Open Transfer

Strategien und Werkzeuge
für den Transfer von
Forschungsergebnissen
im Open-Science-Kontext

www.OpenTransfer.eu

GUIDELINE

OPEN SOURCE HARDWARE

Dr. Ivonne Bieber, Dr. Tobias Engert, Dr. Kathrin Göbel, Peter Häfner,
Dr. Thomas M. Jahn, Dr. Vladimir Voroshnin, Dr. Björn Wolf

GSI

Leibniz ipht
LEIBNIZ-INSTITUT für PHOTONISCHE TECHNOLOGIEN

HZDR
HELMHOLTZ ZENTRUM DRESDEN ROSSENDORF

INNOcentric
turning ideas into value

Gefördert durch:
Bundesministerium für Forschung, Technologie und Raumfahrt

INHALT

1 Einleitung

Open Source Hardware (OSH) bezeichnet greifbare Produkte, deren Konstruktions- und Fertigungsdaten – z. B. Schaltpläne, CAD-Modelle, Stücklisten, Montage- und Prüfanleitungen – öffentlich und frei zugänglich sind. Jede:r darf diese Daten studieren, verändern, weiterverbreiten und daraus eigene Geräte herstellen oder verkaufen.

Das Arbeiten mit OSH folgt den Prinzipien Offenheit, Transparenz und Nachvollziehbarkeit. Durch die vollständige Offenlegung von Design-, Produktions- und Testinformationen wird nicht nur die Reproduzierbarkeit wissenschaftlicher Ergebnisse gesichert, sondern auch ein nachhaltiger Innovationszyklus geschaffen: Dritte können Designs übernehmen, verbessern und weiterentwickeln. Dieser gemeinschaftsorientierte Ansatz ist ein zentraler Motor für neue technische Lösungen.

Der Erfolg einer OSH-Entwicklung hängt jedoch nicht ausschließlich vom rein technischen Design ab. Gleichwertig wichtig sind Fragen der Dokumentationsqualität, Lizenzwahl, Finanzierung und Community-Betreuung. Fehlentscheidungen in diesen Bereichen treten häufig erst spät im Projekt auf und können die Weiterverwendung oder Kommerzialisierung gefährden.

Ziel dieses Leitfadens ist es, Forschenden, Entwickler:innen, Transfer- und Technologiemanagement-Einheiten sowie Industriepartner:innen eine praxisorientierte Unterstützung zu bieten. Er behandelt rechtliche, technische und geschäftliche Aspekte systematisch und liefert Werkzeuge, um nachhaltige, breit nutzbare OSH-Projekte aufzusetzen und zu betreiben.
Der beschriebene Ablauf ist bewusst iterativ gestaltet: einzelne Schritte können parallel laufen, wiederholt oder angepasst werden, sobald neue Erkenntnisse oder veränderte Rahmenbedingungen auftreten.

Der Leitfaden richtet sich an alle Akteur:innen, die Open-Source-Hardware im Rahmen von Open-Science-Initiativen entwickeln, veröffentlichen und ggf. kommerziell nutzen wollen – unabhängig davon, ob sie an Forschungseinrichtungen, Hochschulen, Unternehmen, Start-Ups oder anderen Organisationen tätig sind.

1.1 Der Weg zur Verwertung als Open-Source-Hardware

Bedarf und Motivation

Der Bedarf an neuer Hardware entsteht in Laboren, Produktionsstätten, Lehrumgebungen oder im Feld: Es fehlt ein Messgerät, ein Adapter, eine Steuerungseinheit oder ein komplett neu zu entwickelndes System. Gleichzeitig besteht oft der Wunsch, durch die Entwicklung einen Mehrwert zu schaffen und die Ergebnisse offen zu teilen – sei es zur Beschleunigung der Wissenschaft, zur Stärkung der Lehre oder zur Erschließung neuer Märkte.

Projektstart und Anforderungsanalyse

Jedes Projekt beginnt mit einer klaren Idee und einem definierten Ziel: Die zu entwickelnde Hardware soll ein konkretes Problem lösen. Die Entwickler:innen formulieren dabei

- funktionale und leistungsbezogene Anforderungen (z. B. Messgenauigkeit, Datenrate, Größe, Energiebedarf),
- Anwendungsparameter (Labor-, Feld- oder Produktionsumgebung, vorhandene Infrastruktur) und
- den erwarteten Mehrwert (z. B. präzisere Messungen, höhere Durchsatzrate, geringere Kosten).

Eine realistische Risiko- und Machbarkeitsabschätzung ist dabei unverzichtbar.

- **Technische Machbarkeit** – Verfügbarkeit der Bauteile, Kompatibilität mit vorhandener Hardware, notwendige Fertigungsverfahren.
- IP-Situation – Patente, Marken oder Lizenzbedingungen für einzelne Komponenten oder Software; dürfen sie im Open-Source-Kontext verwendet werden?
- Finanzielle Risiken – Beschaffungs- und Herstellungskosten im Vergleich zum Budget, Gefahr von Lieferengpässen oder Preisschwankungen.
- Zeitliche Risiken – Aufwand für Design, Prototyping, Test und Dokumentation; kritische Meilensteine.
- Regulatorische Risiken – Exportkontrollen, REACH-/RoHS-Vorschriften, CE-Kennzeichnung, Produkthaftungsaspekte.

Kooperation und Weg zur Verwertung

Der Weg zur Verwertung einer Open Source Hardware sollte frühzeitig gemeinsam mit Transfermanager:innen und/oder einem Open Source Program Office (OSPO) beschritten werden. Diese unterstützen bei der Klärung offener Fragen zu Lizenzen, Schutzrechten und

Geschäftsmodellen. Ein strukturierter Prozess von Anfang an vermeidet spätere Konflikte und sichert die Nachhaltigkeit des Projekts.

Potenzialanalyse

Um zu beurteilen, wie weit der Entwicklungsstand eines Hardwarevorhabens ist sowie ob und wie gut es für Open Source geeignet ist bzw. an welcher Stelle noch Handlungsbedarf besteht, wurde ein digitales Tool zur Bewertung des Potenzials erarbeitet.[1]

Dafür werden Dimensionen wie Charakteristika der Hardware selbst, das Entwicklungsteam und Nebenbedingungen der Forschungseinrichtung, aber auch der potenzielle Markt und das Potenzial für eine Ausgründung berücksichtigt. Die zugehörigen Fragen sind auf die Spezifika der das Tool nutzenden Einrichtung anpassbar.

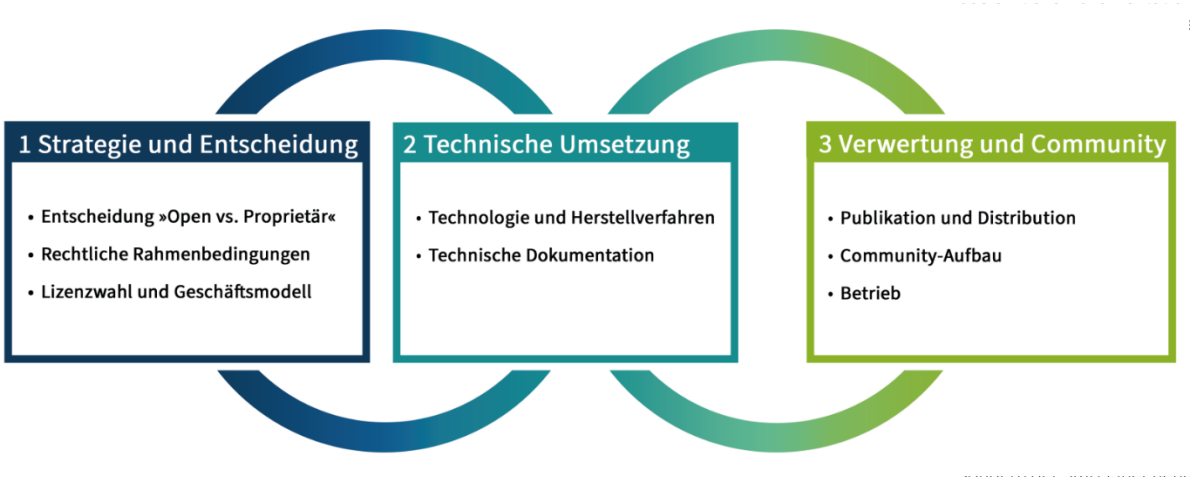
Das Tool kann unter diesem Link heruntergeladen werden:
https://opentransfer.eu/d/OSHPotenzial

Für die Nutzung ist Microsoft Excel notwendig. Die Fragen können in einem gewissen Umfang noch an die Spezifik eines Projektes oder die besonderen Erfordernisse einer Organisation angepasst werden.

