Datafeel Dev-kit
User Guide

Table of Contents

Overview

The Datafeel Dots are multi-modal haptic devices that enable you to create an immersive experience beyond the screen, unlocking physical dimension to a product.

DF Dev Kits are designed to provide plug-and-play experience to first-time users while enabling advanced designers to create new applications or enhancements to existing products.

This Document will walk you through the basic steps to set up your development environment to run DF sample programs.
In addition to the hardware, i.e. the Dots and El Jefe, all you will need to do is the following:

- Install Visual Studio
- Cloning our sample programs git repository
- Load the DF NuGet package onto your Visual Studio
- Connect the Dots to your Computer

After these few easy steps, you will be able to run any of our samples and see the Dots come alive!

Visual Studio for Windows

Visual Studio is an integrated development environment (IDE) from Microsoft. It supports multiple programming languages (C++, C#, Python, JavaScript, etc.) and provides tools for software development, debugging, and version control.

For the purpose of this document, we will be using Visual studio 2022 to run the sample programs made to demonstrate the Datafeel Dot capabilities, showcasing different aspects of the Dots.

*Please note that Visual Studio no longer supports MacOS devices. Please install Visual Studio Code instead, which is described in the next section.

Step 1: Download the Visual Studio Installer

1. Go to the Visual Studio download page.
2. Click **“Download Visual Studio”**.
3. Run the downloaded .exe file

Step 2: Install Visual Studio Using the Installer

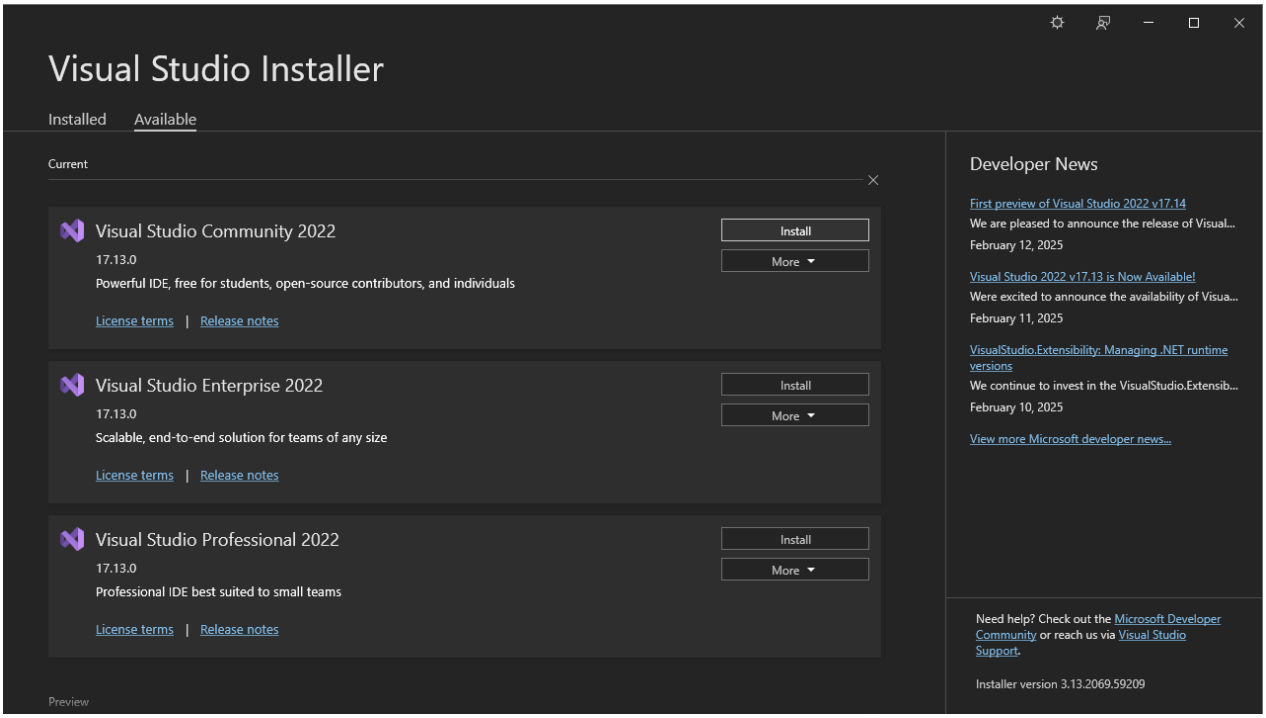

Fig 1. Visual Studio Installer

1. When the **Visual Studio Installer** opens, it will check for updates and download necessary files.
2. Install Visual Studio Community 2022 – free version
3. Once ready, you'll see a **workload selection screen**. Include the following:
    - **.NET desktop development** (C#/.NET applications)

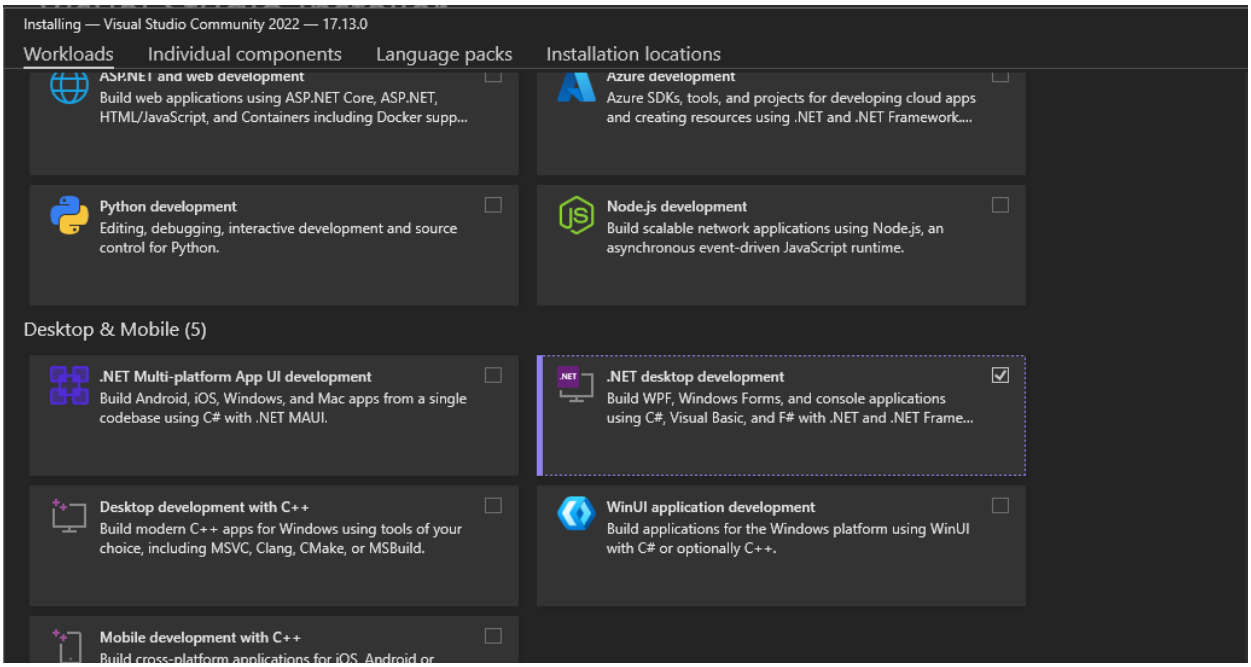

Fig 2. Visual Studio Installer Workloads

4. Click **Install** and wait for the process to complete.

Step 3: Launch and Configure Visual Studio

1. Open Visual Studio after installation.

You are now ready to use Visual Studio!

Additional Resources

- Visual Studio Documentation

Visual Studio Code

Visual Studio Code (VS Code) is a lightweight, open-source code editor developed by Microsoft. It supports multiple programming languages, extensions, and integrations, making it a versatile choice for developers working on C#, Python, and other languages.

This section will show how to install VSCode specifically for MacOS with the corresponding secondary installation (we recommend Visual Studio for Windows).

