4.1 Notch Android App

4.1.1 Notch Outputs

The data inputs to the Sonification System come from the Notch sensors via the Notch API, which theoretically allows for real-time data streaming from up to 6 sensors simultaneously. These sensors output both raw positional data for different points on the body, as well as angular data in multiple degrees of motion for different joints[1]. The available raw outputs from the Notch sensors as well as the possible positions that they can be placed on the body are showing in Figure 3 below.

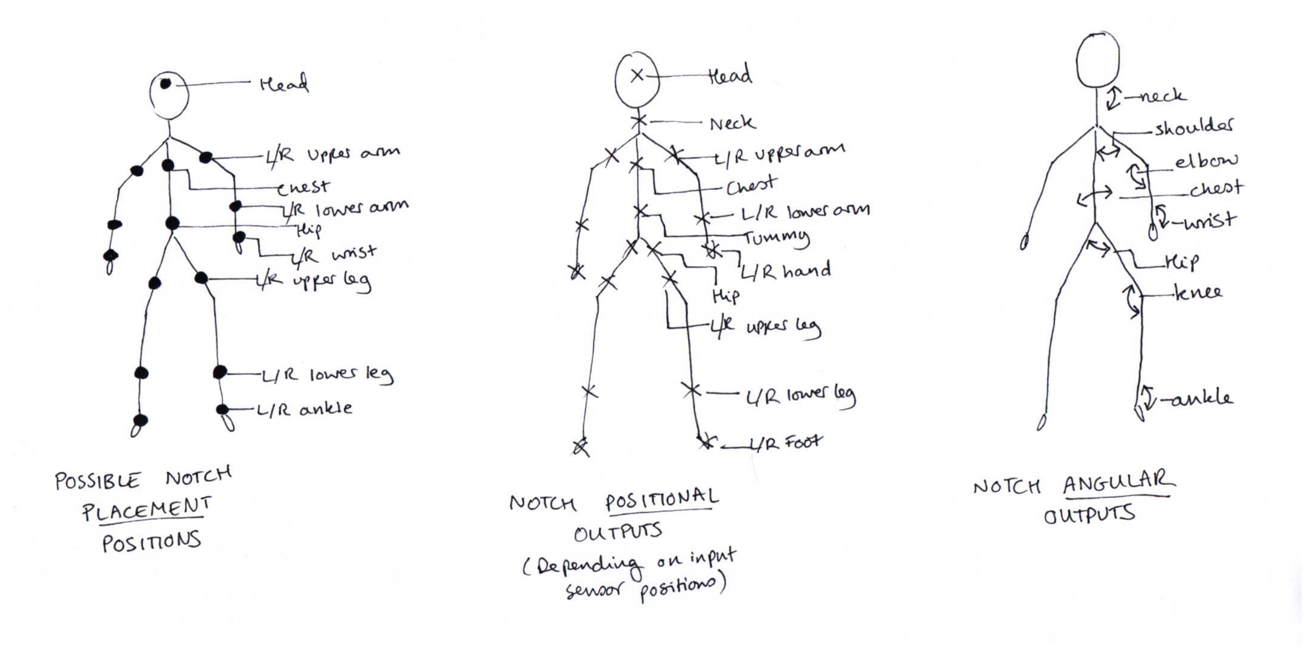


Figure 1 - Notch Outputs

4.1.3 Notch Android Test App

As discussed in Chapter 3, data from Notch sensors is accessible through a proprietary API implemented in Java. The source code for an example 'Tutorial App' is provided by Notch[2], which provided a basis on top of which to build the streaming functionality necessary to pass movement data in real time from Android to the Python analysis application running on Mac. When making a real-time capture, the inProgress() callback within the capture() method of the MainFragment (the main UI class) allows access to a data object, mRealTimeData, which holds the motion data captured from the whole skeleton during the current frame. The implementation of streaming functionality requires extracting the appropriate data points

[1] More information on the Notch outputs can be found at https://docs.wearnotch.com/docs/motion_formats/

