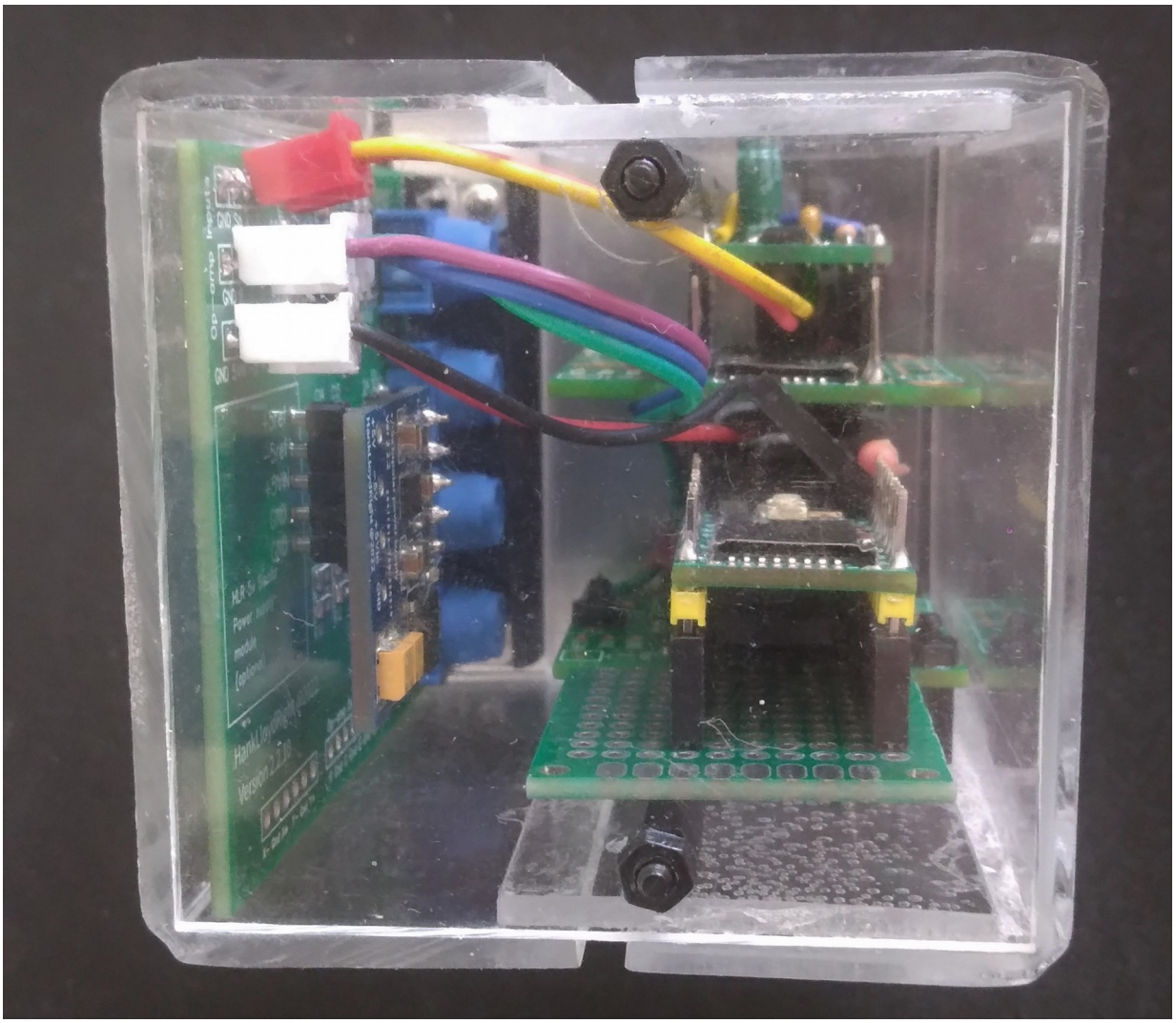
The Hermit's

T4 Laser Synth Alpha

Hardware Assembly Manual



7/15/22

Teensy 4.1 MCU

Why Teensy 4.1, instead of an Arduino, ESP32, or SM32 MCU's?

The T4.x is a rock star among MCU's, but on steroids. The ARM Cortex-M7 processor, with 8MB Flash RAM running @600mHz is incredible. As shown below, the T4.1 board has plenty of GPIO pins to support lots of peripheral devices.
The T4.1 runs on PJRC's unique, yet full featired Teensyduino C++ Library that is fully compatible with the Arduino IDE, as well as Platform IO. So, it is a perfect choice for makers, hobbyists, and laser enthusiasts.