Step 1: Install Visual Studio Code on macOS

1. **Download VS Code** from the official website.
2. Open the downloaded .zip file, which extracts Visual Studio Code.app.
3. Drag and drop **Visual Studio Code.app** into the **Applications** folder.
4. Open **VS Code** from Launchpad or Finder. If prompted, click **"Open"** to bypass security warnings.

Step 2: Install Extensions for C# Development

1. Open VS Code and go to **Extensions** (Cmd + Shift + X).
2. Search for and install:
    - **C# Dev Kit** (Provides IntelliSense, debugging, and project management).
    - **.NET Install Tool for Extension Authors** (Ensures .NET SDK compatibility).
    - **NuGet Package Manager** (Manages dependencies for .NET projects).
    - **Python** (official Microsoft extension for Python support).

Step 3: Install .NET SDK on macOS

1. Download the **.NET 8.0 SDK** from Microsoft's official site.
2. Install the downloaded .pkg file.
3. Verify installation by running this in your terminal: *dotnet –version*

Repository Setup

In this section we will go through how to clone the sample repo from GitHub. In this section you will learn about the devkit-samples repository, however the same process will be applicable to any repository you'd want to clone.

Step 1: Install Git

1. Download **Git** from the official site: Git.
2. Run the installer and follow these steps:
    - Keep default settings unless you need customization.
    - Ensure **"Git Bash"** and **"Git GUI"** are selected.
    - Choose "Use Git from the Windows Command Prompt" when prompted.
    - Select "Windows-style line endings" (recommended).
    - Finish installation and restart your system if needed.

Step 3: Clone a GitHub Repository

1. Open Git Bash or Command Prompt (bash/zsh).
2. Navigate to the directory where you want to clone the repository:
    - cd path/to/your/folder
3. Navigate to the "devkit-samples" repository here.

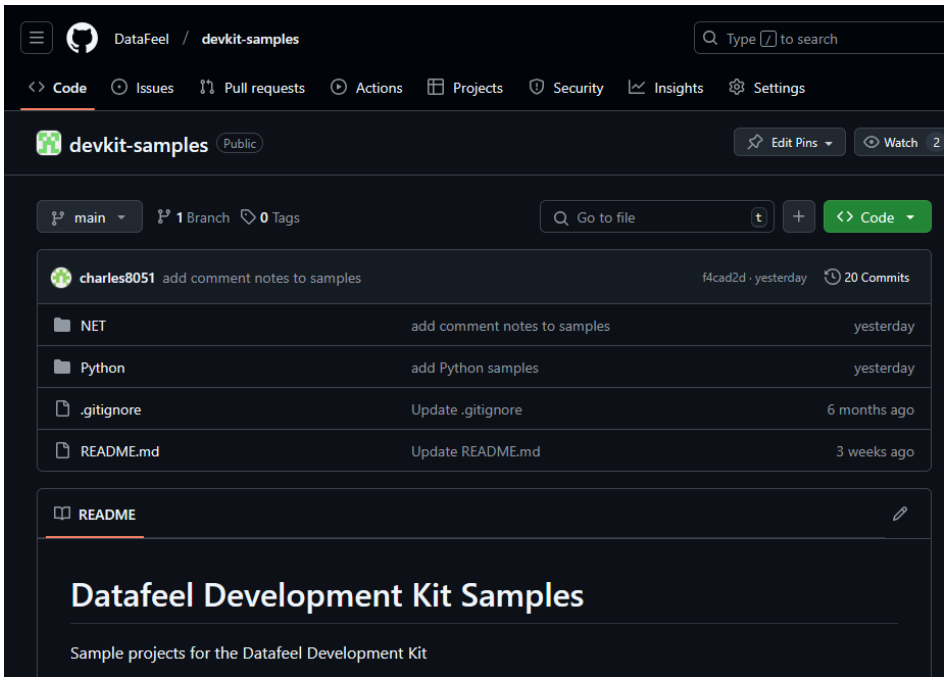

Fig 3. Copy the HTTPS GitHub URL

4. Run the clone command:
    - git clone https://github.com/DataFeel/devkit-samples.git
5. Navigate into the cloned repository:
    - cd “devkit-samples”
    - you can also open the folder you cloned the repo in. it should look something like this:

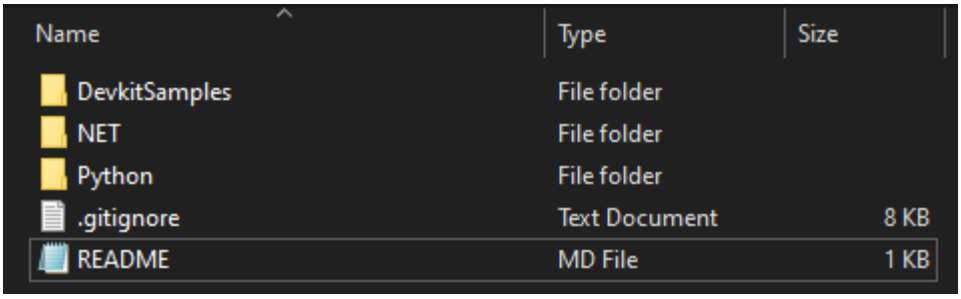

Fig 4. Root repository folder structure

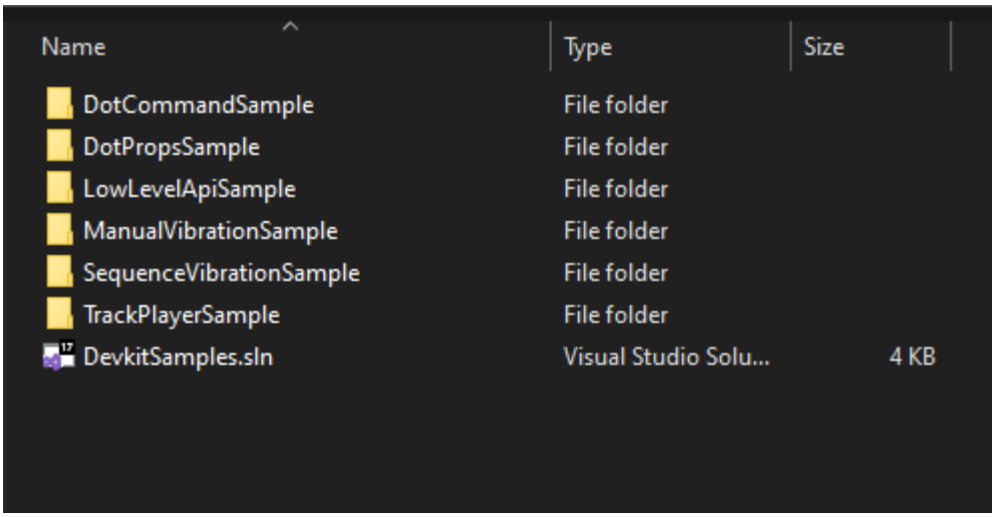

Fig 5. Inside the NET/DevKitSamples folder in the repository

Now you have access to samples repository and can run the programs locally!

.NET and NuGet

.NET Framework

The Datafeel DevKitSamples primarily uses the .NET8.0 framework. If you installed the .Net Desktop development package with your Visual Studio 2022, you have already added and installed the necessary .Net frameworks needed to run the Datafeel project.

If you need to add this framework to your environment, you can download it here.

On VS Code if you have followed our installation procedure, you have already installed the right extension for managing NuGet and the right .NET framework.

You can use the dotnet CLI tool to run any dotnet and NuGet commands in your terminal, without relying on your IDE. However, this document shows you other ways to do so as well.

Datafeel NuGet

The primary Datafeel SDK needed to operate the dots is offered as a NuGet Package that can be added to any project. By cloning the repo, you should already have the necessary Datafeel NuGet Packages installed.

Are the packages installed?

If you open one of the samples and there are compiler errors on the include Datafeel statement at the very top (red squiggly line), that is most probably a sign that there are issues with your packages, either not installed or at the wrong path.

