OxySound Documentation

Table of contents

1. Welcome to OxySound

1.1 Introduction

OxySound sends data seamlessly over sound waves to enhance end-user experiences and add value to existing hardware. It uses any existing speaker or microphone, and is configurable to use audible or inaudible near-ultrasonic frequencies.

Welcome

OxySound is designed for simplicity, eliminating connectivity headaches and simplifying everyday tasks like connecting to Wi-Fi networks, sharing contact details, and making peer-to-peer payments.

What

1.2 Why ?

You can find speakers in most places, and that makes this technology cheap and can be used in cases where communication is restricted.

2. How it works

2.1 Overview

OxySound enables your apps to send and receive information using sound. We encode your data as an audio signal, which can be transmitted by any device with a speaker and received by any device with a microphone with the OxySound SDK. It is designed to be robust over distances of several metres, in noisy, everyday environments.

As the transmission takes place entirely via audio signals, no internet connection or prior pairing is required, and any device within hearing range can receive the data.

Audio can be generated on-device from a dynamic data payload, or recorded as an audio file for later playback with or without a compainion audio track.

3. Use Cases

3.1 Audible vs. near-ultrasonic

3.1.1 Introduction

There are situations when data over sound is more viable than the standard solution over radio waves.

The technology could be used where electro magnetic signals could interfere and are as such prohibited.

It can be used as replacement for NFC and Bluetooth technology, a example is using sound data as a protocol for bus ticket validation.

From a security perspective it can be used as authentication mechanism to login to websites or even physical devices, arguing that they are less likely hacked since physical presence of listening device or person is needed for sniffing data

4. SDK

4.1 Getting Started - iOS

4.1.1 Framework

Import the SDK into your project by dragging and dropping the .framework file into your Xcode project. Set ‘Copy items if needed’ if the framework is not already within your project folder.

Add the framework to Linked Frameworks and Libraries Go to the Project Settings, under the General tab, and add Oxy.framework to "Linked Frameworks and Libraries".

4.1.2 Add a microphone privacy statement

This text is displayed when your app first launches and asks the user for microphone permission, and should describe why you are requesting access to the microphone. The request message should be short and specific and include an example of how the microphone data will be used, for example

We need access to the microphone to receive messages from nearby devices using sound.

Under the Info tab in the Project Settings, add a Custom iOS Target Property called "Privacy - Microphone Usage Description".

4.1.3 Disable Bitcode for the SDK

Under the Build Settings tab in the Project Settings, under Build Options, set "Enable Bitcode" to No.

Add a bridging header to the framework [Swift Only] Add a new Objective-C file to your Swift project. Xcode will prompt if you want to configure your project with a bridging header. Press yes. Once the bridging header is added to your project, you can delete the Objective-C file if you want to.

Inside the bridging header, import the Oxy framework. This will make it available throughout your project’s Swift files.

4.1.4 Now for the code...

Include the SDK and its associated configuration header:

```
import Oxy
```

And instantiate the SDK

(A key will required in the future)

```
let OxyManager = Oxy.instance()

oxyManager?.delegate=self
```

4.2

To receive this data on a second device, simply implement the delegate.

```
class ViewController: UIViewController, oxyDelegate  {
}


func oxyId(with oxy_id: String!) {

DispatchQueue.main.async {

        //Do some magic

    }
}
```

Start

```
oxyManager!.listen()
```

You are now ready to start using OxySound in your own application.

4.2.1 Advanced usage

Preselected frequencies and tone seperations

```
    func custom(freq: Float, sep: Int) {
        oxyManager!.setCustomBaseFreq(freq, withSeparation: Int32(sep))
        oxyManager!.configure(oxyManager, with: CUSTOM)
        self.oxyManager!.listen()
    }
```

To limit user interaction with broadcasts a set distance can be set and when below this distance the SDK will respond.

```
if(self.oxyManager!.distanceVol() < self.amax ) {

}
```

Send audio from the SDK. Type options are 0/1 or 2 this relates to should the broadcast be mixed with an external media file or not