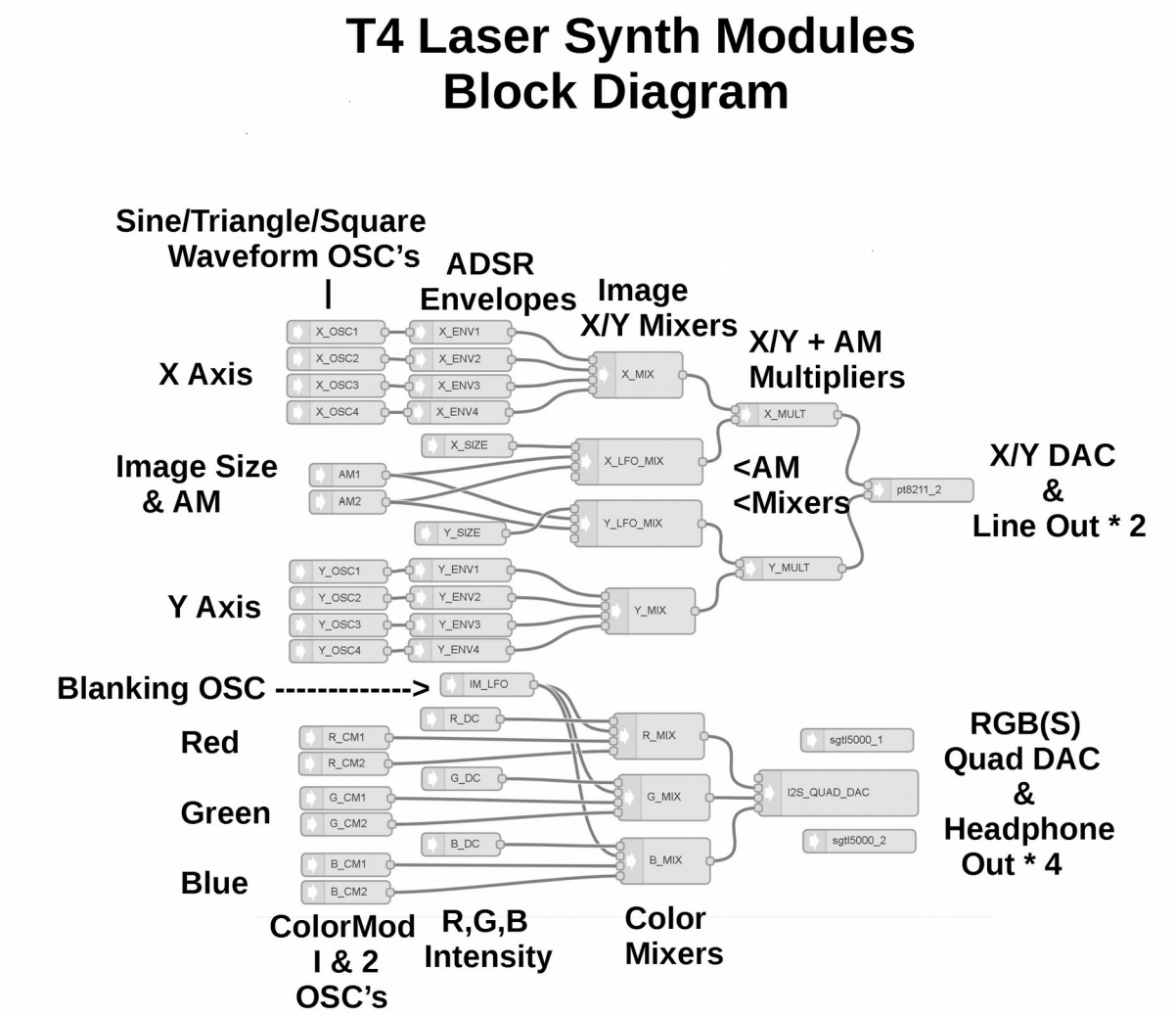
Specifications

- •ARM Cortex-M7 at 600 MHz
- •Float point math unit, 64 & 32 bits
- •7936K Flash, 1024K RAM (512K tightly coupled), 4K EEPROM (emulated)
- •QSPI memory expansion, locations for 2 extra RAM or Flash chips
- •USB device 480 Mbit/sec & USB host 480 Mbit/sec
- •55 digital input/output pins, 35 PWM output pins
- •18 analog input pins
- •8 serial, 3 SPI, 3 I2C ports
- •2 I2S/TDM and 1 S/PDIF digital audio port
- •3 CAN Bus (1 with CAN FD)
- •1 SDIO (4 bit) native SD Card port
- •Ethernet 10/100 Mbit with DP83825 PHY
- •32 general purpose DMA channels
- •Cryptographic Acceleration & Random Number Generator
- •RTC for date/time
- •Programmable FlexIO
- •Pixel Processing Pipeline
- •Peripheral cross triggering
- •Power On/Off management

Feature	Teensy 4.1
Ethernet	10 / 100 Mbit DP83825 PHY (6 pins)
USB Host	5 Pins with power management
SDIO (4 bit data)	Micro SD Socket
PWM Pins	35
Analog Inputs	18
Serial Ports	8
Flash Memory	8 Mbyte
QSPI Memory	2 chips + Program Memory
Breadboard I/O	42
Bottom SMT Pads	7
SD Card Signals	6
Total I/O Pins	55

PJRC Audio Design Tool

Teensyduino includes a full featured, well documented Audio Library, with code examples available within the Arduino IDE. PJRC also has a very welcoming online forum to provide current detailed resources, as well.
There is also an excellent web based PJRC Audio Design Tool, very similar to Code Red. simply browse through the list of modules on the left panel and drag whatever in/out synth modules your heart desires onto the design window, then connect them together with a few mouse clicks.
The following image is a screenshot of the PJRC Design Tool's window, after being populated with the T4 Laser Synth's modules.



