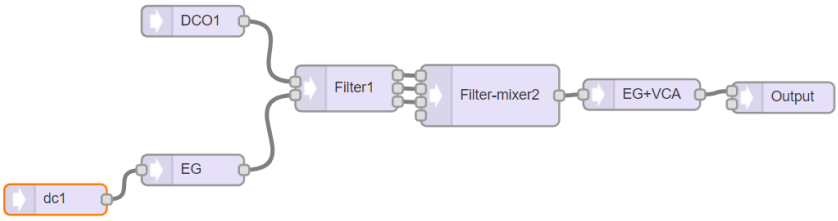
Add your Plugin Checklist

- [ ] .) Think yourself of a cool Plugin
- [ ] .) Implement the modules into the Teensy-Touch-DAW Audiochain.
- [ ] .) Export the new AudioChain and replace it with the existing one
- [ ] .) Add the 6 Gainstages to the Array of pointers
- [ ] .) Place your control variables in variables.ino.
- [ ] .) Add these 7 functions to your new plugin *.ino file
- [ ] .) “void Plugin_3_Settings()”
- [ ] .) “void Plugin3_Control()”
- [ ] .) “void Plugin3_Page1_Dynamic()”
- [ ] .) “void Plugin3_Page_Static(byte Pagenumber)”
- [ ] .) “void Plugin3_Change()”
- [ ] .) “void savePlugin3()”
- [ ] .) “void loadPlugin3()”

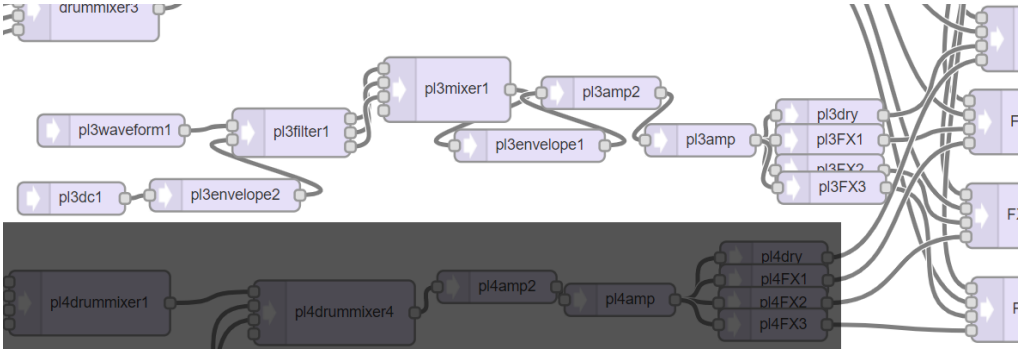
☐ .) Think yourself of a cool Plugin

and what controls you want to have

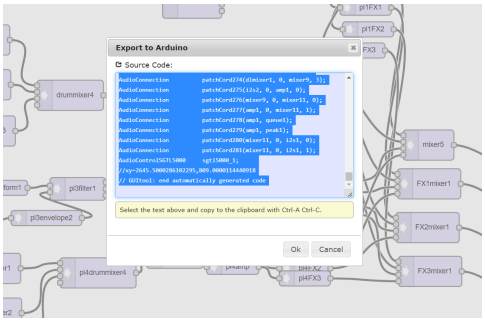


☐ .) Implement the modules into the Teensy-Touch-DAW Audiochain.

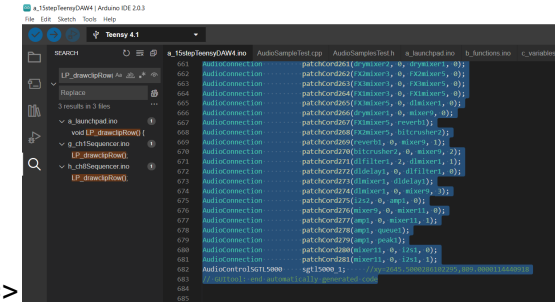
Copy the existing AudioChain-Code into Newdigate´s AudioTool. Paste your Plugin inside. Don't forget the 2 Gain stages for MAX Volume and Volume/bar
as well as the 4 Gain modules for the FX´s



