Introduction

Being a musician, I have known of MIDI (Musical Instrument Digital Interface) for a a long time. But being a guitar player I never immersed myself in the technology like most keyboard players did. However, when I read about the SAM-2195 single chip synthesizer I knew it was time to dive in and learn something new. That is what's fun about electronics and computers; there is always something new to learn. Connecting up hardware devices and writing software to make it all work keeps those little gray cells in our brains active, happy and alive. If the end result of our experimentation is something useful, like the MIDI Buddy, that is icing on the cake.

MIDI Buddy has been my platform on which to experiment with and learn more about MIDI. This initial foray into MIDI has helped me to understand the physical protocols. Future investigations will focus on using MIDI to produce sound directly using a micro processor/controller which may in fact lead to another article so stay tuned. First question to answer: “What is MIDI anyway?”

What Is MIDI ?

MIDI is a technical standard/protocol that allows a wide range of electronic musical instruments, musical equipment and computers to communicate. MIDI technology was standardized in 1983 and the standard is maintained by the MIDI Manufacturers Association (MMA). The MIDI specification defines more than a software protocol to control devices. It also defines the connectors and cables which are to be used to convey the protocol. MIDI messages are made up of 8-bit bytes transmitted serially at 31.25 KBaud. The first bit of each byte identifies it as a status byte or a data byte and is followed by seven bits of information. A start bit and a stop bit are added to each byte for framing purposes, so a MIDI byte requires ten bits for transmission. A single MIDI connection can carry up to 16 channels of information each of which could be routed to a different MIDI device.

The MIDI protocol itself doesn't contain sounds but rather specifies a stream of messages/events that cause MIDI devices to generate sounds. This is why a MIDI file for a song will be much much smaller than say an MP3 file of the same song.

What kind of MIDI events are there you ask? There are many but events like NOTE ON when a keyboard key is depressed and NOTE OFF when it is released are simple examples. MIDI events can also control which instrument or voice plays the note (piano, nylon string guitar, tuba, etc.), how loud or soft the note volume should be, how much reverb should be applied to the note and much more. MIDI events provide both course and fine grained control of musical parameters. This allows the nuances of a musical performance to be captured accurately in a MIDI event stream.

Since MIDI's inception electronic musical instruments have continued to evolve so MIDI has had to evolve to stay pertinent. The original MIDI specification made connectivity between

MIDI devices possible but didn't define what voicing would be heard for which notes. In 1991 the General MIDI (GM) standard was established which defined a standard sound bank for MIDI devices (which supported GM) allowing a MIDI file created on one MIDI setup to sound similar when played on other, different setups. In addition, GM established MIDI channel 10 as the percussion channel where specific MIDI note values are mapped to specific percussive sounds. Finally, GM mandated a minimum of 24 note polyphony which would allow complex musical compositions to be adequately expressed.

The evolution of MIDI did not stop with GM. Newer standards like GS, XG and GM2 were developed to further refine MIDI's fine grain control and to take advantages of advances in electronic music hardware. These standards are all backward compatible with GM but not necessarily compatible with each other.

So What Is MIDI Buddy ?

MIDI Buddy is a stand alone device (see Photo One) consisting of the following hardware:

- A Teensy 3.1 micro controller
- An SD card interface for up to 2 GBytes of available storage
- A 2 line by 16 character LCD display used for displaying menus and status
- Two joysticks for menu navigation and control
- A standard MIDI input
- A SAM-2195 synthesizer chip
- A stereo output via 1/4" phone jax

MIDI Buddy is powered by a USB power supply plugged into the Teensy controller via a USB cable. The audio output of MIDI Buddy is very high fidelity with rich piano, pads and string sounds. The sound quality blew me away the first time I heard it.