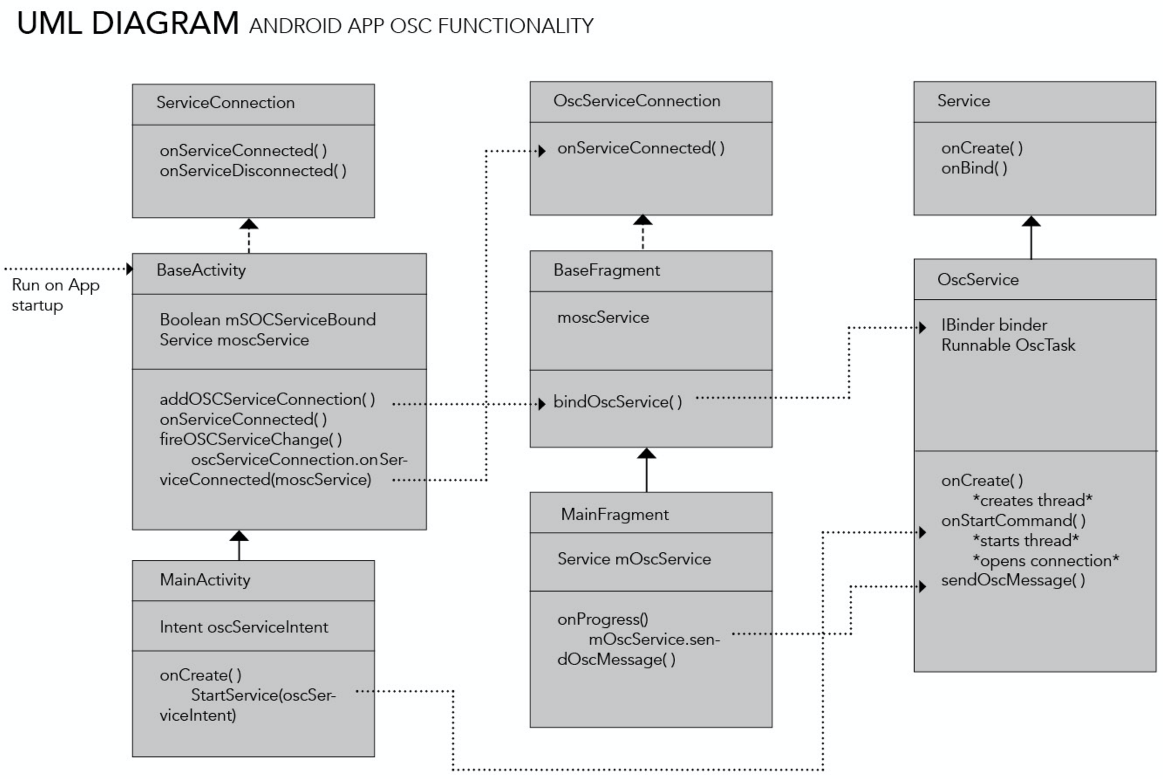
[2] The tutorial app provides functionality and a User Interface for connecting and calibrating the Bluetooth sensors, as well as making recordings that can be uploaded to the Notch server and viewed later.

from mRealTimeData object and passing these to some form of networking function that packages and sends this data to a receiving Mac-based client.

4.1.4 Data Streaming – Networking Threads in Android

The streaming functionality from Android to Mac proved tricky to implement for a number of reasons. Firstly, Android disallows networking operations from the main User Interface (UI) thread, because these could block UI elements whilst waiting for a response. Since the capture() method, from which sensor data is accessible, sits on the main thread, it is necessary to open a network connection in a different thread which remains running in the background and is accessible to data passed from the main thread in a thread-safe way. After several failed approaches, the successful approach uses a Service – an application component designed to run processes in the background of an app, without associated UI components. Services can also be bound to activities and fragments using a Binder and an Intent. The new Service, OSCService, itself creates a runnable (OSCTask), which is run when the Service is started, and opens a network connection which is then accessible through the public SendOSCMessage instance method of the OSCSercive class. The class structure of the OSC networking functionality which was built ontop of the Notch Android app is shown in the UML diagram in Figure 5 below.