[1] Aufbauend auf dem Bewertungsschema und dem zugehörigen Excel-Tool »Potenzialanalyse – Bewertung des Verwertungspotenzials von Forschungssoftware« des SoftWert-Konsortiums. → www.SoftWert.org

1.2 Der Lebenszyklus eines Open-Source-Hardware-Projekts

Die Entwicklung von OSH ist kein lineares Vorgehen, sondern ein **iterativer, feedbackgesteuerter Prozess**, bei dem technische, rechtliche und geschäftliche Aspekte parallel bearbeitet werden. Die nachstehenden Phasen bilden einen orientierenden Rahmen; Übergänge sind fließend und Rückkopplungen jederzeit möglich.



Phase 1: Strategie und Entscheidung

Bevor mit der technischen Entwicklung begonnen wird, müssen die grundlegenden Weichen gestellt werden.

- **Entscheidung „Open oder proprietär“** – Klärung, ob das Projekt als Open-Source-Hardware veröffentlicht werden soll oder ein proprietäres Geschäftsmodell verfolgt wird.
- **Rechtliche Rahmenbedingungen** – Prüfung von Exportkontrollen, bestehenden Schutzrechten und Haftungsfragen; Einbeziehung aller Rechteinhaber (Mitentwickler:innen).
- **Lizenzwahl und Geschäftsmodell** – Gemeinsame Festlegung einer passenden OSH-Lizenz (z. B. Copyleft vs. permissiv) und des damit verbundenen Geschäftsmodells, da die Lizenz die zulässigen kommerziellen Nutzungsmöglichkeiten maßgeblich bestimmt.

Phase 2: Technische Umsetzung

In dieser Phase findet die eigentliche Entwicklung statt, wobei die Dokumentation von Anfang an mitgedacht wird.

- **Technologie und Herstellverfahren** – Auswahl der eingesetzten Hardware- und Software-Technologien sowie der geeigneten Produktionsmethoden (z. B. 3-D-Druck, CNC-Fräsen, Serienfertigung).

- Technische Dokumentation – Vollständige Erfassung aller Konstruktions- und Fertigungsdaten (Schaltpläne, CAD-Dateien, Stücklisten, Montage- und Prüfanleitungen) in einer Form, die Dritte einen sicheren Nachbau und Betrieb ermöglicht.

Phase 3: Publikation, Distribution und Community

Die Veröffentlichung ist kein Endpunkt, sondern der Start einer neuen Phase.

- Publikation und Distribution – Festlegung, auf welchen Plattformen (z. B. GitHub, OSHWA-Registry, Zenodo) die Daten bereitgestellt und wie sie dauerhaft zugänglich gemacht werden (z. B. DOI).
- Community-Aufbau – Definition von Mechanismen zur Einholung von Feedback, zum Onboarding neuer Mitentwickler:innen und zur Moderation der Community.
- Betrieb – Umsetzung des gewählten Geschäftsmodells und Sicherstellung einer langfristigen Wartung und Weiterentwicklung des Projekts.

Praktischer Hinweis für den Projektstart

- Nicht alle Themen müssen zu Beginn vollständig geklärt sein – aber insbesondere nicht-technische Fragen (Lizenz, Finanzierung, Community-Management) werden in der Praxis häufig zu spät behandelt.
- Damit das Entwicklungsteam die Zügel selbst in die Hand nimmt, empfiehlt es sich, bereits zu Projektbeginn die einzelnen Themen kurz zu durchdenken und eine projektbezogene Zeitplanung zu erstellen, in der festgelegt wird, wann welche Entscheidung getroffen werden kann.
- Je nach Organisation gibt es unterschiedliche Unterstützungsangebote:
  - ein bereits bestehendes Open-Source-Program-Office (OSPO),
  - die für Wissens- und Technologietransfer zuständige Struktureinheit,
  - erfahrene Kolleg:innen aus anderen Fachbereichen, die bereits mit Open-Source-Projekten gearbeitet haben.

2 Entscheidung: Open oder Proprietär

Die grundsätzliche Frage, ob ein Produkt als Open-Source-Hardware (OSH) oder als geschlossenes, proprietäres System entwickelt werden soll, muss zu Beginn eines Projekts geklärt werden. In der Praxis kommt es jedoch häufig vor, dass eine bereits zu Projektstart getroffene Entscheidung später mit den Anforderungen an die Verwertung (z. B. Zertifizierung, Marktstrategie) kollidiert. Deshalb ist ein bewusstes, mehr-perspektivisches und möglichst rationales Vorgehen zu empfehlen – nicht das bloße „Alle machen das so"-Prinzip.

Vorteile von Open-Source-Hardware

Open-Source-Hardware bringt mehrere, gut belegte Pluspunkte mit, die gerade in frühen Projektphasen entscheidend sein können.

- Transparenz – OSH-Designs sind keine Black-Box; sie können vollständig eingesehen, nachvollzogen und geprüft werden.
- Flexibilität – Der Quell- und Schaltplan lässt sich jederzeit an neue Anforderungen oder an individuelle Bedürfnisse anpassen.
- Kollektive Ressourcen – Idealerweise tragen zahlreiche Entwickler aus einer Community ihr Fachwissen und zusätzliche Entwicklungszeit bei, wodurch das Projekt schneller vorankommt.
- Niedrige Lizenzkosten – Kostenfreie OSH-Lizenzen bedeuten, dass nur die eigentlichen Herstellungs- und Betriebskosten für die Endnutzer anfallen.

Mögliche Nachteile (insbesondere für die spätere Nutzung)

Auch wenn Open-Source-Hardware viele Stärken hat, gibt es kritische Aspekte, die insbesondere die Markt- und Nutzerakzeptanz betreffen.

- Zertifizierung & Regulatorik – Bei rein kostenlosen OSH ist es selten, dass jemand die hohen Aufwendungen für Zulassungen (z. B. medizinische Geräte) übernimmt. Das kann die praktische Einsatzfähigkeit stark einschränken oder sogar unmöglich machen.
- Produktionskosten – OSH-Projekte werden häufig für Einzel- oder Kleinserien (3-D-Druck, CNC) konzipiert, was pro Stück höhere Kosten verursacht als eine industrielle Massenfertigung.
- Usability – Open-Source-Projekte verfügen oft nicht über dedizierte Ressourcen für benutzerfreundliche Bedienungsanleitungen, UI/UX-Design oder Support-Materialien, sodass die Anwenderfreundlichkeit darunter leiden kann.

Entscheidungs-Kriterien – Fokus auf das langfristige Projektziel

Um die passende Verwertungsstrategie zu finden, sollten Sie die langfristigen Projektziele, die angestrebten Nutzergruppen und den Finanzierungsbedarf systematisch prüfen.

1. Projektziel – Soll die Hardware möglichst breit genutzt werden (z. B. in Bildung, Forschung) oder steht die kommerzielle Erlösmaximierung im Vordergrund?
2. Zielgruppen – Welche Nutzer*innen oder Nutzergruppen werden die Hardware künftig einsetzen und zu welchem Zweck?
3. Finanzierungsbedarf – Muss das Projekt bereits jetzt oder zu einem späteren Zeitpunkt Einnahmen generieren, um weiter zu existieren? Wenn ja, wie hoch sind die benötigten Mittel und wofür genau?

Ausschlusskriterium: Exportkontrollen

Hardware, die militärisch einsetzbar oder anderweitig sanktionierten Bereichen zugeordnet ist, unterliegt häufig Export-Kontrollen (z. B. EU-Dual-Use-Verordnung VO 2021/821, Außenwirtschaftsgesetz). In solchen Fällen gilt die Veröffentlichung im Internet – etwa auf GitHub – bereits als Export und ist ohne spezielle Genehmigung rechtlich riskant oder sogar unzulässig. Deshalb sollte diese Prüfung vor jeder öffentlichen Freigabe erfolgen, idealerweise bereits im Rahmen der grundlegenden Entscheidung für oder gegen Open Source (siehe Abschnitt 2.1).