Visualize MIDI Buddy as a standard MIDI input which feeds MIDI data into a serial input on the Teensy controller. A serial output on the Teensy streams MIDI data to the SAM-2195 synthesizer chip which converts (renders) the MIDI events to sound for output. The SD card interface also feeds MIDI events to the Teensy controller. Since the Teensy controller is the middle man in all this, it has the ability to:

- Pass the data it receives from the MIDI input through unmodified to the synthesizer chip for rendering into audio.
- Interpret and/or modify the data it received from the MIDI input before passing it along to the synthesizer chip.

- Generate MIDI data programmatically and pass it to the synthesizer chip
- Read a MIDI file and pass the decoded MIDI events to the synthesizer chip for playback

With the software I provide with this article, MIDI Buddy can:

- Function as a complete music synthesizer if a keyboard or other MIDI controller is plugged into its MIDI input. MIDI Buddy can then be played in real time and can sound like a piano, organ, cello, bagpipes, steel drums or any of the many other instrument voices supported by the synthesizer chip.
- Play any type 0 (single channel) Standard MIDI File (SMF) stored on the SD card. MIDI Buddy could function as a backing band for a solo performer with this functionality.
- Can assign any of the 256 synthesizer voices to any of the 16 MIDI channels
- Can enhance a musical performance by adding reverb, chorus and/or delays to the produced sounds.
- Record and playback (up to 1000 note) MIDI riffs to/from the SD card. In this respect MIDI Buddy can be considered a MIDI sequencer/librarian but without note editing capabilities.
- Play four demos which illustrate various aspects of MIDI.
  - Demo 1 – All voices demo. Plays all available voices using a random algorithmic rhythm.
  - Demo 2 – Note bending demo. Illustrates how pitch bending in MIDI sounds.
  - Demo 3 – Cross fading demo. Demonstrates smooth cross fading between two different voices on two MIDI channels.
  - Demo 4 – Drum pattern demo. Plays a programmatically generated drum pattern.
- Can be used as a platform for MIDI experiments

Since the Teensy 3.1 controller is running at 96 MHz and has 256K of flash (program memory) and 64K of RAM (most of which are still unused) a lot more functionality could be added to MIDI Buddy. Almost any MIDI function one could imagine could be added.

I've mentioned the SAM-2195 synthesizer chip used in the MIDI Buddy several times so I had better describe its capabilities now in a bit more detail. The SAM-2195 is a single chip (about the size of the fingernail on your little finger) with:

- 128 general MIDI instrument sounds and an additional bank of 128 variations built in

- 4 built in drum kits
- Supports 64 voice polyphony when not using effects or 38 voice polyphony with full effects
- 14 bits of pitch bend range
- Master volume and per channel volume control
- 8 types of reverb effects with variable level and variable feedback
- 8 types of chorus, flange and delay effects
- 4 band equalizer
- Spatial effects
- Portamento and modulation effects

When I first read about this chip I immediately tried to buy some only to find out they are impossible to get in small quantities. What I did find was a company called Modern Device that sells an Arduino shield called the Fluxamasynth containing the SAM-2195 chip for a reasonable price ($35 US). So I used this shield in a non standard way (which I will describe shortly) in MIDI Buddy. If I understand the history correctly, the SAM-2195 chip was originally designed by the US company Atmel but production has since been taken over by a French company called Dream. One other note. In March of 2014 the SAM-2195 was replaced by an even more powerful chip called the SAM-2695. I have yet to find one of these to play with, however.

I should note also that MIDI Buddy doesn't take full advantage of the synthesizer chip's power. I only implemented features I thought would be useful to me and to you of course.

Hardware

Figure One shows the schematic of the MIDI Buddy. Please refer to it during the following discussion. Since MIDI Buddy would be a one off, it was built using a breadboard and point to point wiring as can be seen in the various photos.