Figure 2 - Android App UML Diagram



4.1.5 Android-Mac Networking Protocols

To implement the streaming functionality, a fast and lightweight networking protocol was necessary - since the Notches transmit data at 40hz, it needed to send 40 messages a second - and that therefore didn’t involve individual handshakes every time a message needed to be sent. Attempts to
use a Socket based protocol, where an initial TCP handshake was used to establish a reliable connection over which subsequent packets could be sent without further handshaking, were motivated by a desire for reliable transport, ensuring that messages arrived in order and without omission. However, several Socket based libraries (such as Socket IO (Rai, 2013) and the in-built Android Java Socket class), succeeded in establishing a connection but produced only garbled output at the receiving client. Having established that this was most likely due to mismatches between Android Java and Python protocol versions, it became clear that appropriate patches to match protocol versions were unavailable for both libraries.

This led to a change of tactic. In fact, for real time movement data being transmitted at 40 frames per second, the need for speed of transmission trumps the need for reliability. If some packets are dropped or arrived out of order, this is unlikely to have a big effect on the final sonification output, as long as most packets are delivered correctly. The successful solution therefore makes use of UDP – a connectionless protocol which does not rely on handshaking, and is therefore lightweight but by nature unreliable – and more specifically Open Sound Control (OSC), a simple protocol run over UDP which is often used in music and sound applications that require real-time communication (Freed, 1997). The open source JavaOSC library (JavaOSC, 2021) is used to implement OSC in the OSCService class, which enables the packaging of data as OSC Messages and sending these over UDP to a designated IP and port.

4.1.6 Notch Real-Time Capture: Challenges

Despite the successful implementation of functionality to pass messages out of the onProgress() callback and use UDP to transmit these to a Mac-based receiver, an additional hurdle was found when trying to extract data from the mRealTimeData object, which holds the raw notch data to be transmitted. The mRealTimeData has several instance methods, such as getPos() and getQ(), which allow access to the raw notch data stored within it. Both methods take a Bone argument (the bone for which data is required) and a Int FrameIndex argument – but it is not mentioned anywhere in the Notch documentation what this "FrameIndex" argument refers to, how frames are indexed, or the appropriate values to pass in order to extract data. Using a repeated value (e.g. 0, 1), initialising a counter which iterated through incrementing frames, or even feeding random values, all failed to produce any meaningful output from the getPos() and getQ() methods, resulting in repeated NaN (Not a Number) returns. Furthermore, repeated attempts to contact Notch through their helpdesk went unanswered. Due to these persistent problems, attempts to pull data from the mRealTimeData object, and thus to complete the full real-time streaming pipeline, were ultimately unsuccessful. However, the development of a parallel testing module (see section 4.4), allowed progress to continue unimpeded on the other parts of the development of the project, using pre-recorded test data.

Musical Sonification of Body Movement for Chronic Pain
System/User Manual

This is the combined system and user manual for the musical sonification software developed as part of a UCL BSc Project in 2020-2021.

This manual is designed to be used by researchers who wish to use the sonification software in their own research projects. A good level of competency with computers is helpful, since the software is provided as source code, and some basic configuration is required to run the software. However no explicit knowledge of programming is required to set up and use the software as it is provided.

The manual also contains enough detail to be used as documentation for researchers or software developers who wish to build on the existing source code for their own projects. Some of this information is more in depth than is needed to simply run the sonification. This information is marked in blue, and can be safely skipped by non-programmers who just wish to run the software.

Finally, the manual lists software and library versions that were used to develop and test the project. The software may still work on different versions, but it hasn't been tested using them so there may be unexpected behaviour or errors.