2.1 Das OpenTransfer-Entscheidungs-Tool

Wie sich die Zielüberlegungen auf die Entscheidung Open Source oder Proprietär auswirken, hat OpenTransfer in einem ausführlichen Tool *„Entscheidungshilfe Open vs. Proprietär“* aufbereitet. In vier Themenbereiche geordnet werden eine Reihe von Fragen gestellt und mögliche Antworten eingeordnet, ob sie mehr für oder gegen eine Open-Source-Veröffentlichung sprechen.

Im ersten Bereich Personen / Team werden Motivation und Engagement-Potential der Projektbeteiligten erfasst, etwa die Bereitschaft zu freiwilligem Aufwand, individuelles Erlösinteresse sowie das Vorhandensein oder der Aufbau einer Community.

Der zweite Bereich Organisation klärt die strategische Präferenz der Einrichtung – ob sie eher auf offene oder proprietäre Verwertungsmodelle ausgerichtet ist – und ob bereits Vor-Lizenzen bzw. Lizenz-Restriktionen die Wahl einschränken.

Der dritte Bereich Produkt beinhaltet Fragen zur technischen Reife, zu Sicherheits- und Zuverlässigkeitsanforderungen sowie zur Bedeutung einer Open-Source-Reputation für die Marktdurchdringung.

Im vierten Bereich Finanzierung wird der Finanzierungsbedarf, das erwartete Marktvolumen, die Realisierbarkeit von Open-Geschäftsmodellen (Service, Support, Dual Licensing) und die generelle Erfolgswahrscheinlichkeit einer wirtschaftlichen Verwertung beleuchtet.

Die Antworten können auch in einem elektronischen Fragebogen aggregiert und zu einem Polaritätsprofil zusammengefasst werden. Dieses bildet visuell die Tendenz zu einer offenen, gemischten oder proprietären Verwertungsstrategie ab. Darin können auch unterschiedliche Präferenzen einzelner Teammitglieder wie auch die, ggf. konträren, Auswirkungen einzelner Themenkomplexe dargestellt und dann diskutiert werden.

Das Ergebnis ist bewusst kein abschließendes Urteil, sondern ein Entscheidungsimpuls, der die nächsten Analyse- und Planungsschritte – etwa einen detaillierten Lizenz-Check, eine Marktstudie oder die Ausarbeitung eines Business-Cases – gezielt steuert. Das Tool lässt sich einfach in bestehende Projekt-Governance-Prozesse einbinden und unterstützt Teams dabei, bereits in der Anfangsphase die passende Verwertungsstrategie zu identifizieren.

Das Tool kann unter diesem Link heruntergeladen werden:
https://opentransfer.eu/d/OpenProp

2.2 Exportkontrolle und Ausfuhrregularien

Von den meisten Staaten – auch von Deutschland – wird der Export von Produkten, die militärisch, sicherheitstechnisch oder kerntechnisch eingesetzt werden können, streng reguliert. Das gilt ebenso für Open-Source-Software (OSS) und Open-Source-Hardware (OSH). Sobald ein Entwurf im Internet (z. B. auf GitHub) veröffentlicht wird, gilt das als Export, weil theoretisch jede Person weltweit darauf zugreifen kann.

Unterliegt die Technologie bzw. Hardware Ihres Projekts der Exportkontrolle oder Sanktionen, ist eine Open-Source-Veröffentlichung rechtlich verboten, bis Sie eine behördliche Genehmigung erhalten haben.

Die maßgeblichen Rechtsgrundlagen sind

- EU-Dual-Use-Verordnung (VO 2021/821)[2] – regelt die Ausfuhr von Gütern, die sowohl zivil- als auch militärisch nutzbar sind.
- Außenwirtschaftsgesetz (AWG) – nationale Ergänzung zur EU-Verordnung.
- EU-Sanktionen – Verbote für bestimmte Länder, Organisationen oder Personen.

In Deutschland wird die Durchsetzung durch das Bundesamt für Wirtschaft und Ausfuhrkontrolle (BAFA)[3] vorgenommen. Meist betreffen die Regelungen Exporte außerhalb der EU, aber das Internet lässt keine geografische Beschränkung zu, sodass jede Onlineveröffentlichung unter die Exportkontrolle fällt. Die meisten Forschungseinrichtungen haben interne Spezialisten, die bei der Prüfung der Frage helfen, ob und in welcher Form die Ausfuhr (und damit die Veröffentlichung) eingeschränkt ist.

Für ein Open Source Vorhaben sind grundsätzlich folgende Dinge zu prüfen:

- Sanktionen prüfen
  - Gibt es EU-Sanktionen, die das Ziel-Land, die Zielorganisation oder die Zielperson betreffen?
    - Sind bestimmte Technologiebereiche (z. B. Kryptografie, Raketentechnik) von den Sanktionen erfasst?
- Dual-Use-Liste prüfen
  - Finden Sie Ihre Hardware- oder Software-Komponente im Anhang I der EU-Dual-Use-Verordnung (VO 2021/821)?
  - Wenn ja, benötigen Sie eine Genehmigung, solange das Produkt nicht bereits Public Domain ist.
- Public-Domain-Ausnahme
  - Ist die betreffende Technologie bereits öffentlich zugänglich (z. B. in einer früheren Veröffentlichung, in einer Bibliothek oder in einem öffentlichen Standard)?
  - Ist das der Fall, ist für die erneute Veröffentlichung keine zusätzliche Genehmigung mehr erforderlich.

[2] Im Anhang IV der Dual-Use-Verordnung (VO (EU) 2021/821) gelistete Güter

[3] https://www.bafa.de Hier finden sich auch ausführliche Informationen zur Ausfuhrkontrolle.

- Endverwendungscheck (Catch-all-Prüfung)
    - Können Sie einen sensiblen Verwendungszweck erkennen oder plausibel annehmen? Dazu zählen:
        - Militärische oder rüstungsbezogene Nutzung in einem Embargoland
        - Einsatz in Massenvernichtungswaffen oder deren Trägersystemen
        - Verwendung zur internen Repression oder bei schweren Menschenrechtsverletzungen
    - Wenn einer dieser Punkte zutrifft, muss das BAFA einbezogen werden.
- Dokumentation
    - Es empfiehlt sich, die obigen Schritte zu dokumentieren für eine etwaige spätere Prüfung.

3 Rechtliche Aspekte

3.1 Schutzrechte

Geistiges Eigentum, worunter auch Software oder Designs und Pläne für Hardware fallen, sind in der EU und vielen anderen Ländern geschützt. Prinzipiell kann dieser Schutz auf vier Wegen erfolgen:

a) Urheberrecht
b) Patentrecht
c) Geheimhaltung
d) Einzelvertragliche Regeln

Die Punkte c) und über weite Strecken auch d) sollen hier außen vor bleiben, weil sie dem Grundgedanken von Open Source widersprechen und eher für die proprietäre, kommerzielle Verwertung interessant sind.

3.1.1 Urheberrecht

Das Urheberrecht ist die wichtigste Grundlage für den Schutz und die Lizenzierung von Open Source. Es gilt für sämtliche schöpferischen Ausdrucksformen. Für Hardware gehören dazu Schaltpläne, PCB-Layouts, CAD-Modelle, Stücklisten, Firmware-Quellcode sowie alle begleitenden Dokumente (Handbücher, Tutorials, Bilder, Videos). Der Schutz entsteht automatisch mit der Schöpfung des Werkes; eine gesonderte Registrierung ist nicht erforderlich. Es ist an sich nicht übertragbar und gilt bis 70 Jahre nach Tod des Urhebers.

Das Nutzungsrecht an den Werken kann hingegen frei übertragen werden. Allerdings beschränkt sich der Schutz auf die konkrete Ausgestaltung des Werkes: wenn eine Grundidee neu formuliert oder eine Software zwar in gleicher Art, aber neu programmiert wird (neuer Sourcecode), wirkt der Schutz nicht mehr. Ebenso umgeht die Umformulierung eines Textes, auch wenn die Aussage gleich bleibt, den Urheberrechtsschutz.

Dennoch beruhen alle Nutzungskonzepte und Lizenzen für Open Source Soft- oder Hardware auf dem urheberrechtlichen Schutz.