☐ .) Export the new AudioChain and replace it with the existing one



=>



☐ .) *Add the 6 Gainstages to the Array of pointers

* If we have reached the max amount of Plugins, i´ve made a template for, we have to add new Gain stages. But there is most likely also other stuff to do

```
AudioAmplifier *gainmax[MAX_PLUGINS]{ &pl1amp2, &pl2amp2, &pl3amp2, &pl4amp2, &pl5amp2, &pl6amp2, &pl7amp2, &pl8amp2,
                                      &pl9amp2, &pl10amp2, &pl11amp2, &pl12amp2, &pl13amp2, &pl14amp2, &pl15amp2, &pl16amp2 };
AudioAmplifier *gainPerBar[MAX_PLUGINS]{ &pl1amp, &pl2amp, &pl3amp, &pl4amp, &pl5amp, &pl6amp, &pl7amp, &pl8amp,
                                         &pl9amp, &pl10amp, &pl11amp, &pl12amp, &pl13amp, &pl14amp, &pl15amp, &pl16amp };

AudioAmplifier *dryVolume[MAX_PLUGINS]{ &pl1dry, &pl2dry, &pl3dry, &pl4dry, &pl5dry, &pl6dry, &pl7dry, &pl8dry,
                                        &pl9dry, &pl10dry, &pl11dry, &pl12dry, &pl13dry, &pl14dry, &pl15dry, &pl16dry };
AudioAmplifier *FX1Volume[MAX_PLUGINS]{ &pl1FX1, &pl2FX1, &pl3FX1, &pl4FX1, &pl5FX1, &pl6FX1, &pl7FX1, &pl8FX1,
                                        &pl9FX1, &pl10FX1, &pl11FX1, &pl12FX1, &pl13FX1, &pl14FX1, &pl15FX1, &pl16FX1 };
AudioAmplifier *FX2Volume[MAX_PLUGINS]{ &pl1FX2, &pl2FX2, &pl3FX2, &pl4FX2, &pl5FX2, &pl6FX2, &pl7FX2, &pl8FX2,
                                        &pl9FX2, &pl10FX2, &pl11FX2, &pl12FX2, &pl13FX2, &pl14FX2, &pl15FX2, &pl16FX2 };
AudioAmplifier *FX3Volume[MAX_PLUGINS]{ &pl1FX3, &pl2FX3, &pl3FX3, &pl4FX3, &pl5FX3, &pl6FX3, &pl7FX3, &pl8FX3,
                                        &pl9FX3, &pl10FX3, &pl11FX3, &pl12FX3, &pl13FX3, &pl14FX3, &pl15FX3, &pl16FX3 };
```

☐ .) Place your control variables in variables.ino.

Use a “struct” for saving presets. For each control you need 2 variables:

byte/float/int “module_function”; // your desired range
byte “module_function”_graph; // range: 0-127

Also set your Plugin name into the PluginName[] Array!!

```
//plugin 3 variables
struct plugin3 {
  int Filter1_Frequency = 260;
  byte Filter1_Frequency_graph = 50;
  float Filter1_Resonance = 1;
  byte Filter1_Resonance_graph = 50;
  float Filter1_Sweep = 2;
  byte Filter1_Sweep_graph = 50;
  byte Filter1_Type = 0;
  byte Filter1_Type_graph = 0;
  int Env1_Attack = 50;
  byte Env1_Attack_graph = 50;
  int Env1_Decay = 50;
  byte Env1_Decay_graph = 50;
  float Env1_Sustain = 1;
  byte Env1_Sustain_graph = 50;
  int Env1_Release = 150;
  byte Env1_Release_graph = 50;
  float note1_Velo = 1;
  byte note1_Velo_rnd;
  byte wfSelect;
  byte wfSelect_graph;
};
plugin3 pl3[MAX_PRESETS];
byte pl3presetNr = 0;
```

☐ .) Add these 7 functions to your new plugin *.ino file

void Plugin_3_Settings()
void Plugin3_Control()
void Plugin3_Page1_Dynamic()
void Plugin3_Page_Static(byte Pagenumber)
void Plugin3_Change()

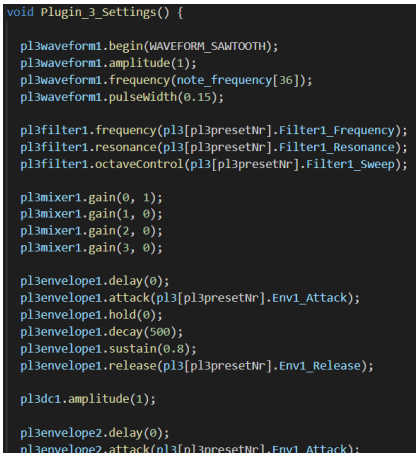
void savePlugin3()
void loadPlugin3()

```
> void Plugin_3_Settings() {···
  }
> void Plugin3_Control() {···
  }
> void Plugin3_Page1_Dynamic() {···
  }
> void Plugin3_Page_Static(byte Pagenumber) {···
  }
> void Plugin3_Change() {···
  }


> void savePlugin3() {···
  }
> void loadPlugin3() {···
  }
```

☐ .) “void Plugin_3_Settings()”