* 4 Quadrature OSC pairs, each w/ADSR to avoid galvo snap upon preset change
* 2 sine/triangle/square AM LFOs mixed into master multipliers
* sine/triangle/square Intensity modulation
* 2 sine/triangle/square Color Modulation OSCs w/ 0. 120, & 240 degree RGB phase shifts
* all oscillators and LFO's have indepentent control of:
    waveform select, 0-~2kHz frequency, amplitude, phase, pulsewidth, & offset
* all waveforms are routed through mixer channels for post signal mixing
* All parameters are controllable via usbMIDI interface

PJRC Audio Design Tool's Exported Code
in Adruino IDE

The Audio Design Tool has **Import** & **Export** Buttons to convert between the graphical design and the code that is eventually loaded onto the T4 MCU. The following is the code that was generated by the Design Tool, copied from the Export window, then pasted directly into my sketch. This code can be copied and Imported into the Design Tool to reconfigure the modules, then Exported back into the same sketch:

```
#include <Audio.h>
#include <Wire.h>
#include <SPI.h>
#include <SD.h>
#include <SerialFlash.h>

// GUItool: begin automatically generated code
AudioSynthWaveform       X_OSC1;         //xy=79.0035171508789,22.999999046325684
AudioSynthWaveform       X_OSC2;         //xy=79.0035171508789,57.999999046325684
AudioSynthWaveform       X_OSC3;         //xy=79.0035171508789,92.99999904632568
AudioSynthWaveform       X_OSC4;         //xy=79.0035171508789,127.99999904632568
AudioSynthWaveform       Y_OSC1;         //xy=79.0035171508789,163.99999904632568
AudioSynthWaveform       Y_OSC2;         //xy=79.0035171508789,198.99999904632568
AudioSynthWaveform       Y_OSC3;         //xy=79.0035171508789,233.99999904632568
AudioSynthWaveform       Y_OSC4;         //xy=79.0035171508789,267.9999990463257
AudioSynthWaveform       R_CM2;          //xy=83.0035171508789,412.9999990463257
AudioSynthWaveform       IM_LFO;         //xy=84.0035171508789,307.9999990463257
AudioSynthWaveform       R_CM1;          //xy=84.0035171508789,378.9999990463257
AudioSynthWaveform       G_CM1;          //xy=84.0035171508789,479.9999990463257
AudioSynthWaveform       G_CM2;          //xy=85.0035171508789,513.9999990463257
AudioSynthWaveform       B_CM1;          //xy=86.0035171508789,581.9999990463257
AudioSynthWaveform       B_CM2;          //xy=86.0035171508789,617.9999990463257
AudioSynthWaveformDc     R_DC;           //xy=89.0035171508789,342.9999990463257
AudioSynthWaveformDc     G_DC;           //xy=91.0035171508789,445.9999990463257
AudioSynthWaveformDc     B_DC;           //xy=93.0035171508789,546.9999990463257
AudioEffectEnvelope      Y_ENV4;         //xy=237.0035171508789,266.9999990463257
AudioEffectEnvelope      Y_ENV1;         //xy=238.0035171508789,161.99999904632568
AudioEffectEnvelope      X_ENV4;         //xy=239.0035171508789,127.99999904632568
AudioEffectEnvelope      Y_ENV2;         //xy=239.0035171508789,197.99999904632568
AudioEffectEnvelope      Y_ENV3;         //xy=239.0035171508789,233.99999904632568
AudioEffectEnvelope      X_ENV3;         //xy=241.0035171508789,93.99999904632568
AudioEffectEnvelope      X_ENV1;         //xy=242.0035171508789,22.999999046325684
AudioEffectEnvelope      X_ENV2;         //xy=242.0035171508789,58.999999046325684
AudioMixer4              B_MIX;          //xy=263.0035171508789,576.9999990463257
AudioMixer4              R_MIX;          //xy=268.0035171508789,371.9999990463257
AudioMixer4              G_MIX;          //xy=268.0035171508789,473.9999990463257
AudioSynthWaveform       AM1;            //xy=421.4335021972656,129.85888671875
AudioSynthWaveform       AM2;            //xy=421.4334945678711,164.14454650878906
AudioMixer4              Y_MIX;          //xy=426.1463279724121,248.57140350341797
AudioMixer4              X_MIX;          //xy=438.14634704589844,43.71428108215332
AudioSynthWaveformDc     X_SIZE;         //xy=444.2907371520996,94.14457130432129
AudioSynthWaveformDc     Y_SIZE;         //xy=445.71921157836914,198.43028926849365
AudioOutputI2SQuad       I2S_QUAD_DAC;   //xy=499.0035171508789,468.9999990463257
AudioMixer4              X_LFO_MIX;      //xy=618.5764427185059,85.5731397356306
AudioMixer4              Y_LFO_MIX;      //xy=622.8621406555176,202.87471771240234
AudioEffectMultiply      X_MULT;         //xy=798.5763816833496,47.00175094604492
AudioEffectMultiply      Y_MULT;         //xy=798.5763893127441,239.85883045196533
AudioOutputPT8211_2      pt8211_2;       //xy=928.289192199707,145.42855072021484
AudioConnection          patchCord1(X_OSC1, X_ENV1);
AudioConnection          patchCord2(X_OSC2, X_ENV2);
AudioConnection          patchCord3(X_OSC3, X_ENV3);
AudioConnection          patchCord4(X_OSC4, X_ENV4);
AudioConnection          patchCord5(Y_OSC1, Y_ENV1);
```

```
AudioConnection          patchCord6(Y_OSC2, Y_ENV2);
AudioConnection          patchCord7(Y_OSC3, Y_ENV3);
AudioConnection          patchCord8(Y_OSC4, Y_ENV4);
AudioConnection          patchCord9(R_CM2, 0, R_MIX, 3);
AudioConnection          patchCord10(IM_LFO, 0, R_MIX, 1);
AudioConnection          patchCord11(IM_LFO, 0, G_MIX, 1);
AudioConnection          patchCord12(IM_LFO, 0, B_MIX, 1);
AudioConnection          patchCord13(R_CM1, 0, R_MIX, 2);
AudioConnection          patchCord14(G_CM1, 0, G_MIX, 2);
AudioConnection          patchCord15(G_CM2, 0, G_MIX, 3);
AudioConnection          patchCord16(B_CM1, 0, B_MIX, 2);
AudioConnection          patchCord17(B_CM2, 0, B_MIX, 3);
AudioConnection          patchCord18(R_DC, 0, R_MIX, 0);
AudioConnection          patchCord19(G_DC, 0, G_MIX, 0);
AudioConnection          patchCord20(B_DC, 0, B_MIX, 0);
AudioConnection          patchCord21(Y_ENV4, 0, Y_MIX, 3);
AudioConnection          patchCord22(Y_ENV1, 0, Y_MIX, 0);
AudioConnection          patchCord23(X_ENV4, 0, X_MIX, 3);
AudioConnection          patchCord24(Y_ENV2, 0, Y_MIX, 1);
AudioConnection          patchCord25(Y_ENV3, 0, Y_MIX, 2);
AudioConnection          patchCord26(X_ENV3, 0, X_MIX, 2);
AudioConnection          patchCord27(X_ENV1, 0, X_MIX, 0);
AudioConnection          patchCord28(X_ENV2, 0, X_MIX, 1);
AudioConnection          patchCord29(B_MIX, 0, I2S_QUAD_DAC, 3);
AudioConnection          patchCord30(R_MIX, 0, I2S_QUAD_DAC, 1);
AudioConnection          patchCord31(G_MIX, 0, I2S_QUAD_DAC, 2);
AudioConnection          patchCord32(AM1, 0, X_LFO_MIX, 1);
AudioConnection          patchCord33(AM1, 0, Y_LFO_MIX, 1);
AudioConnection          patchCord34(AM2, 0, X_LFO_MIX, 2);
AudioConnection          patchCord35(AM2, 0, Y_LFO_MIX, 2);
AudioConnection          patchCord36(Y_MIX, 0, Y_MULT, 1);
AudioConnection          patchCord37(X_MIX, 0, X_MULT, 0);
AudioConnection          patchCord38(X_SIZE, 0, X_LFO_MIX, 0);
AudioConnection          patchCord39(Y_SIZE, 0, Y_LFO_MIX, 0);
AudioConnection          patchCord40(X_LFO_MIX, 0, X_MULT, 1);
AudioConnection          patchCord41(Y_LFO_MIX, 0, Y_MULT, 0);
AudioConnection          patchCord42(X_MULT, 0, pt8211_2, 0);
AudioConnection          patchCord43(Y_MULT, 0, pt8211_2, 1);
AudioControlSGTL5000     sgtl5000_1;     //xy=502.0035171508789,374.9999990463257
AudioControlSGTL5000     sgtl5000_2;     //xy=503.0035171508789,412.9999990463257
// GUItool: end automatically generated code
```