3.1.2 Patentrecht

Ein weiterer Weg OSH zu schützen, ist das Patent. Ein erteiltes Patent bietet rechtlich den stärksten Schutz gegen unerlaubte Verwendung – jedoch nur für 20 Jahre. Allerdings dürfen nur

Erfindungen patentiert werden, nicht alle schöpferischen Werke. Patentierbar sind nur technische Erfindungen, die neu, aus erfinderischer Tätigkeit resultieren, technischer Natur und gewerblich anwendbar sind. Das schließt viele Entwicklungen und die allermeiste Software von vornherein aus. Außerdem müssen Patente beantragt werden und sind mit Kosten verbunden.

Für OSH bieten Patente kaum Mehrwert, weil bei Open Source nicht die Nutzung verboten, sondern gerade erlaubt werden soll. Falls schon ein Patent besteht, kann dieses nach Veröffentlichung aufgegeben werden, weil den jährlichen Gebühren kein Nutzen z.B. durch Lizenzeinnahmen mehr gegenübersteht. Eine Veröffentlich als Open Source verhindert auch effektiv, dass ein Dritter missbräuchlich die eigene Erfindung zum Patent anmeldet (weil veröffentlicht und damit nicht mehr neu).

3.1.3 Marken- und Designschutz – Hinweise

Eine Marke (Wort-, Bild- oder Kombinationsmarke) schützt die Kennzeichnung von Produkten oder Dienstleistungen. Sie gibt dem Inhaber das ausschließliche Recht, die Marke im geschäftlichen Verkehr zu nutzen und Dritten die unberechtigte Nutzung zu untersagen.

Warum ist das für OSH wichtig?

- Produkt- und Projektidentität: Logos, Projektnamen oder Qualitätssiegel werden häufig als Marken eingetragen, um die Herkunft und den Ruf der Hardware zu sichern.
- Rechtssicherheit: Vor der Veröffentlichung muss geklärt sein, ob das Projekt eigene Marken nutzt (schützt die eigene Identität) oder Marken Dritter (z. B. von Kooperationspartnern, Lieferanten oder bekannten Open-Source-Projekten) enthält.
- Lizenz- und Nutzungserlaubnis: Ohne ausdrückliche Erlaubnis dürfen fremde Marken weder im Quell-/Design-Material noch in der Dokumentation verwendet werden – sonst besteht Gefahr von Unterlassungs- und Schadensersatzansprüchen.

Ein eingetragenes Design (in Deutschland: *Geschmacksmuster*, EU: *Community Design*) schützt die ästhetische Erscheinungsform eines Produkts – also die sichtbare Gestaltung von Linien, Konturen, Farben, Oberflächenstrukturen und Gesamtformen. Der Schutz greift unabhängig davon, ob das Design funktional ist, er bezieht sich ausschließlich auf die optische Wahrnehmung.

Warum ist das für OSH wichtig?

- Identität & Wiedererkennungswert: Ein unverwechselbares Gehäuse kann das Projekt stark branden; ein eingetragenes Design bietet hier rechtlichen Schutz.
- Kompatibilität mit Open-Source-Lizenzen: Viele Open-Hardware-Lizenzen (z. B. CERN-OHL) erlauben weder die Weitergabe noch die Modifikation eines geschützten Designs, solange das Design nicht explizit lizenziert wird.
- Risiko von Nachahmern: Ohne Design-Schutz können Dritte das äußere Erscheinungsbild kopieren, selbst wenn die zugrundeliegende Schaltung offen ist.

3.2 Nutzungsrecht

In Deutschland liegt das Nutzungsrecht grundsätzlich beim Urheber. Für im Rahmen eines Arbeitsverhältnisses erstellte Werke gilt in der Regel die Arbeit-für-den-Arbeitgeber-Regel: Das Nutzungsrecht geht auf die Einrichtung (Universität, Forschungseinrichtung, Unternehmen) über, sofern kein abweichender Vertrag besteht.

Klärung der Rechteinhaberschaft

Nur der Inhaber des Nutzungsrechts ist befugt, das Werk unter einer Open-Source-Hardware-Lizenz zu veröffentlichen. Deshalb müssen vor der Freigabe folgende Punkte geprüft werden:

1. Arbeits- und Kooperationsverträge : Liegt ein klassisches Anstellungsverhältnis vor, ist die Einrichtung (Arbeitgeber) der Rechteinhaber und darf die Lizenz erteilen.
2. Externe Beiträge (Gast-Wissenschaftler*innen, Industriepartner, Community-Contributor) benötigen eine ausdrückliche Rechte-Übertragung. Dafür sind etablierte Verfahren empfehlenswert:
    - Developer Certificate of Origin[4] (DCO) – Der Beitragende bestätigt per Commit-Message-Footer, dass er die erforderlichen Rechte besitzt und die Lizenzierung erlaubt.
    - Contributor License Agreement (CLA) – Ein kurzer Vertrag, in dem der Contributor dem Rechteinhaber (z. B. der Universität) das notwendige Verwertungsrecht überträgt oder eine Lizenzgewährung erklärt.

Durch die Kombination aus eindeutig dokumentierter Urheberschaft und einer geprüften, vertraglich gesicherten Verwertungsrechtslage (DCO/CLA bzw. vertragliche Regelung) ist

[4] https://developercertificate.org

sichergestellt, dass die anschließend angewandte Open-Source-Hardware-Lizenz rechtlich wirksam und problemlos durchsetzbar ist.

Dokumentation bei der Publikation

Damit die Zuordnung von Urheberschaft und Verwertungsrecht transparent ist, muss bei der Publikation die Autorenschaft dokumentiert sein. Die gängige Praxis ist das Anlegen eines Autorenregisters im Projekt-Root, etwa einer Datei AUTHORS.md. Darin werden für jede mitwirkende Person Name, E-Mail-Adresse und optional ORCID-ID oder sonstige eindeutige Kennung festgehalten. Dieses Register dient sowohl internen Verantwortlichkeiten (Nachweis, wer das Werk geschaffen hat) als auch externen Nutzer*innen, die die Namensnennungspflicht der Lizenz erfüllen müssen.

3.3 Produkthaftung

Wenn eine Hardware nicht richtig funktioniert, können echte Schäden entstehen - wirtschaftliche Schäden oder auch Personenschäden. Viele Open Source Projekte sind frei und kostenlos verfügbar. Da keine Einnahmen erzielt werden, wollen die Entwickler:innen natürlich auch nicht haften.

Deshalb findet sich in allen Open Source Lizenzen ein möglichst weitgreifender Ausschluss dieser Haftung. Die sogenannte Produkthaftung hingegen ist per Gesetz geregelt und kann nicht ausgeschlossen werden. Das wird durch das Produkthaftungsgesetz geregelt. Das stammt im Kern noch aus den 1980er Jahren und wurde angepasst: die EU hat im Oktober 2024 eine neue Produkthaftungsrichtlinie verabschiedet (RL 2024/2853), die bis Ende 2026 in deutsches Recht überführt wird. Die folgenden Ausführungen gründen auf der neuen Richtlinie.

Produkthaftung heißt, dass wenn durch den Fehler eines Produktes ein Schaden an einer Person oder Sache entsteht, der Hersteller haften muss. Dabei ist völlig egal, was für ein Fehler das ist (Fehler in der Konstruktion, der Fabrikation oder der Anleitung) oder ob der Hersteller etwas dafür kann (Vorsatz, einfache oder grobe Fahrlässigkeit). Er muss immer haften.

Neu ist nun, dass auch digitale Produkte wie z.B. eine Software als Produkte im Sinne des Gesetzes gelten, was bis dato nicht so war. Auch werden nun Schäden an immateriellen Dingen wie z.B. Daten mit erfasst.

Es gibt aber Grenzen. Der Hersteller haftet nicht für fehlerhafte Nutzung, bei offenkundigem Missbrauch, für Mängel, die erst beim Nutzer entstanden sind, für Mängel die er bei Herstellung des Produkts nach dem damaligen Stand von Wissenschaft und Technik gar nicht erkennen konnte und für Mängel, die auf zwingend einzuhaltenden rechtlichen Vorgaben beruhen. Allerdings kann der Hersteller durchaus haften, wenn die Fehler des Nutzers auf eine falsche oder mangelhafte Bedienungsanleitung zurückgeht oder wenn aus irgendwelchen Gründen der fehlerhafte Gebrauch vorhersehbar war.

Für nicht-kommerzielle Anbieter und Anbieter kostenloser Hard- und Software sind jedoch Ausnahmen vorgesehen.

