Guide for compiling the cisst library for iOS:

1. Modify the CMakeList.txt to include new settings specific for iOS. Add the following line to the end of the file:
   ```
   INCLUDE(iphone.cmake)
   ```

2. Create the file iphone.cmake in the same folder. This file essentially sets the paths for the necessary compilers and architectures. Modify the versions as necessary. Code:
   ```
   SET (CMAKE_SYSTEM_PROCESSOR arm CACHE STRING "" FORCE)
   SET_PROPERTY(GLOBAL PROPERTY TARGET_SUPPORTS_SHARED_LIBS TRUE)

   SET (SDKVER "4.2" CACHE STRING "" FORCE)
   SET (DEVROOT
   "/Developer/Platforms/iPhoneOS.platform/Developer" CACHE
   STRING "" FORCE)
   SET (SDKROOT "${DEVROOT}/SDKs/iPhoneOS${SDKVER}.sdk" CACHE
   STRING "" FORCE)
   SET (CMAKE_INSTALL_PREFIX "../ipadlib" CACHE PATH "" FORCE)
   SET (CMAKE_OSX_SYSROOT "${SDKROOT}" CACHE PATH "" FORCE)
   SET (CMAKE_OSX_ARCHITECTURES "armv7" CACHE STRING "" FORCE)

   SET (CMAKE_C_COMPILER "${DEVROOT}/usr/bin/gcc-4.2" CACHE
   FILEPATH "" FORCE)
   SET (CMAKE_CXX_COMPILER "${DEVROOT}/usr/bin/g++-4.2" CACHE
   FILEPATH "" FORCE)

   SET (CMAKE_C_FLAGS "-std=c99" "-x objective-c" CACHE STRING ""
   FORCE)
   SET (CMAKE_C_FLAGS_DEBUG ${CMAKE_C_FLAGS} "-DDEBUG=1" "-ggdb"
   CACHE STRING "" FORCE)
   SET (CMAKE_C_FLAGS_RELEASE ${CMAKE_C_FLAGS} "-DNDEBUG=1" CACHE
   STRING "" FORCE)
   SET (CMAKE_C_FLAGS_RELWITHDEBINFO ${CMAKE_C_FLAGS} "-
   DNDEBUG=1" "-ggdb" CACHE STRING "" FORCE)

   SET (CMAKE_CXX_FLAGS "-x objective-c++" CACHE STRING "" FORCE)
   SET (CMAKE_CXX_FLAGS_DEBUG ${CMAKE_CXX_FLAGS} "-DDEBUG=1" "-
   ggdb" CACHE STRING "" FORCE)
   SET (CMAKE_CXX_FLAGS_RELEASE ${CMAKE_CXX_FLAGS} "-DNDEBUG=1"
   CACHE STRING "" FORCE)
   SET (CMAKE_CXX_FLAGS_RELWITHDEBINFO ${CMAKE_CXX_FLAGS} "-
   DNDEBUG=1" "-ggdb" CACHE STRING "" FORCE)

   ADD_DEFINITIONS("-arch armv7")
   ADD_DEFINITIONS("-pipe")
   ADD_DEFINITIONS("-no-cpp-precomp")
   ADD_DEFINITIONS("--sysroot=${SDKROOT}")
   ADD_DEFINITIONS("-miphoneos-version-min=${SDKVER}")

   INCLUDE_DIRECTORIES(SYSTEM "${SDKROOT}/usr/include")
   LINK_DIRECTORIES("${SDKROOT}/usr/lib")

   SET (CMAKE_FIND_ROOT_PATH "${SDKROOT}"  CACHE PATH "" FORCE)
   SET (CMAKE_FIND_ROOT_PATH_MODE_PROGRAM BOTH)
   SET (CMAKE_FIND_ROOT_PATH_MODE_LIBRARY ONLY)
   SET (CMAKE_FIND_ROOT_PATH_MODE_INCLUDE ONLY)

   SET (iPhone 1)
   SET (iPhoneOS 1)
   SET (iPhoneOS_VERSION ${SDKVER})
   ```

3. Use cmake to compile the source code with the newly added cmake settings from the previous step. When `make install` is performed, the built libraries and include files should be in a folder named `ipadlib`.

4. In XCode, add the compiled libaries (.a files in the `ipadlib/lib` folder) to the target application.

5. In the Target Info, add the folder `ipadlib/lib` to the "Library Search Paths" (should be done automatically with step 4).

6. In the Target Info, add the folder `ipadlib/include` to the "Header Search Paths".

7. Build the target as you would normally for a project.

Important Notes:

1. While there is a version of ICE designed for iOS, you must use the C++ version. Otherwise, ICE won't be able to communicate with the cisst library.

2. All class files for the GUI components need to have the extension .mm (Objective C++). Since the cisst library is written in C++, we must use Objective C++ to communicate between the GUI and the library.