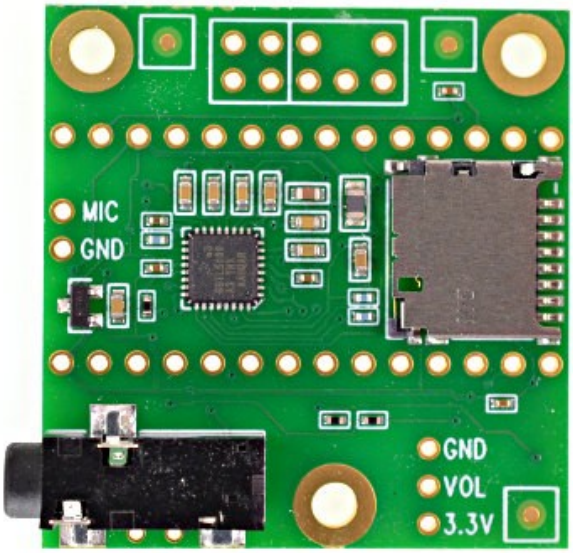
Just look at all of those lines of code that don't need to be manually entered! How cool is that?!

The SGTL5000 Quad DAC Audio Shields are enabled under the void setup() function by adding the following lines of code.

```
void setup() {
  AudioMemory(20);
  usbMIDI.setHandleControlChange(Control_Change);
  usbMIDI.setHandleNoteOff(Note_Off);
  usbMIDI.setHandleNoteOn(Note_On);
  sgtl5000_1.setAddress(LOW);
  sgtl5000_1.enable();
  sgtl5000_1.unmuteHeadphone();
  sgtl5000_1.volume(.8);
  sgtl5000_1.muteLineout();
  sgtl5000_2.setAddress(HIGH);
  sgtl5000_2.enable();
  sgtl5000_2.unmuteHeadphone();
  sgtl5000_2.volume(.8);
  sgtl5000_2.muteLineout();
```

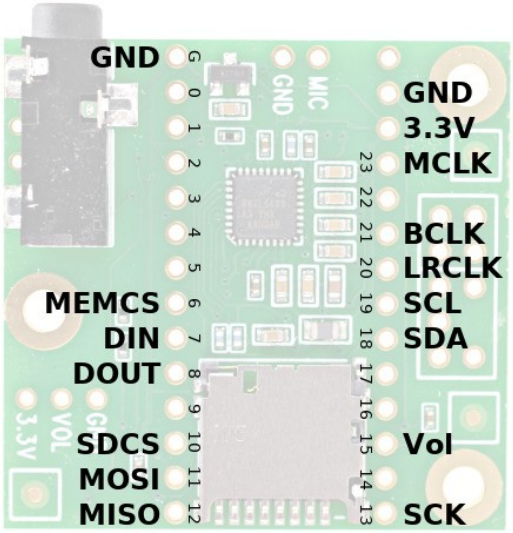
Voila'! Your T4 Laser Synth software modules are now at your command!