Keine Haftung, wenn die Soft- oder Hardware kein Produkt ist

Open Source Soft- und Hardware lebt davon, dass der Nutzer bzw. Lizenznehmer die Soft- oder Hardware selber herstellt: wenn die Software selbst kompiliert wurde oder die Hardware selbst aus einem Hardwaredesign erstellt wurde, dann sind die Erschaffer nicht mehr in der Haftung. Allerdings ist die genaue Grenze zwischen Konzept und Produkt mangels Urteilen dazu nicht eindeutig geregelt.

Keine Haftung, wenn das Produkt nicht selbst in den Verkehr gebracht wurde

Wenn das Produkt durch einen Dritten in den Verkehr gebracht wird, dann haftet auch nur dieser. Wenn mehrere daran beteiligt sind, die z.B. jeweils Komponenten herstellen, dann haftet jeder für seinen Teil.

Keine Haftung, wenn das Produkt nicht im Rahmen einer Geschäftstätigkeit erstellt wurde

Besonders auf die Bedürfnisse von Open Source Projekten trifft die Ausnahme für fehlende Geschäftstätigkeit zu. Dabei ist allerdings zu beachten: fehlende Geschäftstätigkeit heißt nicht einfach nicht-kommerziell. Wenn man sich Unkosten ersetzen lässt oder regelmäßig mit eng verbundener Auftragsforschung Geld einnimmt oder im Dual Licensing einen Teil der Produkte gegen Geld abgibt – dann sind das alles Indizien für eine Geschäftstätigkeit – und damit für eine Haftung. Auch hier gilt (leider), dass die Grenzen vom Einzelfall abhängen und keine eindeutigen Aussagen zu treffen sind.

Diese drei Ausschlusskriterien gelten jedes für sich allein: man haftet nicht, wenn man

- ein „Nicht-Produkt“ in den Verkehr bringt

  oder
- ein Produkt nicht selbst in den Verkehr bringt

  oder
- ein Produkt außerhalb einer Geschäftstätigkeit in den Verkehr bringt

Ausführlichere Informationen finden sich in einer separaten Handreichung zum Thema Produkthaftung, in der auch für Forschungseinrichtungen typische Beispiele behandelt werden.

https://opentransfer.eu/d/Produkthaftung

4 Lizenzen und Kompatibilität

Das Urheberrecht schützt automatisch jede schöpferische Ausdrucksform (Schaltpläne, PCB-Layouts, CAD-Modelle, Firmware-Quellcode, Handbücher usw.). Ohne ausdrückliche Erlaubnis ist jede Nutzung außer den eng definierten Ausnahmefällen (wie z.B. Zitat oder Verwendung in der Lehre) verboten.
Eine Lizenz ist das juristische Werkzeug, mit dem der Urheber (bzw. der Inhaber des Verwertungsrechts) festlegt, wie und unter welchen Bedingungen Dritte das Werk nutzen, verändern und weiterverbreiten dürfen.

Ein Open-Source-Hardware-Projekt besteht typischerweise aus vier Bausteinen, die jeweils urheberrechtlich geschützt sind:

- Hardware-Design – Das technische Layout und die Fertigungsdaten (Schaltpläne, PCB-Layouts, CAD-Modelle, Stücklisten).
- Software / Firmware – Quellcode, Binärdateien, Treiber, Bibliotheken und sonstige für die Hardware notwendige Programme.
- Dokumentation – Bau- und Bedienungsanleitungen, Tutorials, Handbücher, Bilder, Videos, Diagramme und sonstige erläuternde Materialien.
- Marken – Logos, Produktnamen, Qualitätssiegel oder andere Kennzeichen, die unter dem Markenrecht geschützt sind.

Da diese drei Elemente unterschiedliche Anforderungen an Lizenzen haben empfiehlt es sich, für jedes Element eine eigene, jedoch auf die anderen beiden Elemente abgestimmte Lizenz zu wählen:

- Hardware: explizite Hardwarelizenz (CERN OHL 2.0 P/W/S; Solderpad, TAPR)
- Software: Eine von vielen Softwarelizenzen (GPL 3.0, LGPL, Apache, MIT, BSD, …)
- Dokumentation: Creative Commons Lizenz (CC0, CC-BY, CC-BY-SA)

4.1 Lizenzkategorien

Für Open Source Hardware – wie auch für Open Source Software – gibt es drei Kategorien von Lizenzen. Sie definieren, wie frei die Soft- oder Hardware verwendet, verändert und weitergegeben werden dar.

Permissive Lizenzen

Das sind die Lizenzen mit dem höchsten Freiheitsgrad. Nutzer dürfen die Soft- oder Hardware nutzen, verändern, kombinieren und in beliebiger Form weitergeben – also auch verkaufen. Sie sind nicht verpflichtet, etwaige Änderungen auch zu veröffentlichen oder an andere zu lizenzieren. Bekannte permissive Lizenzen sind Apache 2, MIT oder BSD 3 (für Software) bzw. CERN-OHL-P oder Solderpad 2.1 (Hardware).

Stark reziproke Lizenzen (starkes Copyleft)

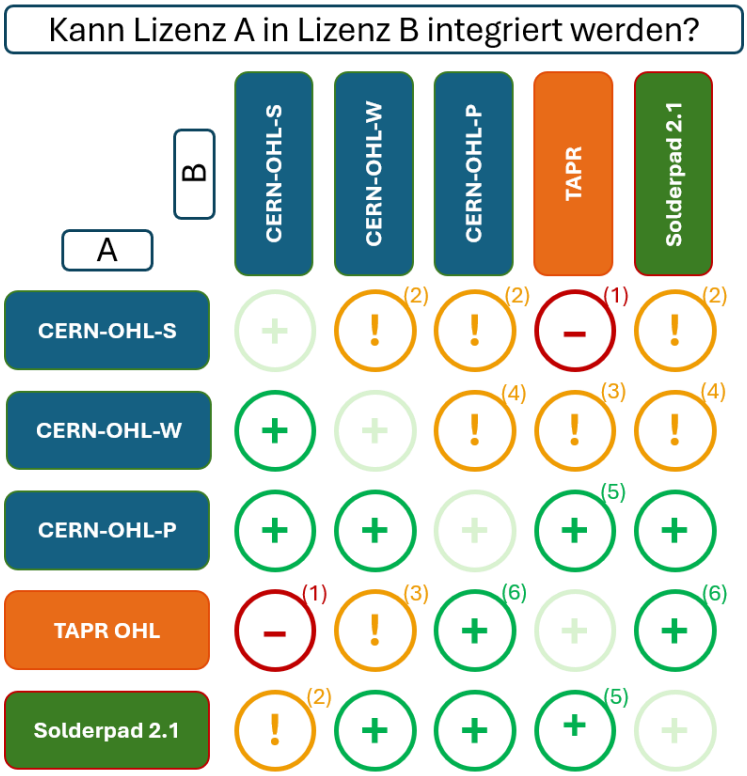
Die sog. Copyleft-Lizenzen sind der Gegensatz zu permissiven Lizenzen: Nutzer dürfen zwar die Soft- oder Hardware nutzen, verändern, kombinieren und weitergeben. Sie müssen dies aber immer unter der ursprünglichen Lizenz tun – auch ihre eigenen Beiträge oder Änderungen. Die Lizenzbedingungen werden quasi an alle „Nachkommen vererbt." Das hat zum Ziel, einen möglichst großen Fundus offener Soft- oder Hardware aufzubauen, beißt sich jedoch häufig mit kommerzieller Nutzung. Unter Umständen zwingen reziproke Lizenzen dazu, das ganze Projekte unter eine stark reziproke Lizenz zu stellen, auch wenn nur ein stark reziprok lizenziertes Bauteil integriert wird (siehe unten Lizenzkompatibilitäten). Bekannte Copyleft-Lizenzen sind GPL 3 (Software) bzw. CERN-OHL- S oder TAPR (Hardware).

Schwach reziproke Lizenzen (schwaches Copyleft)

Schwache Copyleft-Lizenzen sind ein Kompromiss zwischen starkem Copyleft und permissiven Lizenzen. Sie erlauben in Grenzen, dass Änderungen oder Ergänzungen an der ursprünglichen Open Source nicht offengelegt oder dass nur Ergänzungen am Kern einer Technologie veröffentlicht werden müssen. Hier gibt es unterschiedliche Abstufungen: die Mozilla Public License wird z.B. als sehr schwach, die LGPL 3 als „nur" schwach reziprok beurteilt. Bekannte schwach reziproke Lizenzen sind LGPL 3 oder MPL (Software) bzw. CERN-OHL-W (Hardware).

