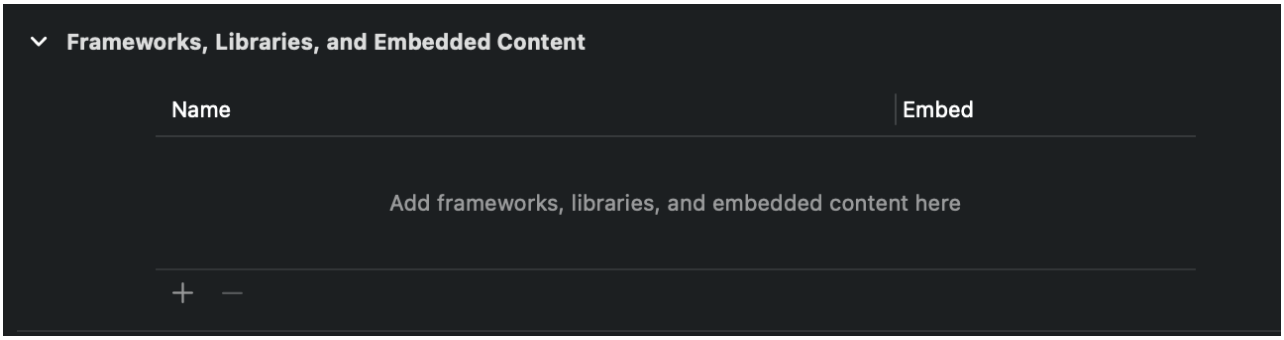
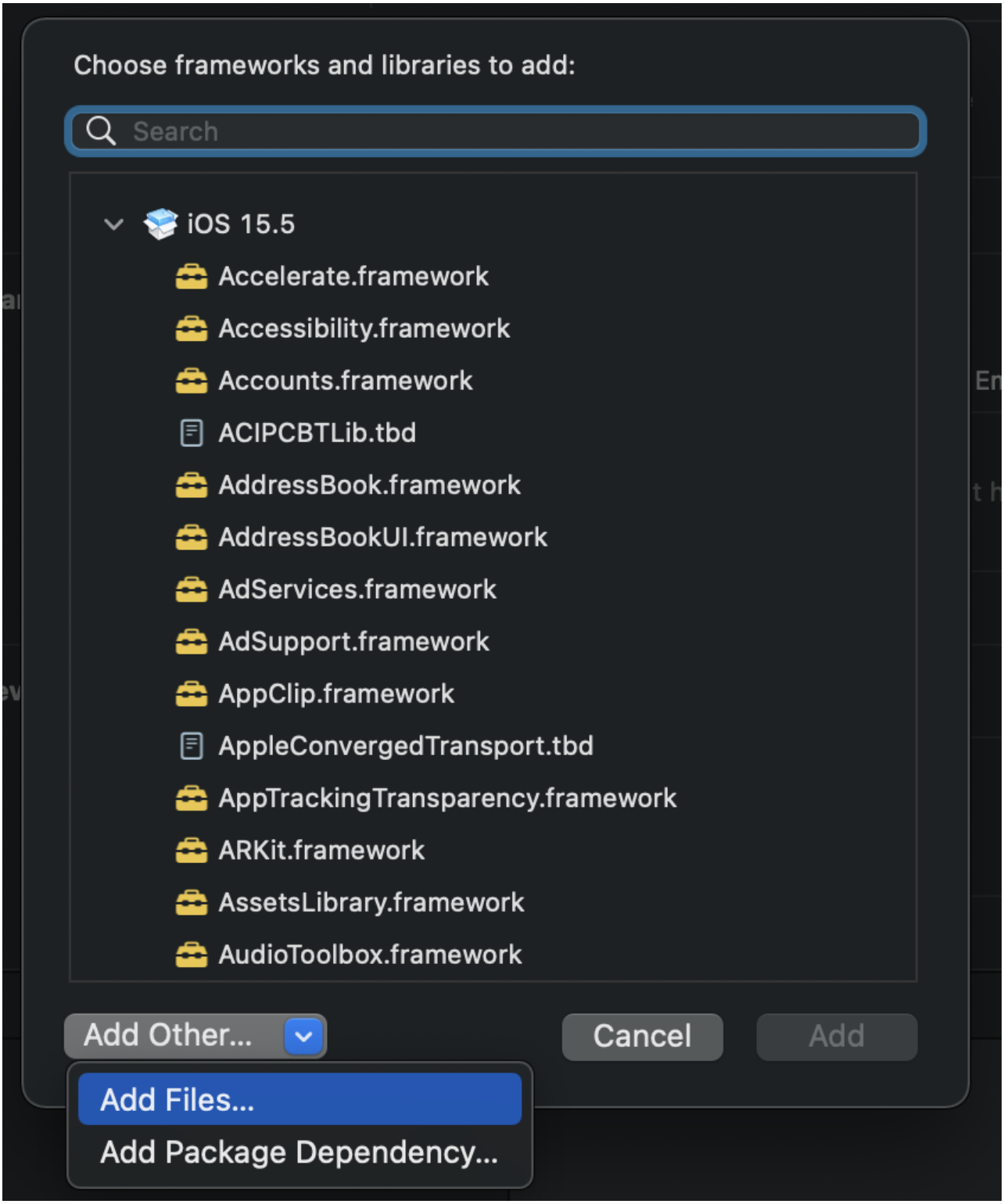
How to integrate transdermal-vitals-scanner library to iOS app