As mentioned, the MIDI specification dictates the hardware aspects of MIDI as well as the protocol aspects. The MIDI input shown in the upper right corner of the schematic conforms to that specification. Every MIDI device must be electrically isolated from other MIDI devices to prevent ground loops and other problems. In MIDI Buddy I used an opto-isolator for isolation. As serial MIDI data arrives at the MIDI input, the LED in the opto-isolator toggle with the data. The photo transistor in the opto-isolator responds to this activity and generates a digital signal which mirrors it. This newly derived digital signal is presented to the Serial 1 receive input on

the Teensy 3.1. A second Serial device on the Teensy, Serial 2, outputs serial MIDI data from the controller to the input of the Fluxamasynth. Software running in the Teensy configures both of these serial interfaces to run at 31250 bps which is close enough to the MIDI standard rate of 32.25 KBaud.

The LCD is connected to the Teensy using a simple four bit interface. Any 16x2 LCD display compatible with the Arduino LiquidCrystal library can be used. A 10K ohm 20 turn trimmer was used for the LCD's contrast adjustment. This trimmer is adjusted to make the LCD readable.

The Flumamasynth shield was meant to be directly plugged onto an Arduino form factor device. Here however, the shield is stood off from the bottom of the breadboard with inline connectors and the five connections required between the shield and the breadboard are directly wired. NOTE: a jumper must be installed on the shield to configure the input to the shield. MIDI Buddy requires pin one as the input.

The Teensy controller's SPI interface is used to interface to the SD memory card. It seems like every SD breakout board I come across uses a different labeling scheme for the signal connections so please ignore the pin numbers on the schematic and make the connections to the Teensy based upon signal names. You should run the SdFat library example programs to verify you have the SD card connected correctly. Just remember to set the SD card chip select in the example programs to pin 14.

The two joysticks are connected to analog and digital inputs on the Teensy. Each joystick has a side to side or horizontal potentiometer and an up and down or vertical potentiometer. In addition each joystick has a switch which is activated by pressing down on the joystick. The joystick potentiometers are connected between the 3.3 VDC supply and ground and the arm of the pot is connected to an analog input. As the joysticks are moved, the voltages at the analog inputs varies and with that the software can tell which direction the joystick is pointing or if it is active at all. Although connected and configured, the joystick switches are not currently used in the MIDI Buddy software.

Most of the parts used in the MIDI Buddy are available from trusted sources including Adafruit, SparkFun, PJRC.com, Digikey, Jameco, etc. The Fluxamasynth shield I believe is only available from a single source. See Resources section for vendor information. If you don't mind a longer wait, many of the parts are available on eBay from Chinese sources. Note, that while the prices on eBay are great, the quality can be hit or miss. One example. I looked at all the usual suppliers for the SD card breakout board used in the MIDI Buddy and the least expensive one I could find was around $10 plus shipping. Instead I ended up buying six of them on eBay for $0.99 each with free shipping and they worked perfectly.

I bought a game controller with two joysticks at a local Goodwill store for $3.50 and extracted the joysticks for use in MIDI Buddy. Of course you can buy these new, but why?

I would suggest the Teensy 3.1 controller be purchased directly from Paul Stoffregen of pjrc.com who designed it and rigorously supports it. Paul has worked hard making the Teensy

usable in the Arduino Integrated Development Environment (IDE) and has ported many of the most popular Arduino libraries to the Teensy. This means you can develop code in the Arduino IDE and download it via USB to the Teensy just as you would if you were using an Arduino. He calls his adaptation of the Teensy into the Arduino environment, Teensyduino. All MIDI Buddy software was developed in the Teensyduino environment.

Part numbers listed on the schematic with an AF prefix are Adafruit part numbers. Parts with SF prefix are SparkFun part numbers.

Software

MIDI Buddy software is available as a zip file from the Nuts and Volts website. The table below will give you an idea of what each file contains.