This function is used to set all functions from the used Modules to a desired start value.
Give all functions a desired value. Some functions may not be used, but then they are set to the correct value.



```
void Plugin_3_Settings() {

  pl3waveform1.begin(WAVEFORM_SAWTOOTH);
  pl3waveform1.amplitude(1);
  pl3waveform1.frequency(note_frequency[36]);
  pl3waveform1.pulseWidth(0.15);

  pl3filter1.frequency(pl3[pl3presetNr].Filter1_Frequency);
  pl3filter1.resonance(pl3[pl3presetNr].Filter1_Resonance);
  pl3filter1.octaveControl(pl3[pl3presetNr].Filter1_Sweep);

  pl3mixer1.gain(0, 1);
  pl3mixer1.gain(1, 0);
  pl3mixer1.gain(2, 0);
  pl3mixer1.gain(3, 0);

  pl3envelope1.delay(0);
  pl3envelope1.attack(pl3[pl3presetNr].Env1_Attack);
  pl3envelope1.hold(0);
  pl3envelope1.decay(500);
  pl3envelope1.sustain(0.8);
  pl3envelope1.release(pl3[pl3presetNr].Env1_Release);

  pl3dc1.amplitude(1);

  pl3envelope2.delay(0);
  pl3envelope2.attack(pl3[pl3presetNr].Env1_Attack);
```

Copy this function in “functions.ino to:
void setup {}

☐ .) “void Plugin3_Control()”

This function is used to map the _graph value to the desired “function”-range.
For every function you want to control add your assignment here. Each switch-case stands for one of the four rows. This is the most code you have to write. In the end your “PluginX_Control” function will be about 100 lines long, if you use this method:

```
if (pl3[pl3presetNr].***_graph != Potentiometer[x]) {
  pl3[pl3presetNr].***_graph = Potentiometer[x];
  pl3[pl3presetNr].*** = map(pl3[pl3presetNr].***_graph, 0, 127, min, max);
  module.function(pl3[pl3presetNr].***);
  drawPot(CTRL_COL_x, CTRL_ROW_x, pl3[pl3presetNr].***_graph, pl3[pl3presetNr].***,
"***", trackColor[desired_track]);
  }
```

where *** is the desired function.

Copy this function in “functions.ino to:
void Plugin_View_Dynamic() {}

☐ .) “void Plugin3_Page1_Dynamic()”

This function is used to assign the Encoder Movement to the desired _graph value. Here we add the Encoder value to the last stored “plX[desired_track].***_graph” value and give it to “Potentiometer[x]”. Do this for every control you have added above.

```
Potentiometer[x] = pl3[pl3presetNr].***_graph;
if (enc_moved[x]) {
  Potentiometer[x] = constrain((pl3[pl3presetNr].***_graph + encoded[0]), 0, 127);
  }
```

where *** is the desired function.

Copy this function in “functions.ino to:
void Plugin_View_Dynamic() {}

☐ .) “void Plugin3_Page_Static(byte Pagenumber)”

This function is used to show your controls once after you switch presets.
Copy the drawPot() functions from “PluginX_Control” into this function. Whenever you call your Plugins Page, these are the things that will show up. Add “clearworkspace()”, your “PluginX_Change()” and the rectangle for the Preset Number functions. This function is only called once.Don't put any interactive stuff inside.

```
void Plugin3_Page_Static(byte Pagenumber) {
  clearWorkSpace();
  Plugin3_Change();
  drawNrInRect(18, 1, pl3presetNr, ILI9341_PURPLE);

  //case 0
  drawPot(CTRL_COL_0, CTRL_ROW_0, pl3[pl3presetNr].wfSelect_graph, pl3[pl3presetNr].wfSelect, "WF
  //case 1
  drawPot(CTRL_COL_0, CTRL_ROW_1, pl3[pl3presetNr].Filter1_Frequency_graph, note_frequency[pl3[pl
  drawPot(CTRL_COL_1, CTRL_ROW_1, pl3[pl3presetNr].Filter1_Resonance_graph, pl3[pl3presetNr].Filt
```

Copy this function in “functions.ino to:
void Plugin_View_Static() {}

☐ .) “void Plugin3_Change()”

This function is used for preset changes.
Copy the `module.function(pl3[pl3presetNr].***);` module´s functions into this page.

Copy this function in “functions.ino to:
void beatComponents {}
and your
void Plugin3_Page_Static(byte Pagenumber)

☐ .) “void savePlugin3()”

This function is used to save your Presets.
Copy one of the existing save_Plugin() functions into your file and change the _graph variable to yours.

Copy this function in “songmode.ino to:
void savebutton {}
and your
void Plugin3_Page_Dynamic()

☐ .) “void loadPlugin3()”

This function is used to load your Presets.
Copy one of the existing load_Plugin() functions into your file and change the _graph variable to yours.

Copy this function in “songmode.ino to:
void loadbutton {}
and your
void Plugin3_Page_Dynamic()

☐ .) NoteOn/NoteOff´s

Add your NoteOn/NoteOff´s into the “void PluginPlay()” function.
From here your Plugin will be played via the sequencer, a MIDI Keyboard or the Launchpad.

Done!!!

Useful Lines:
selectFilterType(pluginchannel, mixerchannel) //if you want a selectable filtertype add your filtermixer inside

drawPot(xPos, yPos, fvalue, dvalue, dname, color); //fvalue = ***_graph; dvalue = ***
drawNrInRect(xPos, yPos, dvalue, color); // dvalue = ***
clearWorkSpace();