4.2 Kompatibilität von Lizenzen

Lizenz-Kompatibilität beschreibt die Möglichkeit, zwei (oder mehr) urheberrechtlich geschützte Bausteine – etwa ein Hardwarelayout, eine Firmwarebibliothek oder eine Dokumentation – gemeinsam zu veröffentlichen oder zu verbreiten, ohne dass die jeweiligen Lizenzbedingungen einander widersprechen. Für Softwarelizenzen gibt es inzwischen zahlreiche, gut gepflegte Kompatibilitätsübersichten; für Hardwarelizenzen fehlt bislang ein vergleichbarer Standard – die folgende Grafik ist daher selbst erstellt.



Die Frage dabei ist, inwiefern ein Modul unter der Lizenz A in ein Gesamtvorhaben unter der Lizenz B integriert werden kann und darf. Grüne Felder bedeuten, dass eine Integration problemlos möglich ist, bei roten Feldern ist dies unmöglich, da sich die beiden Lizenzen widersprechen. Bei gelben Feldern ist die Situation komplex, meist mit dem Ergebnis, dass eine Integration zwar möglich ist, allerdings für das Modul oder gar das Gesamtvorhaben eine Lizenzänderung – meist eine Lizenzverschärfung hin zu stärker reziproken Lizenzen – notwendig ist.

Bei der Integration ist vorab zu unterscheiden, ob das Modul A integriert wird, z.B. indem es in Designfiles oder die Dokumentation vollständig integriert wird – oder ob es ein unabhängiges Bauteil bleibt mit eigenständigen Designfiles, Dokumentation usw. – so als ob man es als unabhängige Komponente zugekauft hätte.

Im Fall „unabhängige Komponente“ kann eigentlich fast immer kombiniert werden – so wie man auch ein kommerzielles, proprietär lizenziertes Bauteil „aus dem Laden“ immer mit verbauen könnte. Lediglich die CERN-OHL-S-Lizenz erlaubt bei unabhängigen Komponenten nur reine Hardware. Sobald eine Komponente Software mit enthält, gilt sie nicht mehr als unabhängige Komponente.

Erläuterungen:

(1) Die TAPR OHL verlangt, alle Modifikationen ebenfalls unter TAPR zu stellen, die CERN-OHL-S Lizenz verlangt das Gleiche. Keine der beiden Lizenzen erlaubt Relizenzierung zur anderen. Das schließt sich gegenseitig aus.

(2) Wird eine CERN-OHL-S-Komponente in ein anders lizenziertes Design integriert, muss das gesamte kombinierte Werk unter CERN-OHL-S gestellt werden.

(3) TARP bzw. CERN-OHL-W verlangen nur, dass Modifikationen und Dokumentation unter die gleiche Lizenz gestellt werden. Der Rest des Projekts kann weiter unter seiner jeweiligen Lizenz bleiben.

(4) Das W -Modul selbst muss unter W bleiben, auch Modifikationen am W-Modul. Der „Rest“ des Projekts kann seine Lizenz behalten.

(5) TAPR verlangt, dass Modifikationen und Dokumentation unter TAPR stehen. Da P bzw. Solderpad permissiv sind, können P-/Solderpad-Teile unter TAPR relizenziert werden. Das Gesamtprojekt bleibt ein TAPR-Projekt.

(6) TARP verlangt nur, dass Modifikationen und die Dokumentation des TAPR-Moduls unter die gleiche Lizenz gestellt werden. Der „Rest“ des Projektes kann weiter unter Solderpad- oder P-Lizenz bleiben. Es entsteht aber kein permissives Gesamtprojekt.

Bei inkompatiblen Lizenzen stehen im Wesentlichen zwei Wege offen:

1. Das problematische Bauteil durch ein gleichwertiges mit einer kompatiblen Lizenz ersetzen. Oder, wenn möglich, das problematische Bauteil selbst neu entwerfen und bauen, dann kann man die Lizenz selbst bestimmen.
2. Mit dem Rechteinhaber des betroffenen Bausteins verhandeln, um eine Lizenzänderung oder die Erteilung einer zusätzlichen, kompatiblen Lizenz zu erreichen.

In Ausnahmefällen kann auch eine Dual-Licensing-Strategie sinnvoll sein (vgl. Kapitel 5.3). Dort wird erläutert, wie beispielsweise ein stark reziprokes Werk zusätzlich unter einer permissiven Lizenz anbieten, um die Integration in Projekte mit unterschiedlichen Lizenzanforderungen zu ermöglichen.

Ansonsten bleibt noch die Strategie der rationalen Ignoranz: wie wahrscheinlich ist es, dass ein der Offenheit verpflichteter Lizenzinhaber einen sich ganz ähnlich der Offenheit verpflichtenden Nutzer tatsächlich für juristische Feinheiten der kollidierenden Lizenzen zu verklagen? Eine CERN-OHL-S- und eine TAPR-Lizenz sind zwar inkompatibel, aber in der Zielrichtung gleich: sie wollen beide gleichermaßen ein starkes Copyleft für die Hardware.

4.3 Lizenzmodell

Wer ein Werk veröffentlichen will, braucht eine eindeutige Lizenz.

- Der Urheber legt fest, welche Rechte (Vervielfältigung, Bearbeitung, Verbreitung, kommerzielle Nutzung u. a.) eingeräumt werden.
- Eine Lizenz kann exklusiv sein (nur ein Lizenznehmer) oder nicht-exklusiv (Mehrfachnutzung möglich).
- Ohne Lizenz gilt: keine Nutzungsrechte.

Ein zentrales Instrument für die rechtssichere Nutzung, Weiterverbreitung und Abänderung von Open Hardware ist die Wahl einer passenden Lizenz. Es ist **empfohlen, Standardlizenzen zu verwenden**, um das Verständnis und die Einhaltung der Lizenzbedingungen zu erleichtern sowie Inkompatibilitäten zu verhindern. Eine gute Quelle dafür ist die Liste der durch OSI anerkannten Lizenzen:
https://opensource.org/licenses

Für Open-Source-Hardware existieren eigene Standardlizenzen, die sich in ihrer Logik an Open-Source-Software orientieren. Dazu gebräuchlichsten sind die CERN Open Hardware License (in den Varianten OHL-P, OHL-W und OHL-S), die TAPR Open Hardware License und die Solderpad License.

Die **CERN-OHL-P** (Permissive) ist eine offene Lizenz mit geringen Einschränkungen. Sie erlaubt eine freie Nutzung, Veränderung und Kommerzialisierung. Bei der Veränderung des Hardwaredesigns besteht keine Verpflichtung, die Änderungen öffentlich zu veröffentlichen. Allerdings müssen bei der Weitergabe der Hardware (z. B. Verkauf) die geänderten Baupläne und Dokumentation an den Empfänger bereitgestellt werden.

Das Design kann unter diesen Bedingungen proprietär weiterentwickelt werden, sofern die Lizenzbedingungen bei der Verteilung eingehalten werden. CERN-OHL-P-lizenzierte Hardware kann in andere Systeme integriert werden, wobei die ursprünglichen CERN-OHL-P-Teile unter dieser Lizenz bleiben, das Gesamtsystem aber andere Lizenzen annehmen kann.

Die **CERN-OHL-W** (Weak) kombiniert Offenheit mit der Möglichkeit, bestimmte Komponenten proprietär zu halten (sogenanntes ‚Weak Copyleft'). Änderungen an den spezifischen, lizenzierten Dateien müssen unter CERN-OHL-W veröffentlicht werden.

Diese Lizenz ermöglicht es, eigene Hardware-Module in ein größeres, proprietäres Gesamtsystem zu integrieren. Die Änderungen am Modul bleiben offen, während da übergeordnete System proprietär bleiben darf. Dies ist ideal für Unternehmen, die ihre Kernkomponenten offenlegen wollen, aber das Endprodukt schützen möchten.

Die CERN-OHL-S (Strong) verfolgt eine strikte Copyleft-Strategie. Sie stellt sicher, dass das Design und alle abgeleiteten Werke offen bleiben. Alle Änderungen oder Weiterentwicklungen des Designs müssen ebenfalls unter CERN-OHL-S veröffentlicht werden.