Filename	Function
Demos.h/Demos.cpp	File containing the code for the four MIDI Buddy demos. These demo were ported from the Modern Device library for the SAM-2195 chip.
FileFunctions.h/FileFunctions.cpp	Caches the names of up to 30 MIDI SM files on the SD card. The limit of 30 was picked out of the air and can easily be changed if access to more files is required.
MIDIBuddy.ino	Main MIDI Buddy sketch
MIDIInfo.h	Header file with voicing names and effect parameter names. Used in the user interface.
MIDIParser.h/MIDIParser.cpp	A parser for type 0 Standard MIDI Files.
MIDIPlayer.h/MIDIPlayer.cpp	Code which takes events generated by the MIDIParser while reading MIDI files and sends them to the SAM-2195 for rendering
SAM2195.h/SAM2195.cpp	An API for controlling the SAM-2195 synthesizer chip
UISMFunctions.h/UISMFunctions.cpp	User interface state machines implementing the MIDI Buddy functionality.
Utils.h/Utils.cpp	Various printing and logging functions in addition to functions for polling and reading the joysticks.

The functions provided in the current MIDI Buddy software are described below.

Menu Entry	Function
Play MIDI File	Allows user to browse the available SMFs on the SD card and select one for playback.
Record Riff	Allows user to record a riff played on an

	attached MIDI device and save it on the SD card.
Play Riff	Allows user to playback a previously recorded riff from the SD card
Master Volume	Allows user to set the master volume level of the synthesizer
Drum Set	Allows the user to select which of the four drum sets/drum kits is to be used on MIDI channel 10
Synth Reset	Resets the SAM-2195 chip to its power on state
Chan Volume	Allows the user to set the volume of individual MIDI channels
Chan Program	Allows the user to set the voicing of individual MIDI channels.
Chan Pan	Allows the user to determine where in the stereo sound field a channels' voice should be heard.
Chan Chorus	Allows the user to configure chorus on a channel by channel basis.
Chan Reverb	Allows the user to configure reverb on a channel by channel basis.
Chan Portamento	Allows the user to configure portamento on a channel by channel basis. Portamento is the glide between notes. From none to very long.
Chan Modulation	Allows the user to configure modulation (tremolo) on a channel by channel basis.
D1: All Voices	Runs demo one
D2: Note Bending	Runs demo two
D3: Cross Fade	Runs demo three
D4: Drum Pattern	Runs demo four

Most of MIDI Buddy's software is straightforward but how the user interface (UI) is implemented is somewhat unconventional because it was very important to allow the user to hear changes they make in real time. Keeping the UI responsive while at the same time allowing MIDI data to be received and rendered is a challenge without a multitasking operating system. This was accomplished using a series of nested finite state machines which run continually. Anytime the UI is waiting on the user for joystick input, control is returned to the main loop so that MIDI data can flow without interruption. As soon as user

input is detected, the state of the UI state machine is changed and the process starts over. See the loop() function in MIDIBuddy.ino to see how this is done.

MIDI Buddy requires the LiquidCrystal and the SdFat libraries to compile and run. The LiquidCrystal library comes preinstalled with the Arduino IDE but the SdFat library must be installed. See Resources.

The SD card must be formatted with FAT16 format before any standard MIDI files can be stored on it. Filenames must in 8.3 format; that is a maximum of eight characters in the filename and up to three characters in the filename extension. Remember only single channel type 0 MIDI files (the most common type) can be played by MIDI Buddy.

Operation and Navigation

I pondered many different user interface schemes for MIDI Buddy but finally settled on using two joystick largely because of the cool factor. The left joystick is used to select a function for MIDI Buddy to perform. Clicking the left joystick up or down scrolls through the possible functions (enumerated above) which are shown on the LCD. Clicking the left joystick to the right selects the function. Clicking the left joystick left does nothing.

Once a function is selected, the right joystick become operational. You use the right joystick to increment and decrement values and to select additional parameters needing configuration. If you click the right joystick to the left, you immediately backup one menu level. As a simple example, click the left joystick up or down until Master Volume is shown on the LCD. Click it right to select this function. With the right joystick, move up and down to set the desired volume which will be shown on the LCD. You will be able to hear the volume changes in real time if you have a MIDI source plugged in and are listening to MIDI Buddy's output. Once you arrive at the volume level you want, click the right joystick to the right and the volume will be set and control will return to the top level MIDI Buddy menu.

