```swift
//
//  AppDelegate.swift
//  RGB USB
//
//  Created by Erik Nordlund on 4/30/19.
//  Copyright © 2019 Erik Nordlund. All rights reserved.
//
//  Arm USB includes the following open-source components:
//      • peertalk-simple: https://github.com/kirankunigiri/peertalk-simple
//      • ORSSerialPort: https://github.com/armadsen/ORSSerialPort

import Cocoa

@NSApplicationMain
class AppDelegate: NSObject, NSApplicationDelegate {



    func applicationDidFinishLaunching(_ aNotification: Notification) {
        // Insert code here to initialize your application
    }

    func applicationWillTerminate(_ aNotification: Notification) {
        // Insert code here to tear down your application
    }

    // MARK: - Core Data stack

    lazy var persistentContainer: NSPersistentContainer = {
        /*
         The persistent container for the application. This implementation
         creates and returns a container, having loaded the store for the
         application to it. This property is optional since there are
          legitimate
         error conditions that could cause the creation of the store to fail.
        */
        let container = NSPersistentContainer(name: "RGB_USB")
        container.loadPersistentStores(completionHandler: { (storeDescription,
         error) in
            if let error = error {
                // Replace this implementation with code to handle the error
                 appropriately.
                // fatalError() causes the application to generate a crash log
                 and terminate. You should not use this function in a shipping
                 application, although it may be useful during development.

                /*
                 Typical reasons for an error here include:
                 * The parent directory does not exist, cannot be created, or
                  disallows writing.
```

```
             * The persistent store is not accessible, due to permissions
              or data protection when the device is locked.
             * The device is out of space.
             * The store could not be migrated to the current model
              version.
             Check the error message to determine what the actual problem
              was.
             */
            fatalError("Unresolved error \(error)")
        }
    })
    return container
}()

// MARK: - Core Data Saving and Undo support

@IBAction func saveAction(_ sender: AnyObject?) {
    // Performs the save action for the application, which is to send the
     save: message to the application's managed object context. Any
     encountered errors are presented to the user.
    let context = persistentContainer.viewContext

    if !context.commitEditing() {
        NSLog("\(NSStringFromClass(type(of: self))) unable to commit
         editing before saving")
    }
    if context.hasChanges {
        do {
            try context.save()
        } catch {
            // Customize this code block to include application-specific
             recovery steps.
            let nserror = error as NSError
            NSApplication.shared.presentError(nserror)
        }
    }
}

func windowWillReturnUndoManager(window: NSWindow) -> UndoManager? {
    // Returns the NSUndoManager for the application. In this case, the
     manager returned is that of the managed object context for the
     application.
    return persistentContainer.viewContext.undoManager
}

func applicationShouldTerminate(_ sender: NSApplication) ->
 NSApplication.TerminateReply {
    // Save changes in the application's managed object context before the
     application terminates.
    let context = persistentContainer.viewContext
```

```
        if !context.commitEditing() {
            NSLog("\(NSStringFromClass(type(of: self))) unable to commit
             editing to terminate")
            return .terminateCancel
        }

        if !context.hasChanges {
            return .terminateNow
        }

        do {
            try context.save()
        } catch {
            let nserror = error as NSError

            // Customize this code block to include application-specific
             recovery steps.
            let result = sender.presentError(nserror)
            if (result) {
                return .terminateCancel
            }

            let question = NSLocalizedString("Could not save changes while
             quitting. Quit anyway?", comment: "Quit without saves error
             question message")
            let info = NSLocalizedString("Quitting now will lose any changes
             you have made since the last successful save", comment: "Quit
             without saves error question info");
            let quitButton = NSLocalizedString("Quit anyway", comment: "Quit
             anyway button title")
            let cancelButton = NSLocalizedString("Cancel", comment: "Cancel
             button title")
            let alert = NSAlert()
            alert.messageText = question
            alert.informativeText = info
            alert.addButton(withTitle: quitButton)
            alert.addButton(withTitle: cancelButton)

            let answer = alert.runModal()
            if answer == .alertSecondButtonReturn {
                return .terminateCancel
            }
        }
        // If we got here, it is time to quit.
        return .terminateNow
    }

}
```