Wird die Hardware mit anderen Komponenten zu einem neuen Produkt kombiniert und dieses vertrieben, muss das gesamte veränderte Design unter CERN-OHL-S lizenziert werden. Dies verhindert, dass Open-Hardware-Designs in geschlossene, proprietäre Systeme ‚eingesperrt' werden.

Die TAPR OHL 1.0 ist eine der älteren Open-Hardware-Lizenzen. Sie erlaubt die Nutzung, Veränderung und Kommerzialisierung von Hardware-Designs. Sie verpflichtet jedoch zur Beibehaltung von Urheber- und Lizenzhinweisen, zur transparenten Kennzeichnung von Änderungen und zur Bereitstellung der vollständigen Dokumentation für mindestens drei Jahre nach der letzten Verteilung. Auch sie erlaubt die Einbettung in geschlossene Systeme.

Die Solderpad License (basierend auf der Apache 2.0 Lizenz) ist analog zur CERN-OHL-P aufgebaut und erlaubt eine freie Nutzung mit minimalen Einschränkungen. Sie ist besonders beliebt in der Industrie, da sie auf der Apache-Lizenz aufbaut. Vor allem in für Prozessordesign relevanten Bereichen wie dem akademischen RISC-V-/PULP-Ökosystems ist die Solderpad 2.0 Lizenz der de-facto-Standard.

Wie bei anderen permissiven Lizenzen besteht keine Verpflichtung, Änderungen öffentlich zu teilen; bei Weitergabe der Hardware müssen jedoch die ursprünglichen, unter einer permissiven Lizenz veröffentlichten Baupläne dem Empfänger zur Verfügung gestellt werden. Ein wesentlicher Bestandteil ist der Patent-Grant: Der Lizenzgeber gewährt allen Lizenznehmern eine Patentlizenz für das Design. Dies gilt weltweit, unbefristet und gebührenfrei, solange die Urheber- und Lizenzhinweise erhalten bleiben. (Hinweis: Auch die CERN-OHL 2.0 enthält einen solchen Patent-Grant).

Für die meisten Open-Hardware-Projekte werden heutzutage CERN-OHLs in der Version 2 genutzt. Zahlen für OSHWA-zertifizierte Hardware[5] zeigen, dass davon etwas mehr als die Hälfte die streng reziproke Variante CERN-OHL-S nutzt, weitere 15% die schwach reziproke CERN-OHL-W. Der Rest, knapp 30%, nutzen die permissive Version CERN-OHL-P. Auch auf GitHub sind die meisten sind die Verhältnisse ähnlich: knapp 80% CERN-lizenzierte Projekte (davon S: 45%, W: 20%, P: 35%) stehen ca. 20% Solderpad-Projekten und weniger als 1% TAPR-Projekten gegenüber.[6]

Empfehlung: für die meisten Projekte ist eine der drei CERN-Lizenzen sinnvoll. Diese sind auch untereinander kompatibel. Solderpad ist dann zu erwägen, wenn im Bereich Prozessordesign gearbeitet wird. Die TAPR-Lizenz hingegen ist ein Auslaufmodell, sie wird kaum noch genutzt und kann mit weitestgehend gleichem Ergebnis durch die CERN-OHL-S ersetzt werden.

[5] https://michaelweinberg.org/blog/2025/10/05/leading-licenses/

[6] Die Zahlen wurden im Mai 2026 durch eine automatisierte Auswertung auf GitHub.com erhoben. Sie sind aus mehreren, technischen Gründen nicht vollständig verlässlich, spiegeln aber dennoch das grobe Bild korrekt wider. Andere Auswertungen sind nicht bekannt.

5 Geschäftsmodelle

In der Open-Source-Bewegung werden Quell- bzw. Design-Dateien kostenfrei zur Verfügung gestellt. Für kleine Hobby-Projekte, die von Einzelpersonen in ihrer Freizeit entwickelt werden, funktioniert dieses Modell ohne Weiteres. Viele der heute bedeutendsten und technisch anspruchsvollsten Open-Source-Hardware-Initiativen entstehen jedoch in Forschungseinrichtungen, Unternehmen oder großen Konsortien. Dort werden umfangreiche Ressourcen – Laborinfrastruktur, Prototypen-Fertigung, Test- und Validierungsaufwände – eingesetzt, und die beteiligten Fachkräfte erhalten ein reguläres Gehalt oder Honorar.

Deshalb muss zwischen der Kostenfreiheit der Lizenz (Nutzer:innen können das Design ohne Lizenzgebühr herunterladen und weiterverwenden) und den Entstehungskosten des Projekts unterschieden werden. Letztere werden in der Regel durch Fördermittel, Unternehmenssponsoring, Service- und Support-Modelle oder den Verkauf von zugehörigen Produkten finanziert. Transparente Lizenz- und Finanzierungsangaben sichern einerseits die faire Vergütung des Entwicklerteams und ermöglichen andererseits, dass das Ergebnis weiterhin als offene, reproduzierbare Ressource für die gesamte Community bereitsteht.

Die untenstehenden Geschäftsmodelle beschreiben, wie Open Source Projekte dieses Geld erlangen können, ohne den grundlegenden Charakter von Open Source zu verlieren. Ausführlicher werden die Geschäftsmodelle in einem separaten Dokument vorgestellt.

https://opentransfer.eu/d/OBM

Für das Verständnis der Geschäftsmodelle sind zuerst Überlegungen zum Nutzen und vermarktbaren Ressourcen nötig. Jedes Geschäftsmodell braucht etwas zu Vermarktendes, was die jeweiligen Nutzer als wertvoll, erstrebenswert und damit kaufwürdig erachten. Für Open Source Projekte muss dies zum einen ein Nutzenversprechen beinhalten: die OSH muss aus Perspektive des Nutzers bzw. des Käufers einen handfesten, nachhaltigen Wert besitzen, damit es lohnt, sie zu erlangen und zu nutzen. Zum anderen muss für ein funktionierendes Geschäftsmodell eine vermarktbare Ressource (ein Asset) vorhanden sein. Etwas, was sich abgrenzen, spezifizieren und verkaufen lässt. Das kann zum Beispiel ein physisches Produkt, ein spezifischer Service für einen definierten Zeitraum oder die Arbeitszeit einer Person sein.

Das Nutzenversprechen einer OSS/OSH liegt allem Weiteren zugrunde. Ohne Nutzen keine Nutzer, aber auch: ohne mehr bzw. mindestens gleich viel Nutzen wie konkurrierende (auch kommerzielle) Lösungen keine bzw. nur wenige Nutzer. Dabei muss zumindest der Kernnutzen realisiert werden: die OSS/OSH erfüllt auf technischer Ebene das, was sie soll – so fehlerfrei, einfach nutzbar und schnell wie möglich.

Um interessanter und wertvoller als kommerzielle Alternativen zu sein, benötigt es aber noch Zusatznutzen, z.B. einen oder mehrere der folgenden:

- Geringere Kosten: häufig ist OSH preiswerter als kommerzielle Alternativen. Allerdings ist sie nie kostenlos, weil der Nutzer noch Kosten für die Herstellung, ggf. notwendige Anpassungen, Training und weiteres hat.
- Kein Lock-in-Effekt: mit jeder Hardwarelösung müssen Nutzer sich auf einen Lösungsweg festlegen. Eine Änderung ist mit teils recht hohem Aufwand für Suche nach Alternativen, Testen, Anpassen, Schulung usw. verbunden. Bei kommerziellen Lösungen ist man nach der Festlegung oft von einem quasi monopolistischen Anbieter abhängig. Bei OSH hat der Nutzer das ganze Know-how und kann im Fall von Problemen sich andere Dienstleister oder Anbieter suchen.
- Modularität und Robustheit: Der häufig modulare Aufbau von OSH macht Lösungen in vielen Fällen robuster und einfacher zu warten.
- Transparenz und Nachvollziehbarkeit: Nutzer erhalten Transparenz über Aufbau und Funktionsweise, Änderungen und Anpassungen sind damit leichter vorzunehmen. In sicherheitsrelevanten Bereichen sichert diese Transparenz Vertrauen.
- Community-Ressourcen: eventuell vorhandene Communities können große und vielfältige Ressourcen für Entwicklung und Tests beisteuern, womit ggf. schneller geändert und getestet werden kann.
- Immaterieller Nutzen: das gute Gefühl „an etwas Richtigem teilzuhaben“, die Förderung von Potentialen z.B. in der Wissenschaftscommunity, die strategische Unabhängigkeit oder resilientere kritische Infrastruktur können auch Nutzen sein.