How to check in **VS Code using the extension:**

1. Open the Command Palette (Cmd + Shift + P / Ctrl + Shift + P).
2. Type and select "NuGet: Manage Packages for Solution" to view installed packages.

how to check within **Visual Studio**:

- You can check this by Navigating to the VS NuGet package manager for the devkitSamples project, under Tools -> NuGet Package Manager -> Manage NuGet Packages for Solutions:

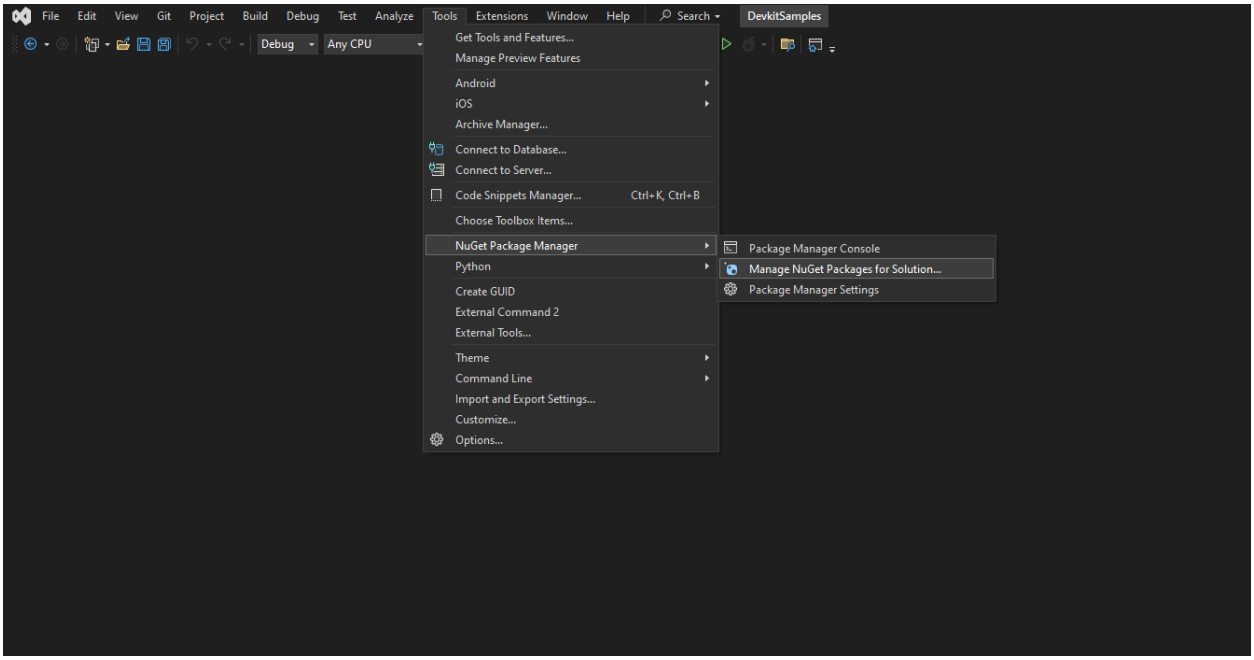

Fig 6. NuGet package manager option in Toolbar in Visual Studio

- If you see the following three packages, you are good to run the programs!
    1. Datafeel – core SDK
    2. Datafeel.NET.BLE – For Bluetooth connectivity
    3. Datafeel.NET.Serial – For Serial port connectivity