The following are required to run the software:

- A Mac laptop (Running OS Catalina 10.15.5). The following software should also be installed:
  - Android Studio (v4.1) with the following libraries:
    - JavaOSC 0.3 (https://github.com/hoijui/JavaOSC)
  - Python (v3.7) with the following libraries:
    - pyOSC3 1.2 (https://pypi.org/project/pyOSC3/)
    - matplotlib 3.4.1
    - oscpy 0.5.0 (https://pypi.org/project/oscpy/)
    - pandas 1.1.5
  - SuperCollider (v3.11.2)
- An Android mobile device (Running Android 8.1.0)
- Cable to connect Android device to laptop
- A deck of 6 notch sensors, their charging case, and elastic straps

Included files:

The package to run the software is provided as a zip file. Download and extract the zip file, and check that it contains the following folders and scripts:

- Android_Notch_App
  - In /app/src/main/java/com/wearenotch/notchdemo …
  - BaseActivity.java
  - BaseFragment.java
  - MainActivity.java
  - MainFragment.java

    - NotchApplication.java
    - NotchServiceConnection.java
    - oscService.java
    - oscServiceConnection.java
- Python_Analysis_Tool
    - Analysis_controller.py
    - Analysis_view.py
    - Analysis_model.py
    - Analysis_realtime_mode.py
    - Analysis_test_mode.py
- SuperCollider_Sonification
    - Sonification_version1_simple.scd
    - Sonficiation_Version2_evolving.scd
    - Sonficiation_version3_responsive.scd
    - Sonification_version4_full.scd
    - Sounds folder with one sound, “piano.wav”

Using SuperCollider:

- For more details about how to use supercollider, see http://doc.sccode.org/Browse.html#Tutorials%3EGetting-Started
- SuperCollider programs can be run section by section. To run a line or block of code, double click until it is highlighted in grey. Then press “Shift” + “Enter” on the keyboard. The code section should be run. Output (or Errors) are shown in the post window in the bottom right of the screen.
- At any time to stop sound whilst it is playing, press “Command” + “.” on the keyboard

Building and Installing the Android App.

NB: The current version of the software has not fully implemented the live streaming functionality from the Notch sensors, due to a problem with the Notch API. This section of the manual is useful for those wishing to build on the existing work, but can be skipped if you just want to run the software as it is.

1. Open the package in Android Studio
2. Add a local.properties file with a field ipaddress with the ipaddress of the host mac laptop running the Python analysis module and SuperCollider
3. Plug in the phone so it appears in devices menu in the top right hand corner
4. Build and run the gradle project
5. The app should launch on the phone
6. In the app, go through the following steps to connect and initialise the notch sensors. For more detail about how to connect and calibrate the sensors, please see https://docs.wearnotch.com/docs/pioneerapp_prepare_capture/
    a. Pair each sensor one by one
    b. Connect to network
    c. Calibration: Unchecked init > configure calibration > start calibration
    d. Steady: 1 notch init > configure steady > start steady
    e. Capture: Check “real time” box
    f. Capture: One notch init > Configure Real Time

7. At this stage you are ready to start realtime capture by pressing “Start real-time”. This will trigger functionality in the onProgress() method of the MainFragment() class in Android.

To Run the sonification script:

NB *To stop the audio at any time, press Command + Full Stop*

1. Select the version of the sonification you want to use, and open in supercollider. See the below “Sonification Versions” table for descriptions of the different versions.
2. The script is separated into parts that are marked by comments. This should help to show you how to run the script. Select the first lines (“s.boot;” and “thisProcess.openPorts”, and run them by pressing shift + enter.
3. Check that the audio output shown in the console is the output you want to hear sound on and that the volume is turned up. If the audio output is incorrect, quit SuperCollider, change the system audio output to the one you want, and restart from Step 1.
4. Select and run sections 2 and 3 (sound file code and synth code) to initialise the sonification settings
    a. These sections contain the definitions for instruments, which can be changed in order to change the sounds of the sonification.
    b. If you wish to use a different sound file for the granular synthesiser, replace the ~path variable with your own file path.
5. Finally, select and run the Background Pattern and the OSCDef section. Sound should begin to play.
6. Leave this running whilst proceeding to launch the Python script.