Broadcast tone with SDK configuration

```
func sendAudio(){
        checkDeviceVolumeLevel()
        oxyManager!.play("829450000", withType: 0)
    }
```

Audible R2D2 tone

```
func sendAudio(){
        checkDeviceVolumeLevel()
        oxyManager!.play("829450000", withType: 1)
    }
```

External media

```
let soundURLg = Bundle.main.url(forResource: "storm", withExtension: "wav")

...
  oxyManager!.play("929450000", withType: 2)
        SPAlert.present(title: "On device audio mixed with message", message: "929450000", image: UIImage(imageLiteralResourceName: "stormz"))
```

4.3 Getting Started - Android

4.3.1 AAR

Copy the oxy.aar file to your app/libs directory. Add the following to the dependencies block of your Module build.gradle Gradle script.

To instruct Gradle where to find the local .aar file, add flatDir section to the repositories block. (You'll need to add a repositories block if one does not already exist).

```
repositories {
  flatDir {
    dirs 'libs'
  }
}
```

4.3.2 Permissions

Declare your app's audio permissions Add the following to your AndroidManifest.xml, inside the bottom of the element.

```
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />

<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

SDK requires at minimum of Android 5.0.x which is Android API level 26.

4.3.3 Import the SDK

```
import com.oxy.AndroidOxyCore.OxyCore
import com.oxy.AndroidOxyCore.OxyCoreEvent
```

Instantiate the SDK

```
private lateinit var SDKOxyCore: OxyCore


class MainActivity : AppCompatActivity(), OxyCoreEvent {

override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        //Provide Context for Callback
        SDKOxyCore = OxyCore(this)
    }
}
```

```
 //Callback
    override fun IdWith(Id: String) {

    }
```

Request microphone permissions and start the engine

```
    override fun onResume() {
        super.onResume()
        SDKOxyCore.Listen()
    }
```

4.3.4 States

Pause

```
    override fun onPause() {
        super.onPause()
        SDKOxyCore.Stop()
    }
```

Destory the instance

Stop and close the SDK when activity is destroyed.

In order to make sure the SDK will be closed, we need to call the close method to empty the memory and to delete the instance when the activity is destroyed.

```
override fun onDestroy() {
        super.onDestroy()
        SDKOxyCore.Stop()
    }
```

You are now ready to start using OxySound in your own application.

4.4 Getting Started - PI/Linux

Audio

The OxySound SDK for Arm Cortex processors allows you to send and receive data-over-sound from within your embedded C/C++ application for encoding and decoding of OxySound signals.

4.4.1 Requirements

To build an embedded system that uses the OxySound Pi SDK, you will need the following components.

Arm Cortex-A72 architecture variant CPU For input, a microphone. We recommend the ReSpeaker 2 mics Hat. For output, a loudspeaker.

4.5 Getting Started - MacOS

4.5.1 Framework

Import the SDK into your project by dragging and dropping the .framework file into your Xcode project. Set ‘Copy items if needed’ if the framework is not already within your project folder.

Add the framework to Linked Frameworks and Libraries Go to the Project Settings, under the General tab, and add Oxy.framework to "Linked Frameworks and Libraries".

4.5.2 Add a microphone privacy statement

This text is displayed when your app first launches and asks the user for microphone permission, and should describe why you are requesting access to the microphone. The request message should be short and specific and include an example of how the microphone data will be used, for example

We need access to the microphone to receive messages from nearby devices using sound.

Under the Info tab in the Project Settings, add a Custom iOS Target Property called "Privacy - Microphone Usage Description".

4.5.3 Disable Bitcode for the SDK

Under the Build Settings tab in the Project Settings, under Build Options, set "Enable Bitcode" to No.

Add a bridging header to the framework [Swift Only] Add a new Objective-C file to your Swift project. Xcode will prompt if you want to configure your project with a bridging header. Press yes. Once the bridging header is added to your project, you can delete the Objective-C file if you want to.

Inside the bridging header, import the Oxy framework. This will make it available throughout your project’s Swift files.

4.5.4 Now for the code...

Include the SDK and its associated configuration header:

```
import Oxy
```

And instantiate the SDK

(A key will required in the future)

```
let OxyManager = Oxy.instance()

oxyManager?.delegate=self
```

4.6

To receive this data on a second device, simply implement the delegate.