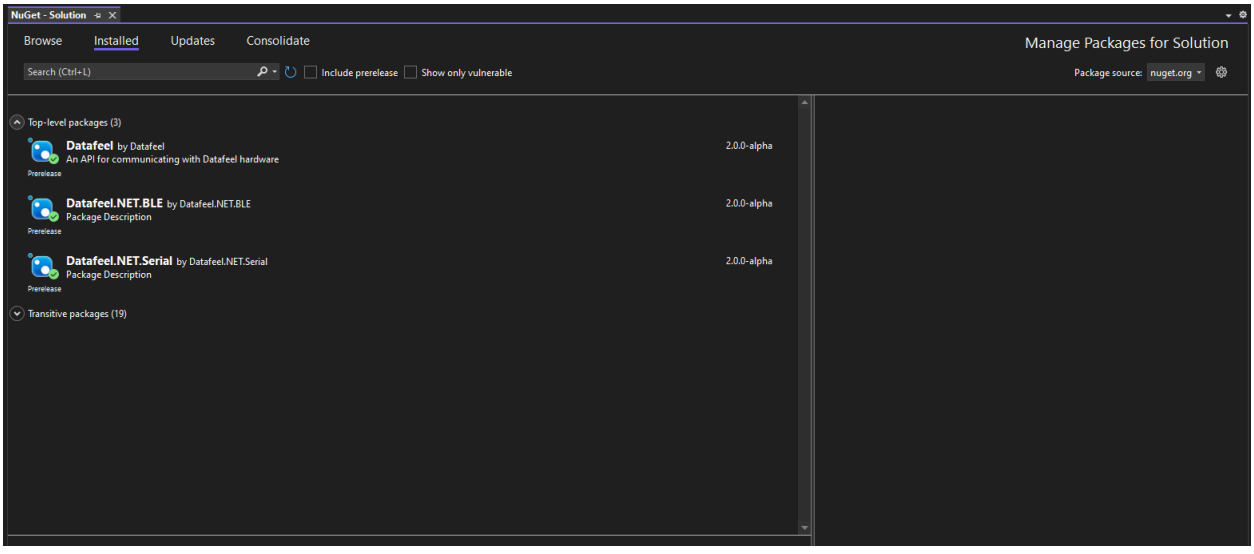

Fig 7. Three Datafeel packages visible and installed in NuGet package manager

Where to find packages

There are multiple ways to access and download the NuGet packages. You can use the Browse tab in the NuGet package manager shown in figure 7, and search for Datafeel

You can also download the NuGet packages here.

How to Connect the Dots

Fig 8. Datafeel's El Jefe, two Datafeel Dots, and 5 pin connector cables

There are two primary ways of connecting the dots to your computer to run the samples:

1. Serial connection using a USB-C cable
2. Bluetooth

However, you must connect the dots in a chain and connect them to the El jefe (control module).

The Datafeel devkit can come with up to four dots.

- Using the 5 pin cables, chain your dots and the El jefe module together

Fig 9. El Jefe and two dots chained

El Jefe ports

El Jefe is the Command module that is the mid-point device between your computer and the dots.

- It has a female 5-pin connector port, a power button, USB-C charging port, and USB-C Serial port, as viewed in Figure 10.

Fig 10. El Jefe ports

El Jefe power supply:

El Jefe comes with batteries and can be recharged, enabling wireless connectivity if desired. It can also run on direct power supply if the batteries are removed or are empty.

- You can power El Jefe with any USB-C charger if it supplies above 20W of power.
- If the batteries are charging, the El Jefe battery LED will turn green, and will turn off once batteries are full.
- If the batteries are in use, the Battery Led will turn Red, indicating that the El Jefe is ON.
- If the USB connection is getting power from your computer, it will turn blue, indicating a connection.
- To turn on the El jefe, press the power button.

Connect

- To connect the dots via Serial port, use a USB-C cable to connect the El Jefe to any of your computer's serial ports. (any USB port should work)
- Turn the dots ON by pressing the power button. The dots will have their blue LEDs turned on and run a "breathing" sequence as seen in figure 9.

You are now ready to run the sample programs!

C# Samples

The "devkit-samples" repository comes with six different C# code samples in *NET/DevKitSamples* folder demonstrating different aspects of the dot's capabilities as well as the SDK's customizability and API.

1. DotCommand sample – Demonstrates how to send simple haptic commands to the dots
2. DotProps sample – Demonstrates all the different properties one has access to through the SDK
3. LowLevelApi Sample – Demonstrates how to send commands to the dots without using the high-level API within the SDK
4. ManualVibration Sample – Demonstrates how manually set Vibration Intensity and frequency of the dots
5. SequenceVibration Sample – Demonstrates how use the pre-made vibration sequences (for example 3 taps)
6. TrackPlayer Sample – Demonstrates how the dots can receive and play a pre-defined JSON Haptic track using the API

To run any of the samples:

1. Open the DevkitSamples.sln file in VS within the repository, visible in figure 5
2. Navigate to the desired sample through the Solutions explorer and open the corresponding Program.cs

