Daisy Tutorial 1 Script

Holland Sersen

September 15, 2021

Contents

1 Introduction

This tutorial was made on the Electrosmith Daisy Seed platform with beginners in mind. It will go over how to upload files to the Daisy and how to look through the example folders provided with the Daisy.

For this tutorial you will need; a Daisy Seed, a breadboard, and an audio jack to get audio from the daisy board. You will also need to follow along the daisy tutorial on how to install in the toolchains (Linked here).

This tutorial will not go into; intermediate C++ coding (for loops, main function, init objects) or hardware design (buttons, potentiometer, rotary encoders, or audio inputs) however they will be available here (Link).

2 Opening Folders in VSCode

First we have to open the folder in VSCode. There are a couple ways of doing this but the easiest way is to open up VSCode, Click File, then Open and then find the example folder you would want to open. For this tutorial we will be exploring the oscillator example folder.

VSCode should automatically show all the file folders and directories. You should see a couple things. First we have a folder called VSCode, this folder just contains settings for VSCode and is unimportant for this tutorial, so ignore it. Next we have the MAKEFILE. Although this isn't covered in the tutorial, the MAKEFILE is what tells the compiler what to compile and how to compile it. If you want to learn more then there is a link in the description for some resources on the basics of MAKEFILES.

Next you should see the file oscillator.cpp, click on this. This file is what our Daisy will run, it defines what our daisy program should do and how it should react to incoming audio.

3 Compile Files to Daisy

When you opened this folder VSCode automatically created a terminal in the current folder of your computer. To open this, click the button called "terminal" on the bottom bar. Next, type in the terminal "make". This will automatically compile the code, it will also tell you if you have any errors in your code or if it can't compile.

If you get an error check to make sure your code matches the code in the examples on Github, or verify that you have the toolchain installed correctly. Then to upload the code to your Daisy, while it is connected via USB type in the console “make -dfu” this should work without trouble.

4 How to get audio in and out of the Daisy Seed

Next we need to get audio from the Daisy and make it audible. I would recommend some sort of bread board so you can test circuits on the Daisy. The first and most important circuit is getting audio out from the circuit. Find the pinout of the Daisy from Github, some wires and your audio jack.

5 C++ File Functions

Now that we got the Daisy working on a basic level let’s take a quick look at what the code is doing. First is the main function:

```
int main(void)
{
    // initialize seed hardware and oscillator daisysp module
    float sample_rate;
    seed.Configure();
    seed.Init();
    sample_rate = seed.AudioSampleRate();
    osc.Init(sample_rate);

    // Set parameters for oscillator
    osc.SetWaveform(osc.WAVE_SIN);
    osc.SetFreq(440);
    osc.SetAmp(0.5);


    // start callback
    seed.StartAudio(AudioCallback);
    while(1) {}
}
```

The main function is what the daisy runs, all of the code is defined here in someway, and this is the function the daisy will run. If you want to learn more about main functions there will be a link in the description below. But for this tutorial all you need to know is that the main function sets the frequency of the oscillator

Next we have the process audio block. The Process audio block is where the audio is actually played back and processed. You can see the oscillator is being played here and is going into the both the left and right audio output.

```
static void AudioCallback(AudioHandle::InterleavingInputBuffer
in, AudioHandle::InterleavingOutputBuffer out, size_t size)
{
    float sig;
    for(size_t i = 0; i < size; i += 2)
    {
        sig = osc.Process();

        // left out
        out[i] = sig;

        // right out
        out[i + 1] = sig;
    }
}
```

6 Compile code and verify that it is working

Let's change the oscillator's pitch so that it sounds different that the default. First lets change the osc.SetFreq to 880 instead of 440.

```
//Changed from 440 to 880
osc.SetFreq(880);
```

This should produce a perfect octave above the previous note. So let's compile this to the Daisy and see this for ourselves. First lets

7 Outro