```
class ViewController: UIViewController, oxyDelegate  {
}


func oxyId(with oxy_id: String!) {

DispatchQueue.main.async {

        //Do some magic

    }
}
```

Start

```
oxyManager!.listen()
```

You are now ready to start using OxySound in your own application.

4.7 Getting Started - WebAssembly

The OxySound WebAssembly SDK brings our technology to the web, allowing you to send and receive data over sound in the browser, using a simple JavaScript interface. With the WebAssembly SDK you can integrate OxySound into web pages and apps, and the exact same code can be executed on many different devices, both desktop and mobile.

The SDK never sends audio data to the cloud, running all of the audio processing locally on your device.

4.7.1 Known Issues

1). Android and iOS devices do not detect frequencies above 8kHz and 11kHz respectively in the browser. However ultrasonic protocols are an option for desktop only applications.

2). The SDK will not work on iOS Chrome, as getUserMedia is not supported in a WKWebView. See the open radar bug report.

4.8 Getting Started - Ardunio

Audio

The OxySound SDK for Arm Cortex processors allows you to send and receive data-over-sound from within your embedded C/C++ application. It harnesses the powerful in-built FPU and optimised CMSIS-DSP library to enable on-chip encoding and decoding of OxySound signals.

4.8.1 Requirements

To build an embedded system that uses the OxySound Arm SDK, you will need the following components.

Arm Cortex-M4 or Arm Cortex-M7 CPU Audio ADC/DAC capable of at least 16kHz operation For input, a microphone. We recommend a MEMS mic, ideally with a flat frequency response. For output, a loudspeaker.

4.9 Getting Started - Unity

Audio

5. Examples

5.1 iPhone App

```
import Oxy
import UIKit
import AVFoundation

/**
- Note: OxySound Framework Example

 This class contains common delegate, methods and constants that are used in all application controllers

The constructor`OxyManager` contains all the controls of the OxySound SDK

**Basic Example SDK configuration:**
```

let OxyManager = Oxy.instance() oxyManager?.delegate=self oxyManager!.listen()

- Note: The method `OxyManager.delegate = self()` sets the callback for `OK` decodes
- The mehtod `OxyManager.instance()` initialize the SDK.
- The method `OxyManager.listen()` starts the SDK into listening mode the default
- The method `OxyManager.stop()` starts the SDK into listening mode the default

```
 */

class ViewController: UIViewController, oxyDelegate  {

    @IBOutlet weak var wipbtn: UIButton!
    //Instance
    let oxyManager = Oxy.instance()
    let amax:Float = -110  //door



    //MARK: Flow control of payload
    var tog:Bool = false
    var payload:String = ""

    override func viewDidLoad() {
        super.viewDidLoad()

        //Assign delegate
       oxyManager?.delegate=self

        //Start engine
       oxyManager!.listen()

    }
    //MARK: Still processing audio push to new thread
    func oxyId(with oxy_id: String!) {
        if(self.payload != oxy_id){
            self.payload = oxy_id
            self.tog = true
        }
        DispatchQueue.main.async {
            switch self.payload {
            case "91876":
                 self.view.backgroundColor = self.random()
                if(self.tog == true) {
                            /* Future use
                        self.tog = false
                         */
                          self.addimage()
                     }
            case "88888":
                self.toggleTorch(on: true)
                self.toggleTorch(on: true)
                self.toggleTorch(on: false)
            case "12345":
                //self.view.backgroundColor = self.UIColorFromHex(rgbValue: 0xD61E1E,alpha: 1)
                //self.view.backgroundColor = self.random()
                //self.showToast(oxy_id)
                if(self.oxyManager!.distanceVol() < self.amax ) {
                    print(self.oxyManager!.distanceVol())
                    self.showToast("Move device closer")
                    self.view.backgroundColor =
                        // Red
                        self.UIColorFromHex(rgbValue: 0xD61E1E,alpha: 1)
                }else{
                    self.showToast("Device within range")
                    // Green
                    self.view.backgroundColor = self.UIColorFromHex(rgbValue: 0x228B22,alpha: 1)
                }
            default:
                self.view.backgroundColor = self.random()
                self.showToast(oxy_id)
                //print(oxy_id as Any)
            }
        }
}
    func random() -> UIColor {
        return UIColor(red: .random(in: 0...1),
                       green: .random(in: 0...1),
                       blue: .random(in: 0...1),
                       alpha: 1.0)
    }