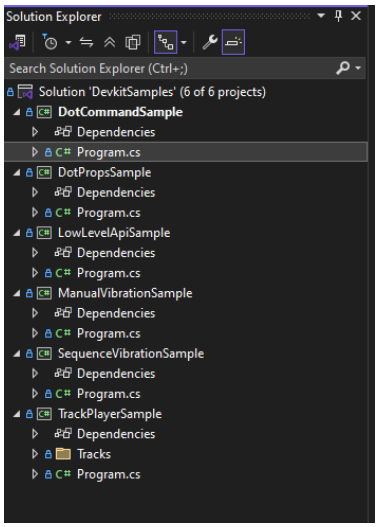

Fig 11. Solution explorer in Visual Studio

3. Choose the desired sample as the startup item in the Toolbar

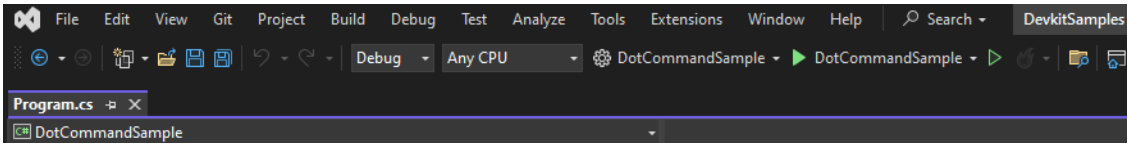

Fig 12. DotCommandSample chosen as the start-up item

4. Assuming your dots are on and connected to the computer, If you click run (green play button), the sample will run and the dots will come alive!
5. To Stop the program, close the terminal window.

C# Sample Example: DotCommand

Below is the Code for the DotCommand Sample with comments describing what each section does:

```
// the 3 packages needed to connect
using Datafeel;
using Datafeel.NET.Serial;
using Datafeel.NET.BLE;

// Create DotManager and specify the address of each dot you have connected
var manager = new DotManagerConfiguration()
    .AddDot(1)
    .AddDot(2)
    .AddDot(3)
    .AddDot(4)
    .CreateDotManager();

// DotpropsWritable lets write any of the writable properties within a dot, based on the address of the dot specified
// Per each dot address, here we set the LEDs to global and breathe mode, and enabling Manual Vibration mode, allowing us to specify the Intensity and frequency of the dot vibration
var dots = new List<DotPropsWritable>()
{
    new DotPropsWritable() { Address = 1, LedMode = LedModes.Breathe, GlobalLed = new(), VibrationMode = VibrationModes.Manual},
    new DotPropsWritable() { Address = 2, LedMode = LedModes.Breathe, GlobalLed = new(), VibrationMode = VibrationModes.Manual},
    new DotPropsWritable() { Address = 3, LedMode = LedModes.Breathe, GlobalLed = new(), VibrationMode = VibrationModes.Manual},
    new DotPropsWritable() { Address = 4, LedMode = LedModes.Breathe, GlobalLed = new(), VibrationMode = VibrationModes.Manual},
};

foreach(var d in dots)
{
    d.VibrationIntensity = 1.0f; // Set to Maximum dot intensity
    d.VibrationFrequency = 150;  // Set to 150 Hz
}

using (var cts = new CancellationTokenSource(10000))
{
    try
    {
        // Create Serial modbus client
        var serialClient = new DatafeelModbusClientConfiguration()
            .UseWindowsSerialPortTransceiver()
```

```
            .CreateClient();
        // Create Bluetooth modbus client
        var bleClient = new DatafeelModbusClientConfiguration()
            .UseNetBleTransceiver()
            .CreateClient();
        var clients = new List<DatafeelModbusClient> { serialClient, bleClient };
        // Start the communications with the dots
        var result = await manager.Start(clients, cts.Token);
        if (result)
        {
            Console.WriteLine("Started");
        }
        else
        {
            Console.WriteLine("Failed to start");
        }
    }
    catch (Exception e)
    {
        Console.WriteLine(e.Message);
    }
}
var random = new Random();

while (true)
{
    // while the program is running, increase the frequency by 10 every 100ms, till it reaches 250 Hz, then reset
to 100Hz and repeat
    var delay = Task.Delay(100);
    foreach (var d in dots)
    {
        d.VibrationIntensity = 1.0f;
        d.VibrationFrequency += 10;
        if(d.VibrationFrequency > 250)
        {
            d.VibrationFrequency = 100;
        }

        try
        {
            using (var writeCancelSource = new CancellationTokenSource(250))
            using (var readCancelSource = new CancellationTokenSource(250))
            {
                await manager.Write(d, writeCancelSource.Token);
                var result = await manager.Read(d, readCancelSource.Token);
            }
        }
        catch (Exception e)
        {
            Console.WriteLine(e.Message);
        }
```

```
    }
    await delay;
}
```

