appliedinformatics

Wissenswertes zum Einsatz von Open Source Software in Embedded Projekten

Günter Obiltschnig
Applied Informatics Software Engineering GmbH
Maria Elend 96/4
9182 Maria Elend
Austria
guenter.obiltschnig@appinf.com

1 Einleitung

Wie in vielen anderen Bereichen der Informationstechnologie findet Open Source Software auch in Embedded Projekten zunehmende Verwendung. Bekanntestes Beispiel ist der Linux-Kernel, der mittlerweile die Basis so mancher Embedded Systeme bildet. Nicht wegzudenken ist Open Source Software aus modernen Smartphones. So ist z. B. der größte Teil der Android Software Open Source, aber auch im iPhone findet sich eine große Menge an Open Source Software. Dennoch wirft der Einsatz von Open Source Software in Embedded Projekten einige spezielle Fragen auf. Diese werden nachfolgend diskutiert.

2 Open Source Software, Freie Software und Lizenzen

Als Open Source Software bezeichnet man Software, die unter einer von der Open Source Initiative[1] (OSI) anerkannten Open Source Lizenz veröffentlicht wurde. Die Open Source Initiative ist eine Non-Profit-Organisation (nach kalifornischem Recht), welche die Förderung von Open Source Software zum Ziel hat und 1998 gegründet wurde. Derzeit sind fast 70 verschiedene Lizenzen von der OSI als Open Source Lizenzen anerkannt. Die OSI hat eine Liste von zehn Kriterien[2] – die sogenannte Open Source Definition – veröffentlicht, welche eine Software-Lizenz erfüllen muss, um als Open Source Lizenz anerkannt zu werden. Die wichtigsten Kriterien davon sind:

- Freie Verteilbarkeit: die Lizenz darf die freie Verteilbarkeit der Software (auch als Teil eines größeren, kommerziellen Software-Paketes) nicht einschränken.
- Source Code: Der Source Code der Software muss verfügbar sein. Wenn die Software in kompilierter Form ausgeliefert wird, muss der Source Code entweder mitgeliefert werden, oder es muss einen einfachen Weg geben, den Source Code (z. B. durch Download von einem Web Server) zu erhalten.
- Die Lizenz muss Änderungen am Source Code sowie abgeleitete Werke erlauben und diese müssen unter der selben Lizenz verteilt werden dürfen.
- Die Verwendung der Software darf nicht eingeschränkt sein (z. B. auf bestimmte Personengruppen oder bezüglich bestimmter Einsatzgebiete).
- Die Lizenz kann die Weitergabe von modifiziertem Source Code verbieten; jedoch muss es in diesem Fall erlaubt sein, sogenannte Patch-Dateien, welche die Änderungen in maschinenlesbarer Form beschreiben, mit dem Source Code zu

[1] http://www.opensource.org/
[2] http://www.opensource.org/docs/osd

verteilen, und es muss erlaubt sein, Programme die mit modifizierten Varianten des originalen Source Codes gebaut wurden, zu verteilen.

Neben dem Begriff Open Source wird auch häufig der Begriff „Freie Software“ verwendet. Während beide Begriffe ähnliche Bedeutung haben, gibt es jedoch in den Details bestimmte Unterschiede. Der Begriff „Freie Software“ wurde von der Free Software Foundation (FSF) (bzw. deren Gründer Richard Stallman) geprägt, und bezeichnet Software, deren Lizenz folgende vier Freiheiten einräumt:

- Das Programm zu jedem Zweck auszuführen.
- Das Programm zu studieren und zu verändern.
- Das Programm zu verbreiten.
- Das Programm zu verbessern und diese Verbesserungen zu verbreiten, um damit einen Nutzen für die Gemeinschaft zu erzeugen.