Conclusions

Learning about MIDI has been fun and enlightening and it has also shown that old dogs can still learn new tricks. Have fun building a MIDI Buddy of your own and jam on !

Resources

The following sources provide more information about this project and its background.

Information about getting started with the Arduino can be found at: http://arduino.cc/en/Guide/HomePage. The Arduino IDE can be downloaded from: http://arduino.cc/en/main/software. Note version 1.0.5 or newer of the IDE is required for use with the MIDI Buddy.

For information about and/or purchase of the Teensy 3.1 micro controller board go to: www.pjrc.com. There is also a forum for information about the Teensy line of micro controllers at: http://forum.pjrc.com/forum.php. Questions regarding the Teensy should be directed there.

The Teensyduino (version 1.18 or newer) software development add on to the Arduino IDE is

available at: www.pjrc.com/teensy/teensyduino.html as are instructions for installing it.

The Fluxamasynth shield containing the SAM-2195 synthesizer chip is available from Modern Device. Info is at: moderndevice.com/product/fluxamasynth-shield/.

A datasheet for the SAM2195 chip is available at: http://serdaco.com/files/SAM2195.pdf

The SdFat library is available at: https://github.com/greiman/SdFat.

For background information about digital audio see my book: “Digital Audio with Java”, published by John Wiley & Sons.

Other micro controller based projects of mine can be found at: www.craigandheather.net/celepage.html.

Photo One

The Completed and Packaged MIDI Buddy

Photo Two

Top View of MIDI Buddy Breadboard

The Teensy 3.1 controller is in the upper left.