T4.x Quad I2S SGTL5000 DAC Audio Shields



As illustrated in the Design Tool's block diagram, the RGB signals feed into an output module labeled 'I2S_Quad_DAC', which is flanked by 'SGTL5000_1' & 'SGTL5000_2' which have no patch connections. The I2S_Quad_DAC is a software module that outputs the CD quality, I2S, 16 bit digital DIN signals to two of the Teensy's GPIO pins, one for each SGTL5000 audio shield, which have DACs to convert the signals into analog audio.

A few more GPIO pins are required for CLK, MCLK, 3.3VDC, & Gnd. The image on the right shows the default connections for the 1st SGTL5000 Audio Shield.



Stackable header pins can be used to make these connections by piggy-backing the SGTL5000 audio shields and T4.1. Note: The pin numbers are the same as the T4.1's

At this point, we should have a Teensy 4.1 connected to the SGTL5000 audio shield, which is capable of being programmed to produce X/Y imagery on an oscilloscope screen, without additional hardware nor mods.

IOW, at this point, all of the software development can be accomplished for testing, synthesizing, and controlling cycloids; mapping sketch functions to MIDI device controls; creating a virtual MIDI GUI control pannel with Pure Data; and interfacing with Cakewalk's DAW.

Proof of concept can be 80% achieved without the need of an expensive laser projector. Alternatively, The T4 Laser Synth + a DIY ILDA conversion op amp circuit can easily interface with cheap components available online.

Adding 2 more audio shields for RGB output signals to laser diode driver boards is our next objective.

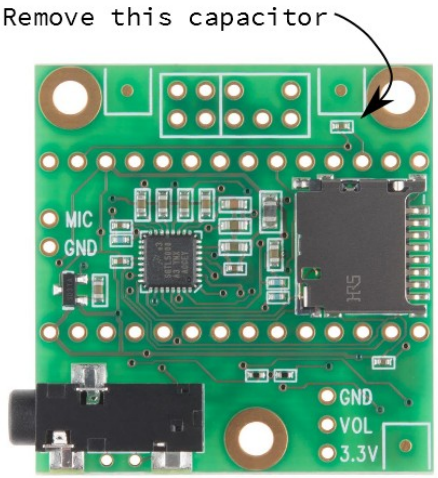
2nd T4.x I2S SGTL5000 DAC Audio Shield Quad Mods

By default, the SGTL5000 Audio shields are configured to output L & R audio signals. So, the 2nd SGTL5000 Audio shield needs a few modifications to change the I2C address and to use GPIO pin 32, for DIN, instead of the default GPIO 7, which is already being utilized on the 1st SGTL5000 for audio channels 1 & 2. These mods are as follows:

(ref:https://www.sparkfun.com/news/2055 w/T4.0 Teensy)

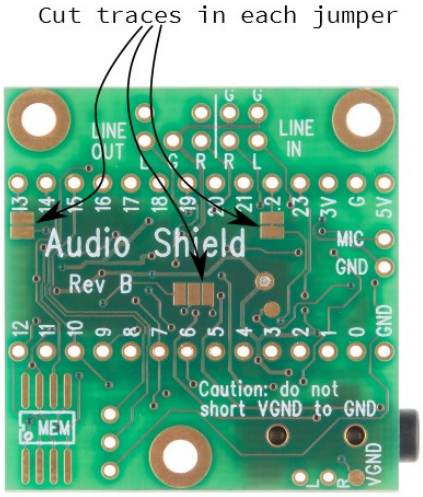
To use two Audio Adapters, you'll need to modify the boards to coexist.

First, you'll need to remove the 0.1 uF capacitor on Pin 15. That's ADC input A1 (or digital I/O 15), which will be repurposed as the TXD pin of the second audio channel.



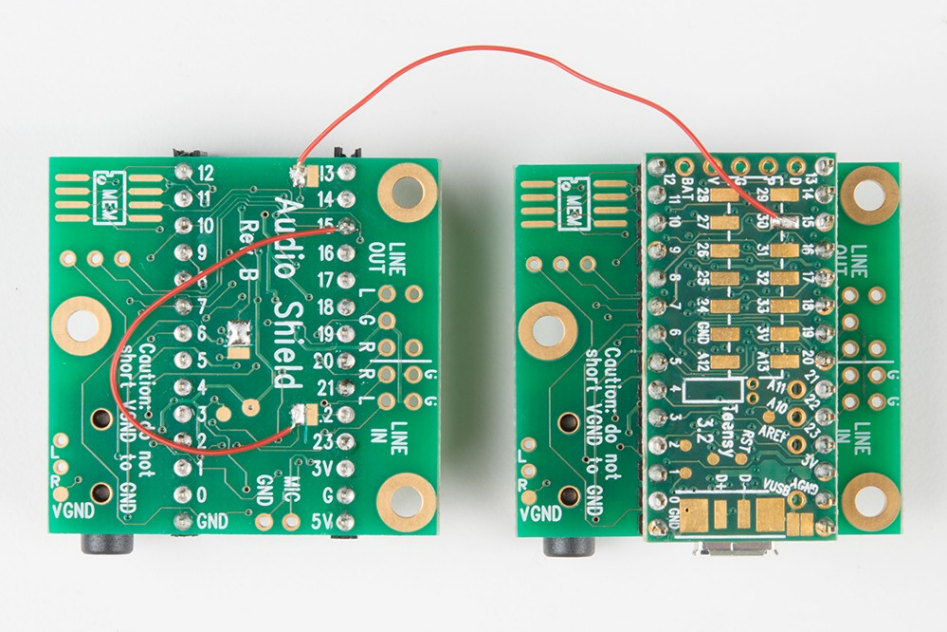
Remove this cap from both boards.