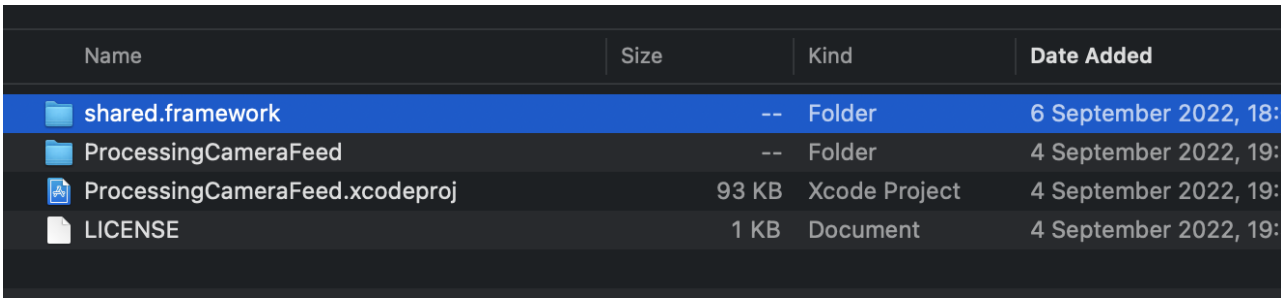
1. Add “shared.framework.zip“ file (on mac you can see this as a folder) to your project root folder and unzip it there. Then add it as a framework in XCode project:
   a. Open your project in XCode
   b. Go to “Targets“ “General“ tab
   c. Scroll to “Frameworks, Libraries, and Embeded Content“



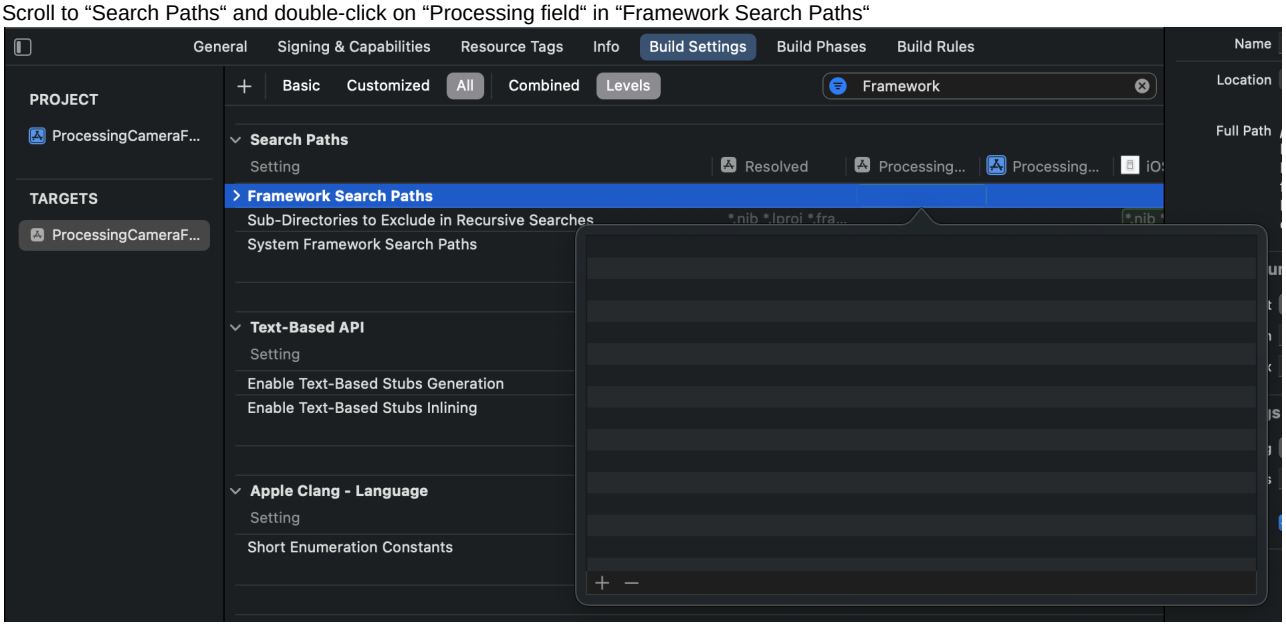
   d. Click on “+” and choose “Add Files …” option