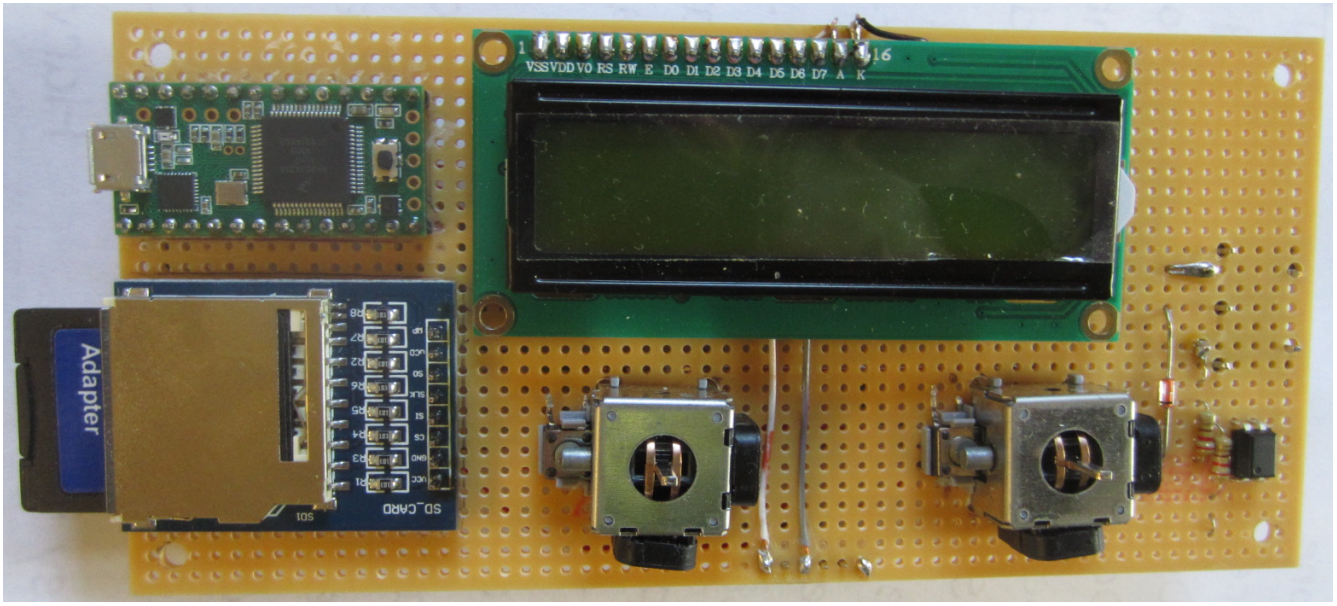

Photo Three

Bottom View of MIDI Buddy Breadboard

LCD Contrast Trimmer is gray component at top

MIDI connector at left

Fluxamasynth shield in middle

NOTE: a jumper is soldered to the pin 1 select on the shield and can be seen in this image.

