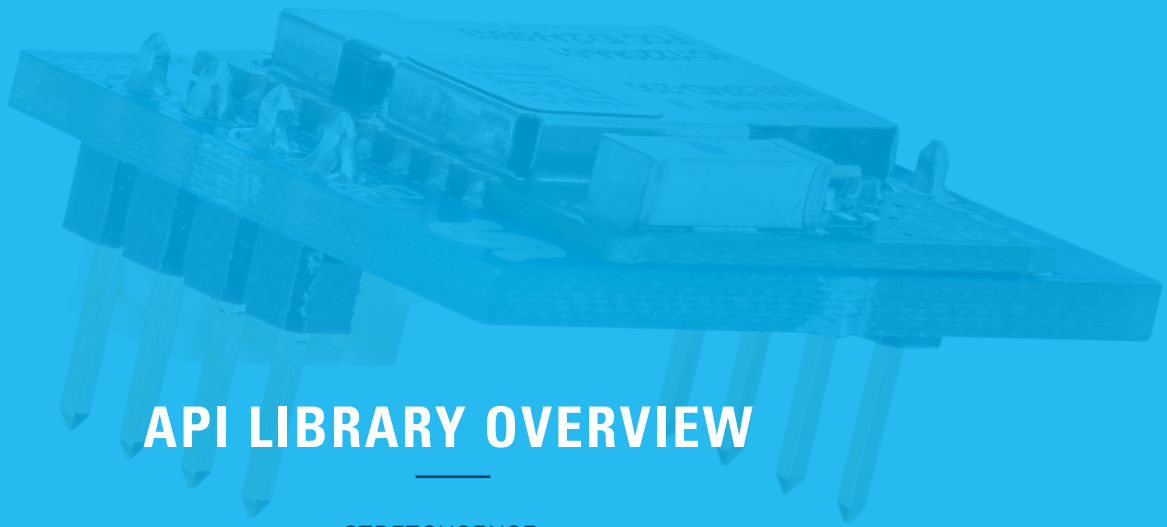
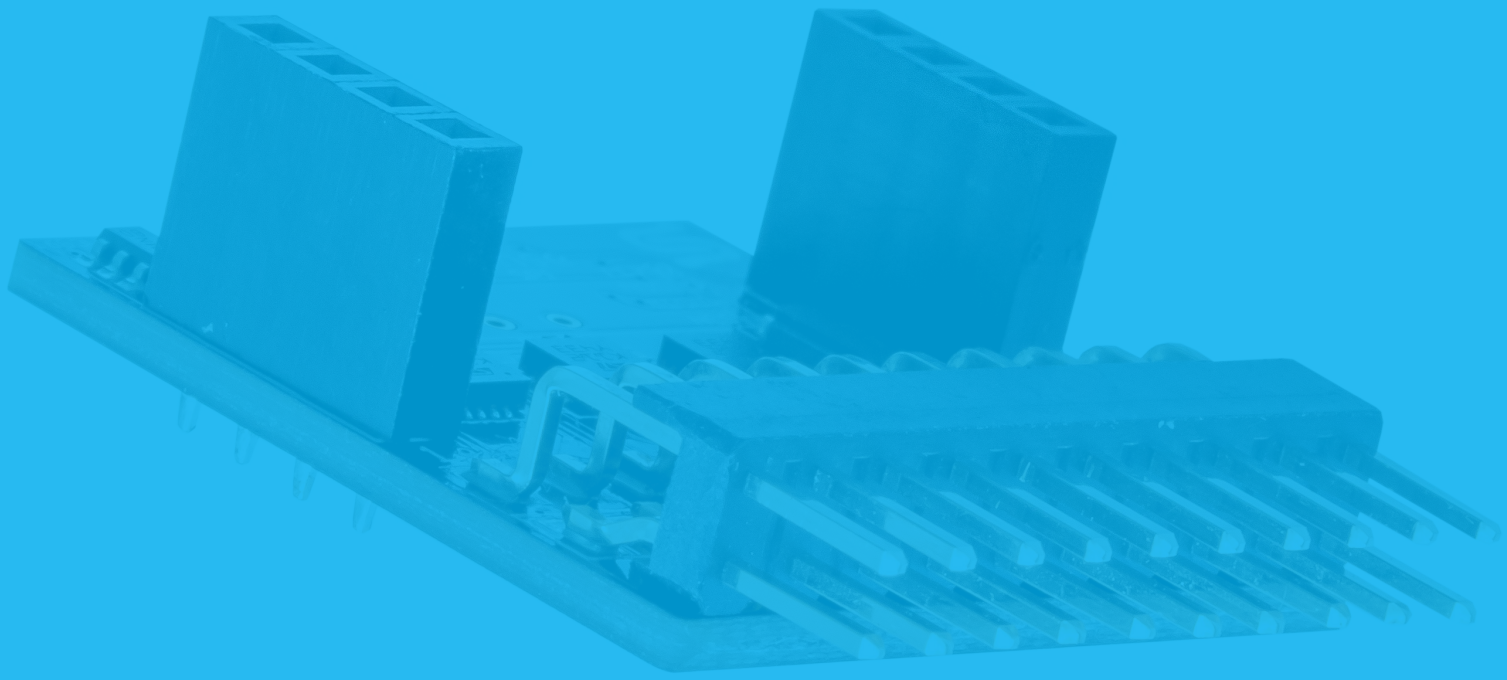


API LIBRARY OVERVIEW

STRETCHSENSE



StretchSense provides APIs (Application Program Interfaces) for Android, iOS, and macOS. APIs are a set of protocols and functions that define how system components interact with each other and can be used like building blocks for software. Our APIs enable communication between a StretchSense circuit and a BLE-enabled device.