    func addimage(){
        var number = Int.random(in: 0 ..< 900)
        var number2 = Int.random(in: 0 ..< 900)
        //Create image view simply like this.
        let imgView = UIImageView()

        imgView.frame = CGRect(x: number, y: number2, width: 200, height: 200)
        imgView.image = UIImage(named: "uv")//Assign image to ImageView
        imgView.imgViewCorners()
        view.addSubview(imgView)//Add image to our view
    }

    func UIColorFromHex(rgbValue:UInt32, alpha:Double=1.0)->UIColor {
        let red = CGFloat((rgbValue & 0xFF0000) >> 16)/256.0
        let green = CGFloat((rgbValue & 0xFF00) >> 8)/256.0
        let blue = CGFloat(rgbValue & 0xFF)/256.0

        return UIColor(red:red, green:green, blue:blue, alpha:CGFloat(alpha))
    }
    func toggleTorch(on: Bool) {
        guard
```

```
            let device = AVCaptureDevice.default(for: AVMediaType.video),
            device.hasTorch
        else { return }

        do {
            try device.lockForConfiguration()
            device.torchMode = on ? .on : .off
            device.unlockForConfiguration()
        } catch {
            print("Torch could not be used")
        }
    }
}

extension UIImageView {
    //If you want only round corners
    func imgViewCorners() {
        layer.cornerRadius = 10
        layer.borderWidth = 1.0
        layer.masksToBounds = true
    }
}
```

5.2 iPhone App

```
package com.example.oxysound

import android.Manifest
import android.content.pm.PackageManager
import android.graphics.Color
import android.os.Bundle
import android.util.Log
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import androidx.core.app.ActivityCompat
import com.oxy.AndroidOxyCore.OxyCore
import com.oxy.AndroidOxyCore.OxyCoreEvent
import kotlinx.android.synthetic.main.activity_main.*
import java.util.*

private lateinit var SDKOxyCore: OxyCore

private const val REQUEST_RECORD_AUDIO = 1
private var t = 1

class MainActivity : AppCompatActivity(), OxyCoreEvent {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        //Provide Context for Callback
        SDKOxyCore = OxyCore(this)
    }

    override fun onPause() {
        super.onPause()
        //Pause SDK
        SDKOxyCore.Stop()
    }

    override fun onDestroy() {
        super.onDestroy()c
    }

    private fun distance() {
        SDKOxyCore.distancevol().toString()
    }

    //Callback
    override fun IdWith(Id: String) {
        when (Id) {
            "qa034" -> {
                val rnd = Random()
                val color = Color.argb(255, rnd.nextInt(256), rnd.nextInt(256), rnd.nextInt(256))

                mainlayout.setBackgroundColor(color)
            }
            else -> {
                distance()
                val rnd = Random()
                Toast.makeText(this@MainActivity, Id, Toast.LENGTH_SHORT).show()
                val color = Color.argb(255, rnd.nextInt(256), rnd.nextInt(256), rnd.nextInt(256))
                mainlayout.setBackgroundColor(color)
            }
        }

    }

    // Request microphone permissions
    override fun onResume() {
        super.onResume()
        SDKOxyCore.Listen()
    }
}
```

6. Samples

6.1 Sample audio

6.1.1 Mixed with audio file

Audio

6.1.2 Youtube / Netflix intergration

Netflix