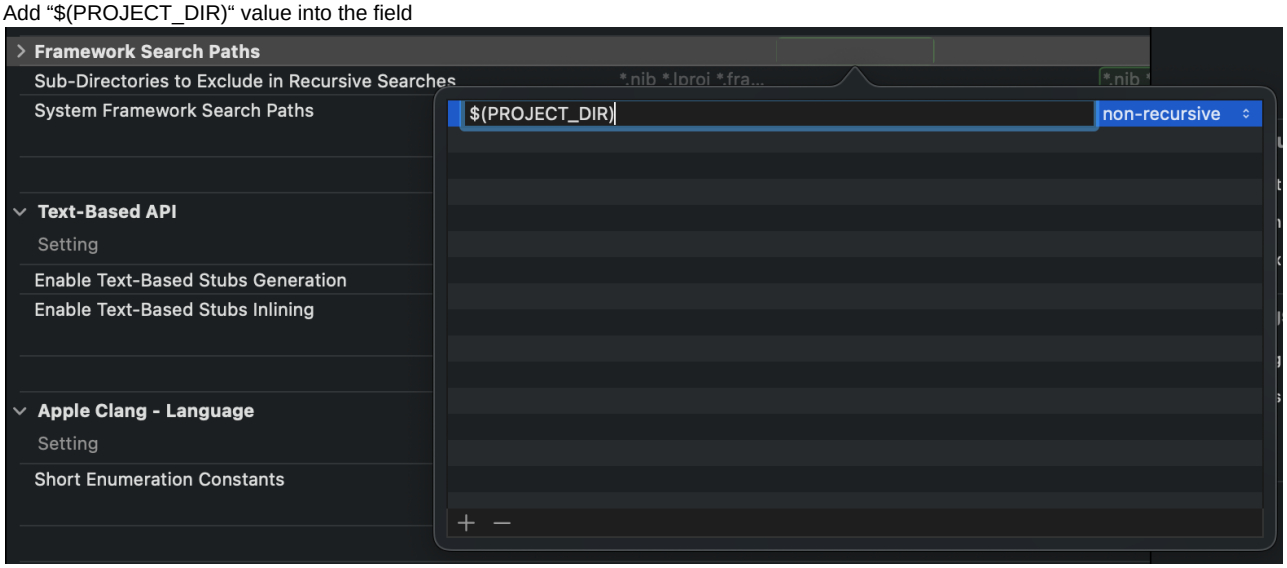
e. Choose the “shared.framework“ file



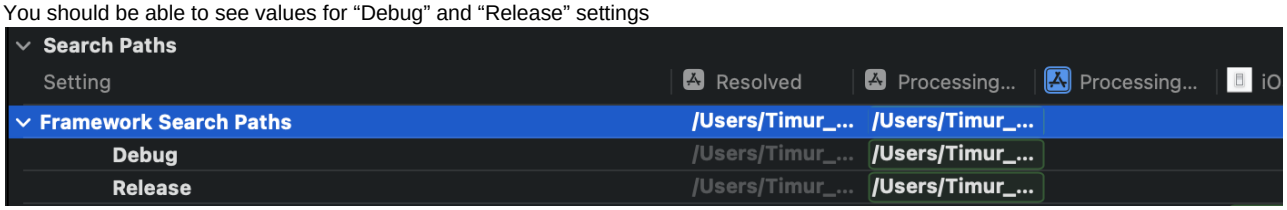
2. Add path to the “shared.framework” in your project
   a. Go to “Target“ “Build settings“ tab
   b. Scroll to “Search Paths“ and double-click on “Processing field“ in “Framework Search Paths“