Then, you'll need to select one board to be channels 1 and 2, and the other to be channels 3 and 4. The 3 and 4 board needs some extra cuts and jumps. The pairs of pads near pins 13 and 22 need to be cut apart. The trace between the center and left pads of the I2C address selector also need to be cut.Cuts for channels 3 and 4.



Then, new connections need to be wired to those pads.

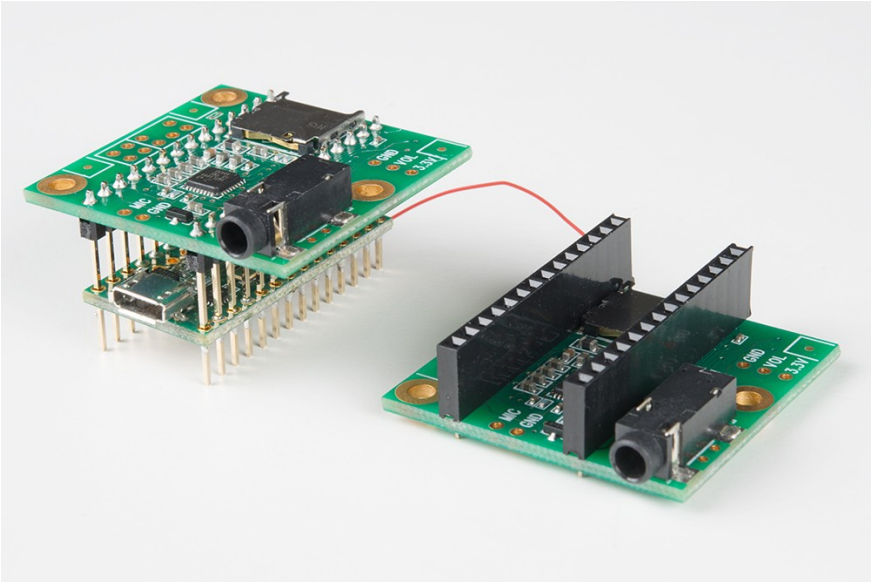
- The first two jumpers configure to codec to read and write alternate channels on the I2S peripheral.
    - From the inner pad near pin 22, run a wire to pin 15.
    - From the inner pad near pin 13, run a wire to pad 30 on the bottom of the Teensy.
- Use a solder blob to connect the center and left pads of the three-way jumper. This assigns the board to its secondary I2C address.



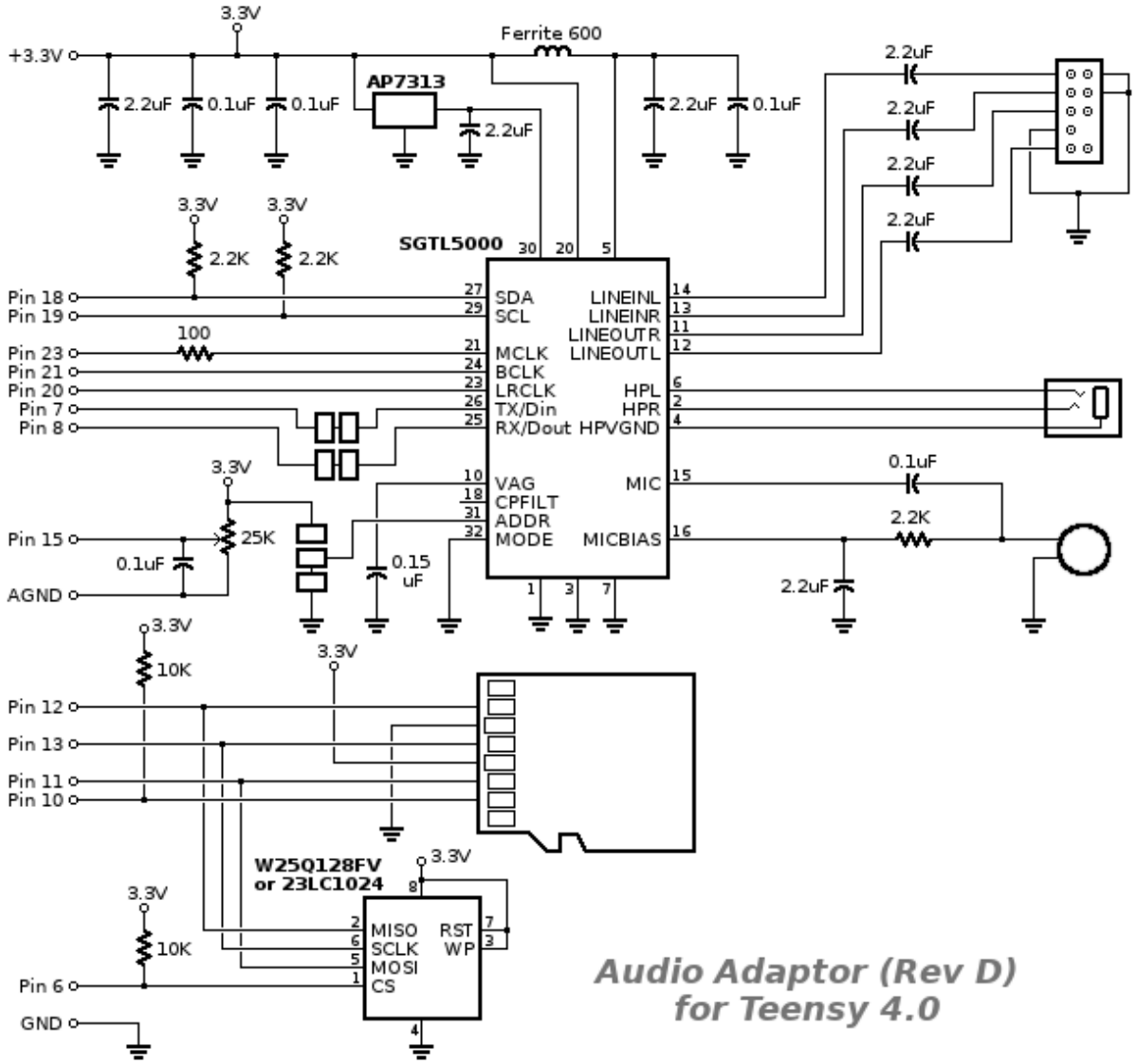
Connecting the channel 3 and 4 data lines.