Sonification Versions

Sonification_version1_simple	Chimes (target threshold reaches), white noise filter (velocity)
Sonification_version2_evolving	Chimes (target threshold reaches), evolving bassline
Sonification_version3_responsive	White noise filter (velocity), Granular synth (energy)
Sonification_version4_full	Chimes (target threshold reaches), evolving bassline, white noise filter (velocity), granular synth (energy)

To run the Python script:

1. The analysis module of the sonification is a python script that needs to be run through the command line or a Python IDE (such as PyCharm https://www.jetbrains.com/pycharm/).
    a. If you are not familiar with command line, there are many good tutorials online, such as https://macpaw.com/how-to/use-terminal-on-mac.
2. Open a Terminal window. Navigate to the directory where you have extracted the python scripts.
    a. You can use the cd command to change directory. For example “cd /Users/bob/software/sonification/analysis”
3. In terminal, run the script using “Python3 script_name.py arg1 arg2 arg3”, where:

   a. Script_name should be replaced with the script you want to run. For "test" mode, where the movement data is read from a file, run "analysis_test_mode.py". Replace arg1 with the name of the folder where you have put the data to be read. Replace arg2 with the dimension you would like to sonify (either "bend", "lean", "twist" or "reach"). Replace arg3 with the target threshold value for that movement (see "Finding the target threshold value" below)
   b. To generate the CSV files to run data on, you can use the Notch Pioneer app to make data recordings, which will then appear in the "Library" section of the Notch cloud. When viewing the recordings in the cloud there is an option to download CSV files. Download and unzip this folder and place it in the root directory of the python analysis module (the same folder where the python scripts are).
   c. For realtime mode, where data is streamed from Android [This functionality is not fully working yet], run "analysis_realtime_mode.py". Replace arg1 with the IP address of the host machine (which you can find in system Settings > Network on Mac). Replace arg2 with the dimension you would like to sonify (either "bend", "lean", "twist" or "reach"). Replace arg3 with the target threshold value for that movement (see "Finding the target threshold value" below)
4. If running in test mode, the script should start reading data immediately and you should see data being printed to the terminal window. The sonification sound should start to change in supercollider, in response to the data changing, and you should see incoming OSC messages appearing in the Post Window in SuperCollider.
   a. Once the data has come to an end, a graph window should appear, showing the output over time of the movement analysis.
   b. NB. Sometimes, errors occur when reading data from a directory which is not in the root directory of the project (e.g. the folder which contains the python scripts.) If you are unable to read data from file, try moving the directory containing the data into the Python_Analysis_Module directory.
5. If running in realtime mode, the terminal should hang (nothing will appear), waiting for input from Android via OSC.
6. IF DOING REALTIME CAPTURE: At this stage it is time to launch the realtime capture on the android app. Press the "real time capture" button. A visualisation skeleton will appear.

Finding the target threshold value:

The Notch "Pioneer" app, which is available from the Google Play store and is used as the interface to the notches, provides tools which help to determine appropriate threshold values. In the app, launch a real-time capture whilst wearing the notches. A visualisation window with a skeleton animation will appear, reflecting your movement. In the top left corner is a drop down menu which allows you to choose a body part to show angle values for. Choose either chest/right knee/right shoulder depending on your preference. Then move to the desired threshold position, and note down the angular value displayed in the visualiser. This is the value which you can enter into the sonification app as a threshold value.

Additional Tools

In order to understand and analyse the outputs of the software, it’s useful to use some additional tools.

Animation

Animations are created in the Notch cloud, but you can also use Blender to animate the .bvh files. Blender is a free animation software which you can download from https://www.blender.org

1. Open an empty file in blender
2. Choose file > import and select the .bvh file
3. Ensure the camera is appropriately positioned (preview by pressing the camera button)
4. Set the output location for renders in the output properties tab (on the right) and set the output to be AVI JPEG
5. In that same tab, set the frames per second to 40
6. In the “Layout” tab view, select “view” (top right) and then select “viewport animation render”
7. Wait for the render to finish
8. The output animation video should appear in the folder you chose

Capture sound output

You can use BlackHole 2ch to re-route the audio from supercollider into a program where you can capture it.
https://existential.audio/blackhole/

Sound visualisations

You can use the free software Sonic Visualiser to create spectrograms of the audio output.
https://www.sonicvisualiser.org