Python API

This section provides step-by-step instructions for installing and using the Python version of the SDK API. It includes installation steps, dependency management, and basic usage guidelines.

Installing Python and Pip

Windows

1. Download the latest version of Python from the official website.
2. Run the installer and check the box "Add Python to PATH" before installing.
3. Complete the installation by following the on-screen instructions.
4. To verify installation, open Command Prompt and run:
    a. *python –version* OR *py –version*
    b. *pip –version*
        i. if this doesn't work, try running: *py -m pip –version*

MacOS

You are in luck! MacOS comes with Python pre-installed.

That said you should install Homebrew, and check to see if your python version is the latest:

1. Follow the installation guide on the official Homebrew Documentation.
    - Alternatively, you can run the following in your terminal: */bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"*
2. To re-install Python run the following in your terminal: *brew install python*
3. Verify your installation:
    - *python3 --version*
    - *pip3 –version*

Installing Python DF Packages

1. Using your terminal command line or PowerShell or Git Bash, navigate to the Python folder in the repo:
    - *cd <root path>/devkit-samples/Python*
    - there should be a file named "*setup.py*" inside this folder
        - you can check this by running the "*ls*" command in your terminal
2. Now you can install all of the sdk dependencies using pip. Run the following in your terminal window:
    - Windows: *pip install ./* OR *py -m pip install ./*
    - MacOS: *pip3 install .*

You are now ready to run the python samples! Open the samples in any IDE you'd like, including VS or VSCode.

API Definition

The python version is essentially the translation of the C# SDK explained in the C# section. Therefore, you have access to the same functionalities in python, as you would in the C# version.

- Open the *device.py* file to see all the functions you have access to
- To use the API, Make sure to import *datafeel.device* in your python script

Python Sample

The underlying logic on how you can interact with the DF dots are the same as the C# version.

- To see how the API is used, Navigate to the *read_write.py* script within the samples folder.
- You are able to run the script by clicking the green play button in your IDE, exactly the same as the way we executed the C# samples
- Make sure the Dots are plugged in before you run the sample
- To Reset the dots, unplug and re-plug them.

Unity SDK

Installation & Setup

1. If you haven't already, Install the driver Silicon labs CP210x Windows Driver here: https://www.silabs.com/documents/public/software/CP210x_Windows_Drivers.zip
2. Set your Unity project API Compatibility Level to .NET Framework.
3. Using the Unity editor Package Manager, select Install package from disk..., navigate to the unzipped Datafeel Unity SDK, and select the package.json in the com.datafeel.unity.sdk directory. The package manager will now install the Datafeel SDK and all of its dependencies.

Samples

Import some examples by clicking on Samples from the Unity Package Manager while the Datafeel SDK is selected and click Import on the Simple Read & Write Sample. Inside you will find unity example scenes, scripts, and prefabs.

Usage

To use the Datafeel Unity SDK in your unity project, follow the installation instructions above.
Once your project has successfully imported the Datafeel Unity SDK package, you will be able to add the DatafeelService component to a game object in your scene. When this game object is enabled in the scene, the DatafeelService component can be accessed directly or via the static DatafeelService.Instance.

Datafeel Contacts

Matthew Leaper – CEO – ceo@datafeel.com

Leigh Sembaluk – CTO – Leigh@datafeel.com

Charles Lee – Technical Director – charles@datafeel.com