Die bekannteste „freie“ Open Source Lizenz ist die GNU General Public License (GPL). Sie wurde von der Free Software Foundation erstmals 1989 veröffentlicht und später überarbeitet (Version 2 im Jahr 1991 und Version 3 im Jahr 2007). Sie gewährt explizit die vier angesprochenen Freiheiten und fordert außerdem, dass veränderte GPL Software, bzw. davon abgeleitete Werke ebenfalls unter der GPL weitergegeben werden müssen. Bei der Verwendung von GPL-lizenzierten Programmbibliotheken führt dies zum bekannten „viralen“ Effekt – sobald eine Applikation eine unter der GPL lizenzierte Programmbibliothek nutzt, muss der gesamte Source Code der Applikation ebenfalls unter der GPL verfügbar sein. Die bekannteste Software, welche die GPL Lizenz benutzt, ist der Linux Kernel. Hierbei gilt allerdings die Regelung, dass Applikationen, die lediglich die definierten Programmierschnittstellen (System Calls) des Kernels nutzen, keine abgeleiteten Werke des Kernels sind, und somit nicht unter die GPL fallen. Linux Applikationen greifen üblicherweise nicht direkt auf den Kernel zu, sondern immer über eine dazwischenliegende Bibliothek – die C Library in Form der GNU C Library, oder, speziell bei Embedded Systemen, der uClibc. Diese Bibliotheken werden unter der GNU Lesser General Public License (LGPL) veröffentlicht, einer etwas „abgeschwächten“ Variante der GPL, welche nicht den „viralen“ Charakter der GPL hat. Etwas komplexer ist die Situation jedoch bei Kernel Modulen, wie sie z. B. für Gerätetreiber verwendet werden. Hier gilt die Regelung, dass ein Treiber, welcher nur die definierten Programmierschnittstellen des Kernels nutzt, nicht unter der GPL veröffentlicht werden muss. Werden allerdings Veränderungen oder Erweiterungen am Kernel selbst vorgenommen so fallen diese wiederum unter die GPL und müssen veröffentlicht werden.

Ein weiterer Kernel, der die GPL nutzt, ist FreeRTOS[3]. Hier wird aber in den Lizenzbedingungen explizit festgelegt, dass die GPL nicht auf jene Applikationen, die den Kernel nutzen, übertragen werden muss. Somit müssen auch hier nur eigene Veränderungen, bzw. Erweiterungen am Kernel unter der GPL veröffentlicht werden.

Die Forderungen der GPL an abgeleitete Werke stellen einen sogenannten „starken“ Schutz der Freiheiten dar. Es wird sichergestellt, dass Weiterentwicklungen von einmal veröffentlichter GPL Software für immer frei bleiben.

Viele Programmbibliotheken werden unter der GNU Lesser General Public License (LGPL) veröffentlicht. Diese wurde speziell für die Verwendung mit wieder verwendbaren Programmbibliotheken konzipiert. Quellcode, welcher eine unter der LGPL lizenzierte Bibliothek lediglich nutzt muss dabei nicht veröffentlicht werden. Allerdings müssen Änderungen und Erweiterungen, die an der LGPL-lizenzierten Bibliothek selbst gemacht wurden, wiederum unter der LGPL veröffentlicht werden. Diese Regelung macht LGPL-lizenzierte Bibliotheken auch geeignet für die Verwendung in kommerziellen Produkten, wie auch in Embedded Systemen.

