, network broadcasting can be used to intercommunicate by entering the computer's local TCP/IP address: 127.0.0.1 and selecting a Port value that is not in use by other applications. The default value for the Port is 7000.

When the applications reside on separate machines, the network TCP/IP Address and listening Port of the TouchDesigner computer are set within both applications.

AMT(P) only broadcasts polyphonic data. When in Mono mode, the network broadcasting will by turned off automatically.

A fully-functional TouchDesigner example is available at Cool Lutherie. Drop the code into your own project, set the values for Local Address and Port within the TouchDesigner "UDP In" object, and your network will be instantly connected to AMT(P). If not, check to ensure that the computers allow UDP. Sometimes a firewall rule must be added.





A note-manipulated animation in TouchDesigner.

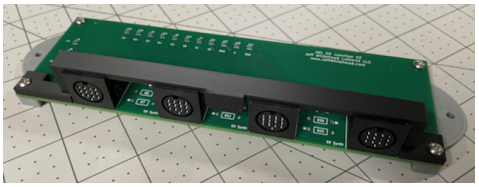
About the author, Jeff Whitehead (Jeff Whitehead Lutherie LLC)

I have been making stringed instruments for many years. It started in my youth, along with an interest in electronics. During my middle life, I was a software engineer and entrepreneur. So, the natural progression was to bring instrument design and computer processing together. This Audio to MIDI Translator application is an important part of that journey, and I am delighted to be able to share it with you.

My instruments are enjoyed by professional musicians, collectors, and enthusiasts. Each is unique. My process involves a lot of experimentation; something that I enjoy immensely. My instruments can be viewed at JeffWhitehead.com.

To share the R&D experience that I have acquired through the years of instrument building and software development, I created the site, CoolLutherie.com.

At Cool Lutherie, you may watch videos about this application and other projects that showcase fun and interesting things in the world of electronic stringed instruments and computer integration. From tshere, you can learn from, and download, helpful content. If you do not yet have one, the most important project to go along with this software is the audio breakout board. The audio breakout board splits the analog output from the 13-pin DIN cable to separate audio jacks. This is the critical piece of hardware to experience true, polyphonic processing. I have designed a variety of PCB's to account for most applications. These range from a simple board that supports a single, 13-pin socket to a junction that can interconnect up to three hardware synthesizers to run in parallel with the audio processing. These PCB's and other related parts can be purchased through Cool Lutherie. I publish all of the designs so that you are free to make them yourself.



It is important to me that this information is provided as open source, enabling you to create your own hardware and software to meet your artistic needs. Therefore, this application and other projects that I provide are done so under the Creative Commons licensing. This also means that my work is supported by donations. So, if you find my content helpful and interesting, please let me know by making your donation at CoolLutherie.com. Your support is greatly appreciated!

The reason that I focused on MIDI integration with stringed instruments, in the first place, was that I found hardware guitar synthesizers to be limited in capability when compared to the exploding software market. I wanted a simple, polyphonic interface that allowed me to use the cool stuff that was being enjoyed by keyboard players. And… I wanted to develop something that could be easily shared with other artists. So, I created this system.

Once you begin working with MIDI integration, new artistic opportunities present themselves, such as MIDI-triggered visual performances, stacking software synthesizers for a huge sound, combining computer with analog audio, adding additional modulation through controllers, and more; a whole new world of possibilities for your studio and live performances shall be open to you.

Where will MIDI lead you?

Enjoy the journey!

Important Information

About this application

This application is written in Max, a premier development environment for cross-platform audio applications. It is the product of the company, Cycling ‘74; therefore, you may refer to the Cycling ‘74 forums to learn more about the Max environment and applications created in Max. Go to: cycling74.com/forums for more information.

This Audio to MIDI Polyphonic Translator application, AMT(P).app, written by Jeff Whitehead, was compiled within Max 9 to run as a standalone app on a Mac computer and is available as an official distribution only from CoolLutherie.com.

If you have not received this software from CoolLutherie.com then it may not be an official distribution.

Source code and tutorials for audio to MIDI translation are available through CoolLutherie.com and can be modified within the Max software development environment, a product of Cycling ‘74. (cycling74.com)

Intended Use

The AMT(P) app is a companion program to the open source audio breakout boards designed by Jeff Whitehead Lutherie LLC, and available as a DIY project through CoolLutherie.com. An audio breakout board separates the output coming from the divided pickup and sent through a 13-pin cable. The output of each string, as well as the output from the normal pickups, are sent onward by the breakout board through separate jacks which can be connected to an audio interface. The AMT(P) app performs polyphonic translation of audio to MIDI notes by analyzing audio streams from the interface.

Download Contents

The archive download for this Mac standalone version includes the following files:

AMT(P).app - this application

AMT(P)_readme.txt - important information about this software

Warranty

License

To learn how to distribute your work under this license, please refer to:

https://creativecommons.org/licenses/by-nc-sa/4.0/

This license enables reusers to distribute, remix, adapt, and build upon the material in any medium or format for noncommercial purposes only, and only so long as attribution is given to the creator. If you remix, adapt, or build upon the material, you must license the modified material under identical terms. CC BY-NC-SA includes the following elements:

BY: credit must be given to the creator, Jeff Whitehead Lutherie LLC, at JeffWhitehead.com

NC: Only noncommercial uses of the work are permitted.

SA: Adaptations must be shared under the same terms.

Adaptations must include the attribution to the original and preceding authors and describe all changes that have been made to the preceding work.