The APIs predefine how communication occurs, and offer an easy way to interact with our sensors, so you don't need to spend time figuring out all the implementation details. Not having to devise a communication method and data capture solution from scratch allows you to more quickly and easily integrate our stretch sensor electronics into your application.

Our iOS and Android libraries have been developed using Bluetooth 4.0 (BLE). A list of compatible devices can be found on their respective StretchSense GitHub pages https://github.com/StretchSense. The macOS API was developed using XCode 7.3 with Swift 3. Similar to the iOS API, it demonstrates how to connect one or more circuits to a BLE-enabled device and has been developed using Bluetooth 4.0.

There are three primary functions included in all our APIs which provide the means needed to communicate with our sensors.

- **Scan**: looks for a StretchSense circuit to connect to
- **Connect**: connects the host device (the device you are using to communicate with the sensor) with the StretchSense circuit
- **Disconnect**: disconnects the StretchSense circuit from the host device

Function: Scan/Connect/Disconnect

```
startBluetooth()
```

Initialisation of the **Manager**

NOTE
Must be the first function called, to check if bluetooth is enabled and initialise the manager

Declaration

```
SWIFT
func startBluetooth()
```

Return Value
Nothing

```
startScanning()
stopScanning()
connectToPeripheralWithUUID(_:)
```

StretchSensePeripheral

```
class StretchSensePeripheral
```

This class defines a single Fabric Evaluation StretchSense sensor

NOTE
Each sensor is defined by:

- A universal unique identifier UUID
- A unique number, based on the order when the sensor is conn
- A unique color, choose with his unique number
- A current capacitance value
- An array of the 50 previous capacitance raw values
- An array of the 50 previous capacitance averaged values

The sensor class has been designed so that attributes such as identifying number and capacitance values can be easily retrieved. Other functions provided include save and share data; get and set for all parameters are also provided. These allow more advanced options regarding data acquisition and processing but are not needed to get up and running with our sensors.

Our APIs are comprehensively documented to make it as straightforward as possible to establish a connection with our sensors and your preferred platform. We separately document each call in our API, with notes on parameters; examples are also provided for each platform to give you an idea of the possibilities our APIs provide.

All components of the Android and iOS APIs are documented in HTML files included in their respective libraries. These files contain details such as explanations of how the different classes are set up. They include descriptions and declarations for all the included functions, their associated parameters and return values, and also covers all the variable declarations.

An HTML file is not available for macOS, but the iOS file can be used as a reference. However, while both iOS and macOS share the same language, there will be slight differences in the implementation due to their interface differences. Across all the platforms, commenting in the code itself also offers instructions, explanations, and guidance on how to use the APIs.

The iOS API includes two example projects that demonstrate how to use the library. The simplest example shows how to connect with a single circuit. The other examples show how to connect to multiple circuits simultaneously. The library was developed using XCode 7.3 with Swift 3 and requires an operating system of iOS 8.0 or newer. Once you have created your application in XCode, import the StretchSense iOS library to get started with the functions. A quick-start guide covering how to initialise the library, scan, connect, and read data from the sensor is also included in the library as a markdown file.

https://github.com/StretchSense/iOS-Library

The Android API contains a project template pre-implemented with the Android API and three example projects. The first demonstrates how to connect manually with a single sensor circuit. The second is similar, demonstrating how to connect to a single sensor circuit automatically. The third example demonstrates how to connect up to 10 circuits simultaneously. This example displays the list of sensors available or connected and differentiates between the sensors by assigning them a unique number and colour.

This library was developed using Android Studio 2.1.2 in Java with a gradle defaultConfig minSdkVersion 19 and a targetSdkVersion 22. It requires a minimum operating system of Android 4.3. The project template provided makes getting started simple. Similar to the iOS API library, the Android API library also includes a quick start guide as part of the markdown file.

https://github.com/StretchSense/Android-Library

If you're an experienced developer and would like added control over the sensor connection, we recommend extending the existing background functions. Rewriting the background functions themselves is not advisable as they are the foundation of the API and altering them may cause unexpected changes.

These APIs offer independent, reusable functions that can be assembled into more complex applications. We hope our documentation allows for ease of use, providing you with support so that the prototyping process is as simple and efficient as possible. If you have any questions, please don't hesitate to get in touch at support@stretchsense.com

StretchSense™

sales@stretchsense.com
support@stretchsense.com
www.stretchsense.com

NZ DDI +64 9 634 1927
US DDI +1 415 800 1003