[3] http://www.freertos.org/

3 Pflichten beim Einsatz von Open Source Software

Viele Open Source Lizenzen erfordern es, den jeweiligen Lizenztext zusammen mit der Software, bzw. bei Embedded Systemen mit dem Gerät auszuliefern. Eventuell müssen auch die verwendeten Softwarekomponenten namentlich erwähnt werden. Dies kann z. B. als Teil der Dokumentation erfolgen, oder in Form eines Dokuments, welches von der Website des Unternehmens, oder auch z. B. vom integrierten Web Server des Gerätes, sollte es über einen verfügen, heruntergeladen werden kann. Bei GPL and LGPL Software empfiehlt es sich außerdem, alle verwendeten Komponenten im Source Code als Download bereitzustellen, auch wenn diese nicht modifiziert wurden. Bei einem Embedded Linux System sollten also der verwendete Kernel, das Board Support Package, die C Bibliothek, sowie eventuell verwendete weitere Tools (Busybox, etc.) als Source Code Download bereitgestellt werden. Damit können mögliche rechtliche Probleme im Zusammenhang mit Verletzungen der GPL von vornherein vermieden werden. Es gab in der Vergangenheit schon Fälle, wo bekannte Unternehmen gerichtlich zur Veröffentlichung des Source Codes gezwungen wurden, weil den Bedingungen der GPL nicht ausreichend Folge geleistet wurden. In diesem Zusammenhang ist zu erwähnen, dass im Falle von lizenzrechtlichen Unklarheiten in jedem Fall die Hilfe eines entsprechend kompetenten Rechtsbeistandes beigezogen werden soll.

Neben den relativ „strengen“ GPL und LGPL Lizenzen sind auch weitere, weniger restriktive Lizenzen weit verbreitet. Dazu zählen neben den bekannten Apache, MIT und BSD Lizenzen auch die Boost Software License, welche gerne bei C++ Bibliotheken benutzt wird. Diese Lizenzen werden auch oft als „liberale“ Open Source Lizenzen bezeichnet. Zu beachten ist noch, dass unter gewissen Open Source Lizenzen stehender Code nicht in unter GPL stehenden Code integriert werden darf. Darunter fällt z. B. Code, welcher unter der Mozilla Public License veröffentlicht wurde, oder z. B. auch die OpenSSL Bibliothek.

Nachfolgende Tabelle gibt eine Übersicht über die Eigenschaften verschiedener Open Source, bzw. Freier Software Lizenzen.

	Starker Schutz der Freiheiten	Schwacher Schutz der Freiheiten	Liberale Open Source Lizenzen	
	GPL	LGPL	Apache	MIT, BSD, Boost
Freier Zugang zum Quellcode	ja	ja	ja	ja
Quellcode darf verändert und mit anderer Software kombiniert werden	ja	ja	ja	ja
Quellcode darf mit proprietärer Software verteilt werden	nein	ja	ja	ja
Veränderungen dürfen verschlossen bleiben	nein	nein	ja	ja
Einzige Pflicht ist das Einfügen eines vorgegebenen Copyright-Vermerks und einer Haftungs-ausschlussklausel im Quellcode	nein	nein	nein	ja

Tabelle 1: Vergleich verschiedener Open Source Lizenzen[4]