Last, add the headers. For my stack, I put long pin headers on the bottom of the first Audio Adapter, then ran them through the Teensy and soldered them in place with the Teensy about halfway up the pins. This leaves the pads on the bottom of the Teensy accessible for future expansion. The other Audio Adapter got female headers on top of the board.

When you plug the boards into each other, make sure the orientation is correct -- the headphone connectors face the same way.



T4.x SGTL5000 Audio Shield Schematic



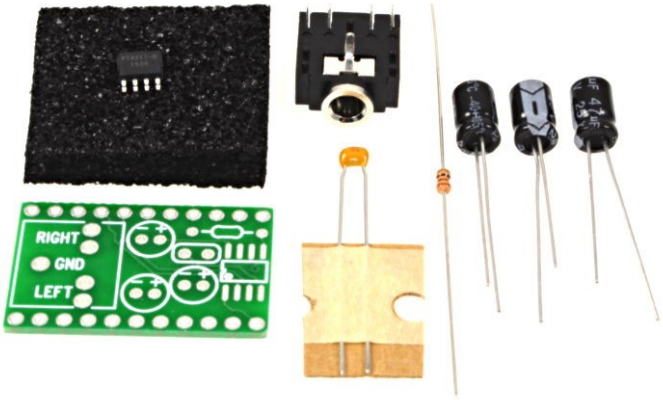
[Kicad Symbol & Footprint]

The schematic above illustrates the connections in and out of the SGTL5000 Audio CODEC IC, including the previously modifiable soldering pads and I2C addresses. Note that there is also a uSD card slot, which he T4.1 already has. There are also mic inputs, line inputs, and line outputs that have 2.2uF capacitors to eliminate DC offset signals for audio applications, but unwanted for controlling lasers.

At the moment, we're only interested in using the DC coupled Headphone outputs, plus the I2S signals, of course. All the rest is fluff. 14 connections to both SGTL5000 IC's is all that's being used.

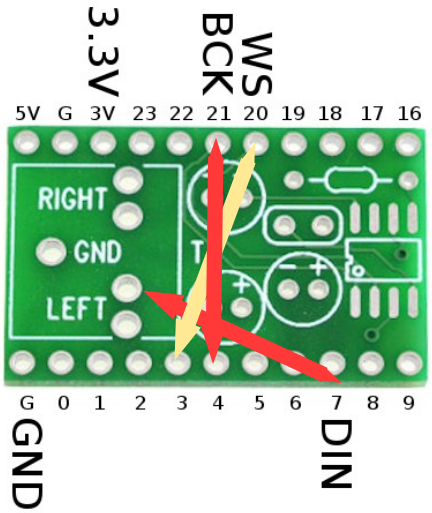
PT8211 I2S Stereo Audio Shield

The I2C solder pads on the SGTL5000 Audio Shields are limited to only 2 I2C addresses which they require to toggle I2S data between the 2 shields. Therefore, audio output channels 5 & 6 require a different I2S Audio shield.
The PT8211 doesn't require I2C at all and PJRC sells a kit with the PT8211 I2S DAC IC, a breakout PCB, plus the necessary caps and a resistor.

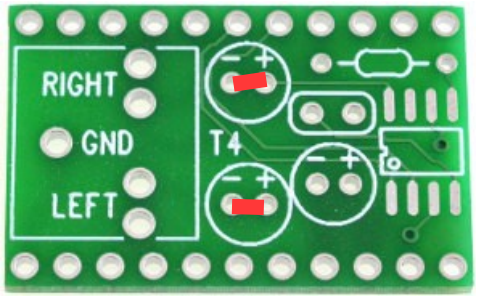


Link to: [Assembly instructions on PJRC's product page]
...Ummm. Except for the following mods, of course:
[PT8211 Datasheet]