   c. Click on “+” sign
   d. Add “$(PROJECT_DIR)“ value into the field

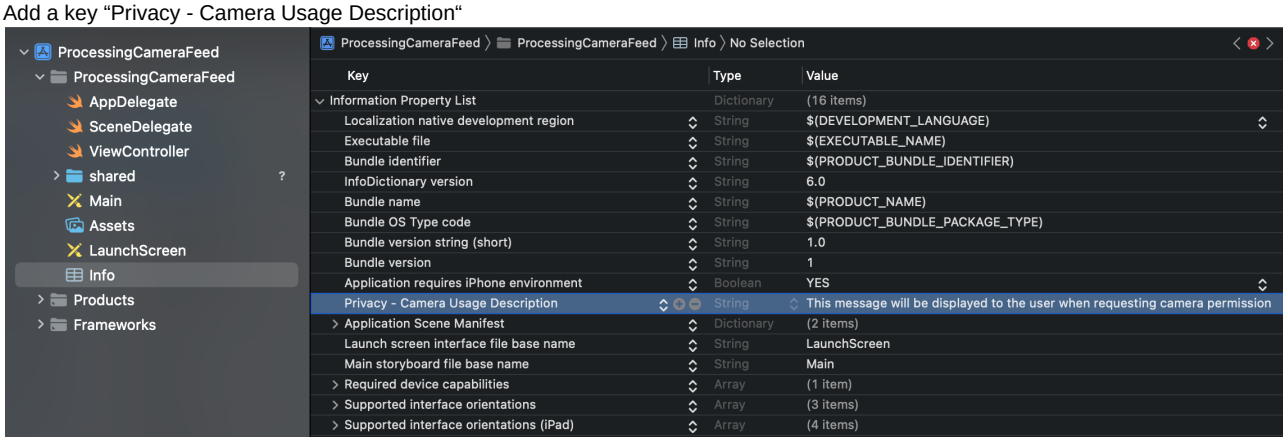


   e. Press Enter and click on “Framework Search Paths”
   f. You should be able to see values for “Debug” and “Release” settings



   g. Now you should be able to call functions from the “shared.framework”
3. To work with the framework in the code you just need to import it using “import shared“