4 Auswahlkriterien für Open Source Software

In Anbetracht von hunderttausenden verfügbaren Open Source Projekten kann es recht aufwändig sein die für ein Projekt nützlichen Open Source Pakete auszuwählen und zu bewerten. Es gibt kaum einen Bereich der Software, für den es noch kein entsprechendes Open Source Projekt gibt; oft hat man die Auswahl unter mehreren Alternativen. Nach welchen Kriterien kann nun Open Source Software bewertet, bzw. ausgewählt werden. Eine der wichtigsten Kriterien, speziell bei Embedded Software, ist zunächst die Lizenz. Programmbibliotheken, welche ausschließlich unter der GPL lizenziert sind, eignen sich kaum für die Integration in ein Embedded Projekt, es sei denn, man möchte seine gesamte Firmware für das Gerät ebenfalls unter der GPL veröffentlichen. Programmbibliotheken, die unter der LGPL, oder unter liberaleren Lizenzen wie MIT License, BSD License oder Boost License veröffentlicht werden, können bedenkenlos verwendet werden, sofern man seine Pflichten in Bezug auf die Veröffentlichung von Copyrightvermerken und Haftungsausschlussklauseln erfüllt. Ein noch wichtigeres Kriterium ist natürlich die Qualität der Software selbst. Diese objektiv zu bewerten kann schwierig sein, verbietet doch alleine schon der Umfang der meisten Softwarepakete oft eine detaillierte Analyse des Quellcodes. Viele Open Source Projekte veröffentlichen auf ihrer Website eine Liste von bekannten Benutzern. Auch lässt der Inhalt von Diskussionen aus den entsprechenden Supportforen, bzw. Mailinglisten eines Open Source Projektes, sowie das Bug Tracking System (wie viele bekannte Fehler gibt es, wie schnell werden diese behoben, etc.) einen guten Rückschluss auf die Softwarequalität zu. Nicht jedes Open Source Projekt ist auch für die Verwendung in einem Embedded Umfeld geeignet. Oft fehlt z. B. eine einfache Möglichkeit, die Software mit einem Cross Compiler für das Zielsystem zu bauen. Auch aus anderen Gründen kann das Projekt für ein Embedded System unbrauchbar sein, z. B. durch Codegröße oder unpassender Nutzung von dynamischer Speicherverwaltung. Ein weiteres Auswahlkriterium ist die hinter dem Open Source Projekt stehende Community. Handelt es sich z. B. um eine einzelne Person, welche die Software als Hobbyprojekt entwickelt, oder steht ein bekanntes Unternehmen (oder sogar mehrere) hinter dem Projekt. In beiden Fällen ist der Einsatz der

[4] Quelle: Open Source Software im geschäftskritischen Einsatz, Ernst & Young, 2011

Software möglich, jedoch müssen bestimmte Risiken eingeplant werden. Steht z. B. nur ein einzelner Entwickler hinter einem Projekt muss man die Möglichkeit einplanen, eventuell selbst die Software weiterpflegen zu müssen, sollte der Entwickler dieser Aufgabe nicht mehr nachkommen können oder wollen. In diesem Fall wird es notwendig sein, intern entsprechendes Know How aufzubauen. Auch sollte in diesem Fall auf die Qualität des Source Codes hinsichtlich der Wartbarkeit vermehrte Aufmerksamkeit gelenkt werden. Andererseits kann auch bei einem Open Source Projekt, das von einem großen Unternehmen betrieben wird, die Gefahr bestehen, dass das Projekt kurzfristig eingestellt wird, und man auf sich alleine gestellt wird. Bei einer entsprechende großen Anzahl von Benutzern bildet sich in so einem Fall aber üblicherweise schnell eine neue Community, welche das Projekt eigenständig weiterführt.

5 Arbeiten mit Open Source Communities

Eine wichtige Frage im Zusammenhang mit dem Einsatz von Open Source Software ist jene des Supports. An wen kann man sich wenden, wenn etwas nicht funktioniert, bzw. wenn sich Fragen zur Benutzung der Software stellen. Die meisten Open Source Projekte bieten zu diesem Zweck Mailing Listen, bzw. Diskussionsforen an, auf welchen man entsprechende Fragen stellen kann. Zu beachten ist jedoch, dass in diesem Fall der Support auf freiwilliger Basis geleistet wird. Darauf sollte man auch entsprechend Rücksicht nehmen und bevor man eine Frage stellt, zunächst einmal die Suchmöglichkeiten der Mailing Liste, bzw. des Diskussionsforums bemühen. Auch in den sogenannten Frequently Asked Questions findet sich häufig schon die Antwort. Es gibt auch Unternehmen, welche kommerziellen Support zu Open Source Projekten anbieten. Man erhält dadurch z. B. einen fixen Ansprechpartner und garantierte Antwortzeiten, es entstehen aber natürlich auch entsprechende Kosten, die sich aber üblicherweise lohnen.