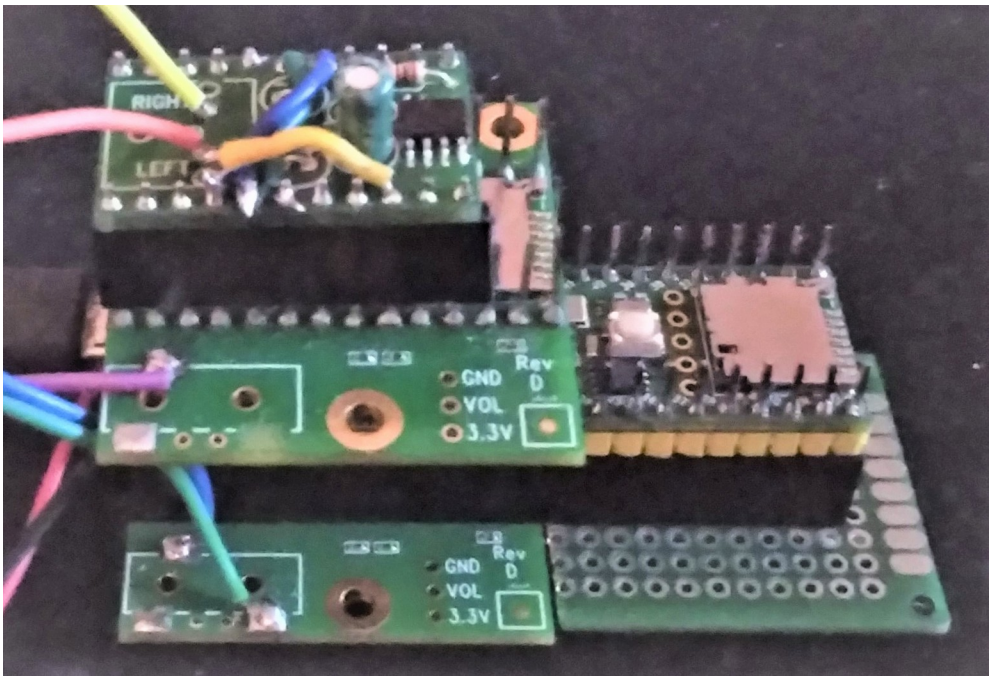
By default, the PT8211 is designed to input the I2S DIN signal from GPIO pin 7. But, we've already used pin 7 for IS channels 1 & 2. So, the DIN input needs to connect to the T4 GPIO pin #2, instead. Likewise, the WS input needs to be redirected to T4 GPIO pin #3 and BCK is redirected to T4 GPIO pin #4.



The final mod is made while assembling the kit. The 2 capacitors above and below the 'T4' silkscreen label need to be discarded and replace with wire jumpers, connecting + & - together ti provide a DC coupled audio output.
Using stackable headers with GPIO pins 7, 20, & 21 removed will allow the PT8211 to be stacked on top of the gTL5000 audio shields, as the T4 Laser Synth Alpha prototype is configured.

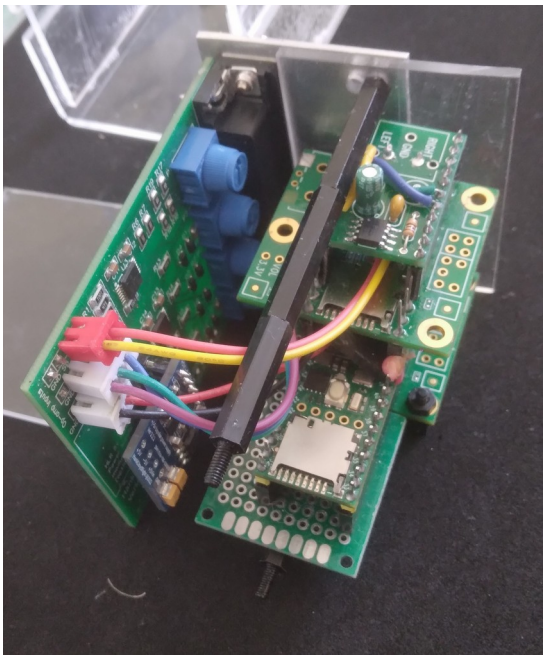


T4 Laser Synth Alpha
with Stacked Audio Shields



The I2S mods for channel 5 & 6 on the PT8211 audio shield can be seen with the green, blue, and yellow wire jumpers, getting I2S signals from T4 pins # 2,3,4 (skipping Gnd & #0). The green and blue wires are running across where the coupling capacitors would normally live, without the DC coupling jumpers.
Note the unused GPIO pins along the back of the T4.1. Model II of the T4 Laser Synth will be installed inside a RGB lumia projector and utilize those GPIO pins for motor speed, RGB intensities, and a servo controlled optical effect turret.

But, we still haven't connected the T4 Laser Synth to a laser projector. For that, we need to add a correction/amplifier circuit. The audio shields output signals that range between 0v and +3.2v peak to peak. That means the signal and image will be offset to the midpoint of +1.6 VDC. So, that needs to be corrected with an op amp circuit, which is also capable of boosting the signal to -/+ 5 V p-p.



Not being an EE, I chose the quick,easy, and more expensive path of purchasing an ILDA standard conversion board from HankLloydRight, a member of PLF.

The photo on the right shows the orange/pink X/Y signals and RGB signals being connected to the ILDA conversion amp. **So, let's get lasing!**

<insert Akai ACP40 MIDI controller pics & info>

<insert PureData Surface 10” Master touchpanel GUI pics & info>

<insert PureData 15” touchscreen GUI pics & info>

<Summarize with phase II plans to consolidate audio shields & ILDA amps to PCB>