4. Allow application to work with camera
    a. Go to “Info“
    b. Add a key “Privacy - Camera Usage Description“



5. There is one method `processImageRGB` in `VitalSignsProcessorIOS` which you should use in your iOS app. It accepts the average colour intensity for Red, Green, Blue taken from Image (video camera frame), and some anthropometric data. It returns `ProcessStatus` enum, with 5 statuses:
    a. `RED_INTENSITY_NOT_ENOUGH("Not good red intensity to process. Should start again"),`
    b. `MEASUREMENT_FAILED("Measurement Failed. Should Start again"),`
    c. `IN_PROGRESS("Processing in progress"),`
    d. `PROCESS_FINISHED("Processing finished"),`
    e. `NEED_MORE_IMAGES("Need more images to process")`
6. In your codebase with iOS components you should implement a class where you will be working with a camera and delegate `AVCaptureVideoDataOutputSampleBufferDelegate` which works with camera and process frames in `captureOutput` method. Example could be found below:
    a. Weight and Height are in centimetres; age in years; “gen“, i.e. gender: 1 - male; 2 - female

```
import UIKit
import AVFoundation
import VideoToolbox
import shared
import Accelerate


class ViewController: UIViewController,
AVCaptureVideoDataOutputSampleBufferDelegate{

    private let captureSession = AVCaptureSession()
    private lazy var previewLayer: AVCaptureVideoPreviewLayer = {
        let preview = AVCaptureVideoPreviewLayer(session: self.
captureSession)
        preview.videoGravity = .resizeAspect
        return preview
    }()
    private let videoOutput = AVCaptureVideoDataOutput()
    private let rrProcessor = RespiratoryRateProcessorIOS()
    private let hrProcessor = HeartRateProcessorIOS()
    private let o2Processor = O2ProcessorIOS()
    private let bpProcessor = BloodPressureProcessorIOS(user: User
(height: 188.0, weight: 76.0, age: 25, gen: 1))
    private let vsProcessor = VitalSignsProcessorIOS(user: User(height:
188.0, weight: 76.0, age: 25, gen: 1))
```

```
    override func viewDidLoad() {
        super.viewDidLoad()
        self.addCameraInput()
        self.addPreviewLayer()
        self.addVideoOutput()
        self.toggleTorch(on: true)
        self.captureSession.startRunning()
        self.captureSession.sessionPreset = AVCaptureSession.Preset.low;
        self.toggleTorch(on: true)
    }

    override func viewDidLayoutSubviews() {
        super.viewDidLayoutSubviews()
        self.previewLayer.frame = self.view.bounds
    }

    func captureOutput(_ output: AVCaptureOutput,
                       didOutput sampleBuffer: CMSampleBuffer,
                       from connection: AVCaptureConnection) {


        let imageBuffer = CMSampleBufferGetImageBuffer(sampleBuffer)
        CVPixelBufferLockBaseAddress(imageBuffer!,
CVPixelBufferLockFlags(rawValue: 0))
        let width = CVPixelBufferGetWidth(imageBuffer!)
        let height = CVPixelBufferGetHeight(imageBuffer!)
        let src_buff = CVPixelBufferGetBaseAddress(imageBuffer!)
        let dataBuffer = src_buff!.assumingMemoryBound(to: UInt8.self)
// *
        var bytesPerRow = CVPixelBufferGetBytesPerRow(imageBuffer!);

        CVPixelBufferUnlockBaseAddress(imageBuffer!,
CVPixelBufferLockFlags(rawValue: 0))

        let numberOfPixels = height * width
        var greenVector:[Float] = Array(repeating: 0.0, count:
numberOfPixels)
        var blueVector:[Float] = Array(repeating: 0.0, count:
numberOfPixels)
        var redVector:[Float] = Array(repeating: 0.0, count:
numberOfPixels)

        vDSP_vfltu8(dataBuffer, 4, &blueVector, 1, vDSP_Length
(numberOfPixels))
        vDSP_vfltu8(dataBuffer+1, 4, &greenVector, 1, vDSP_Length
(numberOfPixels))
        vDSP_vfltu8(dataBuffer+2, 4, &redVector, 1, vDSP_Length
(numberOfPixels))
        var redAverage:Float = 0.0
        var blueAverage:Float = 0.0
```

```
        var greenAverage:Float = 0.0

        vDSP_meamgv(&redVector, 1, &redAverage, vDSP_Length
(numberOfPixels))
        vDSP_meamgv(&greenVector, 1, &greenAverage, vDSP_Length
(numberOfPixels))
        vDSP_meamgv(&blueVector, 1, &blueAverage, vDSP_Length
(numberOfPixels))

        do {
            let status = try vsProcessor.processImageRGB(red: Double
(redAverage), green: Double(greenAverage), blue: Double(blueAverage)).
name
                //print("red avg=", redAverage)
            if(status == "PROCESS_FINISHED") {
                print(vsProcessor.o2)
                print(vsProcessor.Beats)
                print(vsProcessor.Breath)
                print(vsProcessor.DP)
                print(vsProcessor.SP)
            }else{
                print(status)
            }
        }
        catch {
            // Couldn't create audio player object, log the error
            print("Error=\(error)")
        }

    }

    @objc func savedImage(_ im:UIImage, error:Error?, context:
UnsafeMutableRawPointer?) {
        if let err = error {
            print(err)
            return
        }
        print("success")
    }

    func getArrayOfBytesFromImage(imageData:NSData) -> Array<UInt8>
    {

      // the number of elements:
      let count = imageData.length / MemoryLayout<Int8>.size

      // create array of appropriate length:
      var bytes = [UInt8](repeating: 0, count: count)

      // copy bytes into array
```

```
      imageData.getBytes(&bytes, length:count * MemoryLayout<Int8>.size)

      var byteArray:Array = Array<UInt8>()

      for i in 0 ..< count {
        byteArray.append(bytes[i])
      }

      return byteArray


    }


    private func addCameraInput() {
        let device = AVCaptureDevice.default(for: .video)!
        let cameraInput = try! AVCaptureDeviceInput(device: device)
        self.captureSession.addInput(cameraInput)
    }

    private func addPreviewLayer() {
        self.view.layer.addSublayer(self.previewLayer)
    }

    private func addVideoOutput() {
        self.videoOutput.videoSettings =
[(kCVPixelBufferPixelFormatTypeKey as NSString) : NSNumber(value:
kCVPixelFormatType_32BGRA)] as [String : Any]
        self.videoOutput.setSampleBufferDelegate(self, queue:
DispatchQueue(label: "my.image.handling.queue"))
        self.captureSession.addOutput(self.videoOutput)
    }

    func convert(cmage: CIImage) -> UIImage {
         let context = CIContext(options: nil)
         let cgImage = context.createCGImage(cmage, from: cmage.extent)!
         let image = UIImage(cgImage: cgImage)
         return image
    }

    func toggleTorch(on: Bool) {
        guard let device = AVCaptureDevice.default(for: .video) else {
return }

        if device.hasTorch {
            do {
                try device.lockForConfiguration()

                if on == true {
                    device.torchMode = .on
```

```
                } else {
                    device.torchMode = .off
                }

                device.unlockForConfiguration()
            } catch {
                print("Torch could not be used")
            }
        } else {
            print("Torch is not available")
        }
    }

}
```

7. Results could be accessed from the *Processor object fields (for example, `vsProcessor.o2, vsProcessor.Breath, vsProcessor.Beats, vsProcessor.SP, vsProcessor.DP`)