Findet man einen Fehler in einem Open Source Projekt, so gibt es üblicherweise ein Bug Tracking System, in welches der Fehler eingetragen werden kann. Hier sollte man sich die Mühe machen, den Fehler möglichst genau zu beschreiben (eventuell mit einem Codebeispiel zur Reproduktion des Fehlers), und einen entsprechenden Eintrag im Bug Tracking System vorzunehmen. Gerne gesehen wird es auch, wenn man auch gleich einen entsprechenden Patch zur Behebung des Fehlers, oder zur Implementierung eines neuen Features bereitstellt. Dies erhöht die Wahrscheinlichkeit, dass die entsprechende Änderung schnell in die Code Basis des Open Source Projektes übernommen wird, speziell wenn der Fehler, oder das fehlende Feature nur eine kleine Anzahl von Nutzern betrifft.

Es kann sich auch lohnen, selbst an der Entwicklung eines Open Source Projektes mitzuarbeiten. Speziell wenn das Projekt einen wichtigen Stellenwert im eigenen Entwicklungsprojekt einnimmt, sichert man sich dadurch entsprechendes Know How, und auch eine gute Unterstützung durch die restliche Entwicklercommunity.

6 Finanzierung von Open Source Projekten

Die Entwicklung von Software kostet bekannterweise viel Geld, und dementsprechend stellt sich die Frage, wie die Entwicklung von Open Source Projekten finanziert wird. Tatsächlich ist es so, dass die wenigsten Open Source Projekte von Hobbyentwicklern betrieben werden. Bei den bekanntesten Open Source Projekten (z. B. Linux Kernel, Eclipse, etc.) wird ein großer Teil der Entwicklungskosten von Unternehmen getragen, die Mitarbeiter zur Entwicklung von Open Source Projekten abstellen, weil sie diese Software in großem Umfang selbst einsetzen. Oft wird die Entwicklung auch durch Unternehmen betrieben, welche die Entwicklung über Einnahmen aus Supportverträgen (oder sonstigen Dienstleistungen im Zusammenhang mit dem Open Source Projekt) finanzieren. Bei einigen Open Source Projekten gibt es auch Sponsoren, welche die Entwicklung finanziell, oder auch z. B. durch die Bereitstellung von Infrastruktur (z. B. Testsysteme oder Server) unterstützen. In jedem Fall sollte man sich beim Einsatz von Open Source Software überlegen, etwas an die Community zurückzugeben. Sei es durch aktive Mitarbeit an der

Entwicklung, durch Mitarbeit in Diskussionsforen, oder durch Sponsoring. In diesem Zusammenhang muss auch die verbreitete Meinung revidiert werden, dass der Einsatz von Open Source Software keine Kosten verursacht. Es fallen zwar Lizenzkosten und Royalties weg, dennoch können aber Kosten, z. B. durch Supportverträge oder „versteckte“ Kosten durch internen Aufwand für den Aufbau entsprechden Know Hows entstehen.

7 Open Source und sicherheitskritische Systeme

Der Einsatz von Open Source Software in sicherheitskritischen Systemen wirft spezielle Fragen auf, die eine genaue Betrachtung erfordern. Sicherheitskritische Software muss üblicherweise zertifiziert werden, und dies kann im Falle von Open Source Projekten problematisch sein, da der übliche Entwicklungsprozess von Open Source Projekten nicht auf die Entwicklung sicherheitskritischer Software abgestimmt ist. Bei kleineren Open Source Projekten kann es sinnvoll sein, den Open Source Code vollständig zu reviewen, dies entsprechend zu dokumentieren, und das Gesamtsystem zu zertifizieren. Übersteigt der Open Source Code allerdings einen bestimmten Umfang, so wird das nicht mehr zweckmäßig sein. In diesem Fall kann der verwendete Open Source Code unter Umständen mit dem Argument der Betriebsbewährtheit zertifiziert werden. In seltenen Fällen gibt es auch noch Unternehmen, die bereits zertifizierte oder zertifizierungstaugliche Versionen von Open Source Software anbieten. Ein Beispiel dafür ist z. B. SafeRTOS, eine nach IEC 61508 zertifizierte Variante von FreeRTOS, welche kommerziell vermarktet wird.