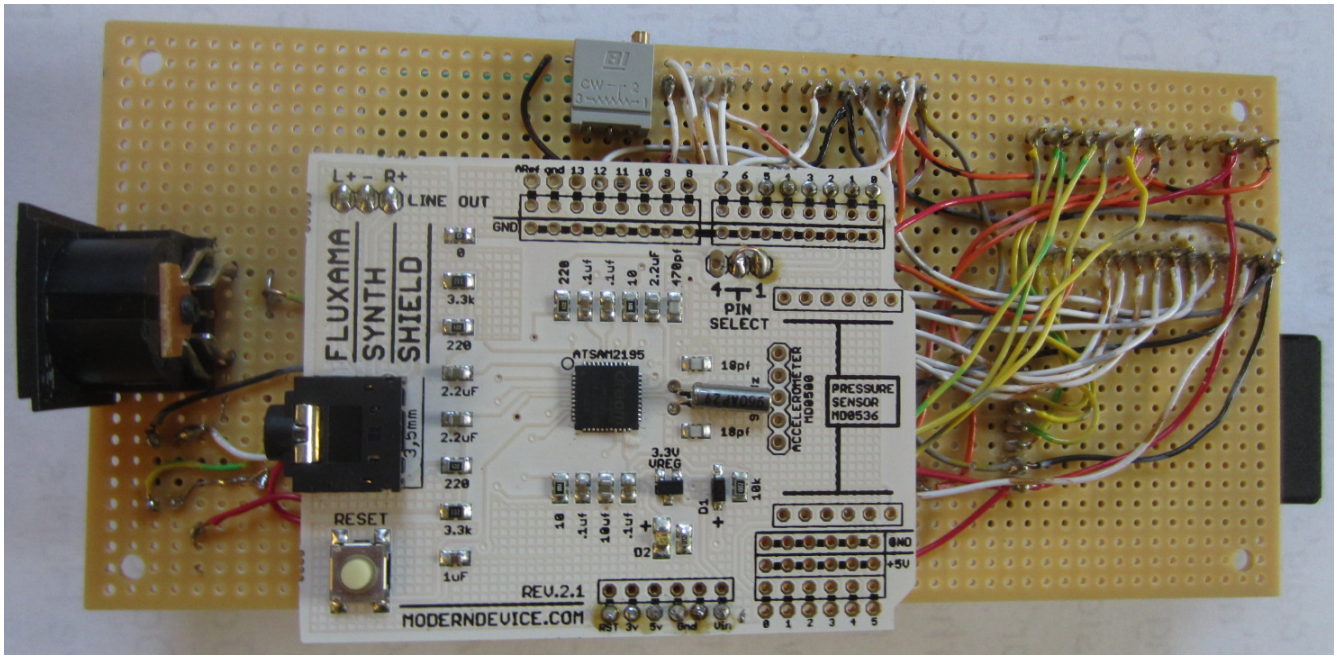

Photo Four

Access to Teensy 3.1 USB Connector and SD Card

Photo Five

MIDI Input and Stereo Audio Output

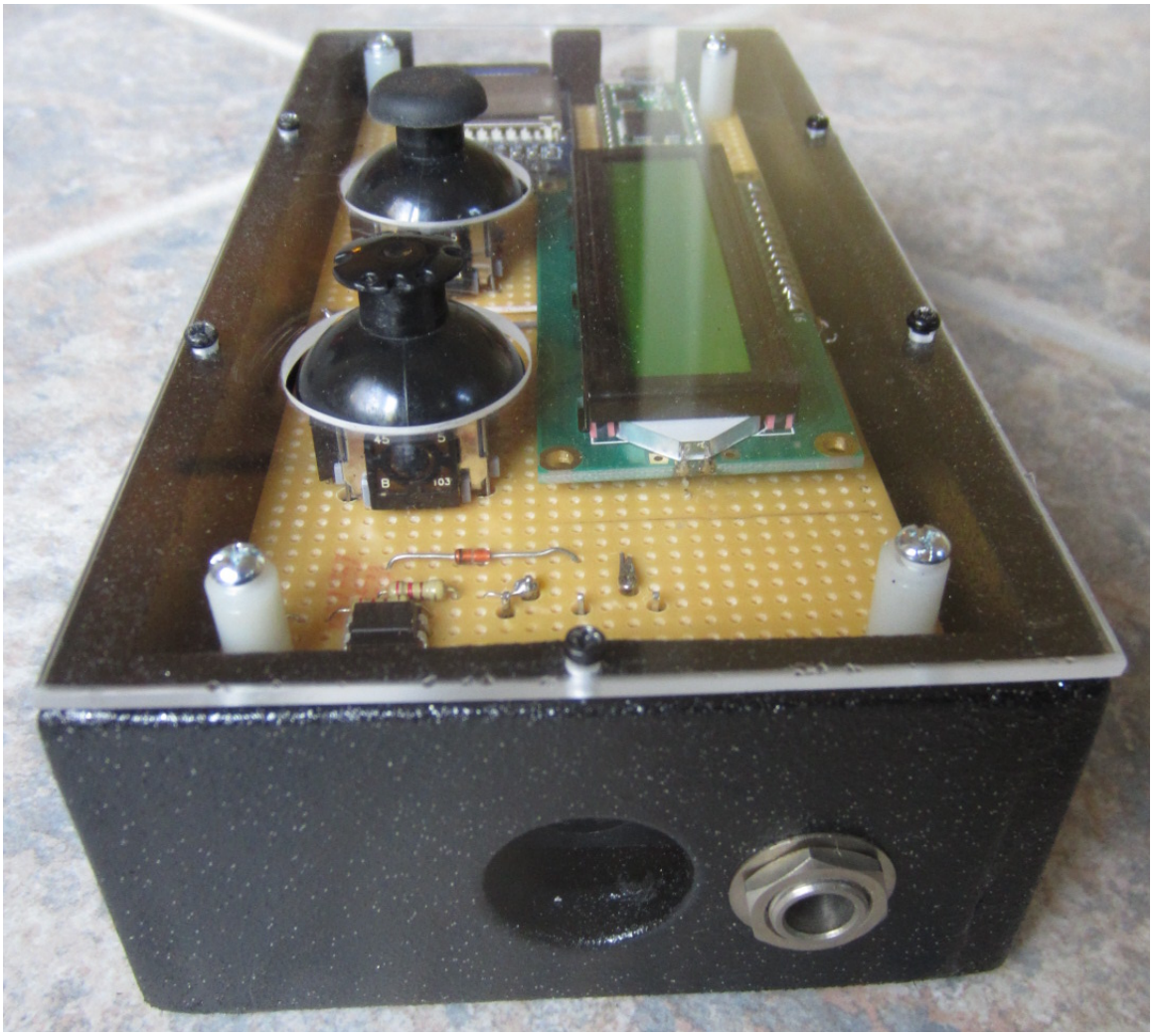

Figure One

MIDI Buddy Schematic