Die konkret vermarktbaren Ressourcen, die in einem Geschäftsmodell ganz konkret verkauft werden? Weil neben dem grundsätzlichen Nutzen von Open Source Projekten braucht es auch – quasi abrechenbare – Ressourcen zum Verkaufen. Das können sein:

- Know-how und Informationen: unterstützen Weiterentwicklung, Anpassung und Implementierung

- Personal mit Know-how: hilft ganz konkret bei Weiterentwicklung, Anpassung und Implementierung
- Fertige Produkte: werden gefertigt und können direkt eingesetzt werden
- Zugänge: Zugriff auf Ressourcen, Entscheidungen und Netzwerke hilft, Einfluss auf zukünftige Entwicklungen zu nehmen
- Vertrauen: durch z.B. Image, Marken, Zertifizierungen, getestetes Zubehör und Material etc. hilft, die Bereitstellung von Produkten zu erleichtern und fördert deren Verkauf an Kunden
- Besondere Rechte: erlauben Dinge, die anderen nicht möglich bzw. erlaubt sind wie z.B. gegen Regeln allgemeiner Open Source Lizenzen (wie z.B. Weiterentwicklungen auch Open Source stellen zu müssen) zu verstoßen, etwas früher zu haben oder etwas Besseres zu bekommen als normale Nutzer mit Open-Lizenzen

Für die Umsetzung der Geschäftsmodelle kann es auch sinnvoll sein, weitere Akteure neben den Entwicklern als Personen und ihre Organisation einzubeziehen. Gründe für zusätzliche Akteure können folgende sein:

- Trennung vom Träger der OSS/OSH: die ursprünglichen Entwickler bzw. die Organisation, der sie angehören, darf, kann oder will bestimmte Geschäftsmodelle nicht umsetzen.
- Finanzierung: manche Organisationen können ggf. notwendige Finanzierungen nur schwierig oder gar nicht erlangen.
- Organisatorische Grenzen: komplexe Regeln wie Tarifverträge oder öffentliches Haushaltsrecht machen bestimmte organisatorische Maßnahmen sehr aufwendig oder unmöglich.

Hier können andere Organisationen diese Aufgaben zumindest teilweise übernehmen. Szenarien wären die folgenden:

- Gründung einer neuen Organisation (Spin-off/Spin-out): wesentliche Aufgaben werden in einer neuen Organisation angesiedelt. Dabei muss auf die Eigentumsverhältnisse geachtet werden, weil öffentliche Einrichtungen bei signifikantem Miteigentum manche Zwänge „vererben“. Wichtig ist sicherzustellen, dass neue Organisation und Entwickler auch langfristig ähnliche Strategien verfolgen.
- Nutzung einer spezialisierten Tochterorganisation: größere Organisationen v.a. aus dem öffentlichen Bereich haben spezialisierte Tochterorganisationen für wirtschaftliche Tätigkeiten, die mitgenutzt werden können.
- Erledigung durch Partnerunternehmen: bestimmte Aufgaben können an unabhängige, partnerschaftlich mit den Entwicklern verbundene Unternehmen ausgelagert werden. Hier

ist die Trennung am einfachsten, allerdings auch das Risiko, dass Entwickler und Partnerunternehmen sich „auseinanderleben“.

Wichtig ist zu klären, in welcher Form das Open Source Projekt weiterhin von dem/den Geschäftsmodell(en) profitiert.

- Die Entwickler und der Betreiber des Geschäftsmodells schließen eine Art Lizenzvereinbarung, die dafür sorgt, dass ein Teil der Erlöse bzw. Gewinne aus dem Geschäftsmodell an die Entwickler weitergereicht wird (Erlösbeteiligung).
- Die Betreiber des Geschäftsmodells übernehmen Kosten des Open Source Projektes (komplett oder nur einen Teil) und finanzieren diese mit Erlösen aus dem Geschäftsmodell (Entwicklungs- oder Supportpersonal beim Betreiber anstellen, Anschaffung von notwendigem Equipment für die Entwickler).
- Die Entwickler bzw. deren Organisation profitieren materiell nicht direkt davon, allerdings fördert das Geschäftsmodell die Nutzbarkeit, die Zuverlässigkeit oder die Verfügbarkeit von OSS/OSH und nutzt dem Open Source Projekt indirekt.

5.1 Dienstleistungen

Der Verkauf von Dienstleistungen begleitend zu einer freien, kostenlosen Lizenz ist das mit Abstand meistgenutzte Geschäftsmodell. Es funktioniert für sehr viele, auch kleinere OSH Projekte. Dabei werden zu einer OSH die unterschiedlichsten Dienstleistungen angeboten, die dem Nutzer helfen, ein fertig einsetzbares Produkt zu erhalten und einzusetzen. Dies sind v.a. Dienstleistungen, die

- dabei helfen, OSH fertigzustellen (implementieren, in Einsatzumgebung einzupassen, Fertigung vorzubereiten, Bauteile zu beschaffen)
- dabei helfen, OSH zu benutzen (Entwicklung von Zubehör / Schnittstellen, Schulung / Training, auf die Anwendungsbedingungen anpassen, Ergebnisse interpretieren und Schlussfolgerungen ziehen, …)
- Qualität sichern und zertifizieren (Konformität mit Standards, Einhaltung von Leistungsparametern)

Für Forschungseinrichtungen naheliegende Dienstleistungen sind klassische Forschungs- und Entwicklungsprojekte bzw. Auftragsforschung. Das können Weiterentwicklungen der zugrundeliegenden Technologie oder Anpassungen an bestimmte Anforderungen der Nutzer sein. Dafür wird die Forschungseinrichtung direkt vom Auftraggeber oder im Rahmen eines öffentlich geförderten Verbundprojektes von einem öffentlichen Geldgeber bezahlt.
Dienstleistungen können von allen möglichen Akteuren angeboten werden – Forschungseinrichtung selbst, Unternehmen im Umfeld der Forschungseinrichtung oder einem unabhängigen Unternehmen / Startup.

Zu berücksichtigen ist allerdings, dass Dienstleistungen schlechter skalieren als ein physisches Produktgeschäft: für ein Mehr an Umsatz muss meist ein ähnliches Mehr an personellem Aufwand betrieben werden. Der zu verkaufende Wert ist das Know-how und die Expertise der beteiligten Personen, welches viel schwieriger zu vervielfältigen ist als ein physisches Produkt in Serienfertigung.

5.2 Herstellung und Verkauf fertiger Hardware

OSH bedeutet, dass die physische Fertigung der Hardware beim Nutzer oder einem Dritten liegt. Dies kann durch die Entwickler bzw. mit ihnen verbundene Akteure übernommen werden. Der Nutzen dieses Geschäftsmodell ist, dass die Nutzer keinen zusätzlichen Aufwand treiben müssen, um eine OSH nutzen zu können – so wie es auch bei kommerziellen Produkten die Regel ist. Damit entfällt der Bedarf für Materialbeschaffung, Anlagen und der für die Fertigung notwendigen Kenntnisse,

Für die Umsetzung ist oft ein Unternehmen als weiterer Akteur notwendig: Die Herstellung fertiger, zuverlässiger OSH-Produkte in größerer Stückzahl ist nicht die Kernkompetenz von Forschungseinrichtungen. Auch die ggf. dafür notwendigen Investitionen für Anlagen können sie schlecht rechtfertigen. In bestimmten Fällen, wenn z.B. für die Fertigung der Zugriff auf spezielle, teure Maschinen notwendig ist, können aber auch forschungsnahe Unternehmen dieses tun.

Für private Unternehmen ist dieses Geschäftsmodell attraktiv, weil die Produktion fertiger Produkte gut skaliert – mit relativ wenig zusätzlichem Aufwand kann ein signifikant höherer Umsatz erzielt werden.

5.3 Duale Lizenzierung

Duale Lizenzierung bedeutet, dass OSH nicht allen Nutzergruppen oder allen Nutzungsarten unter gleichen Bedingungen zur Verfügung steht. Bestimmte Nutzergruppen (Privatpersonen, gemeinnützige Akteure, Bildungs- oder Forschungseinrichtungen, …) erhalten die Hardware unter einer Open Source Lizenz. Andere Akteure, z.B. gewerbliche Unternehmen, müssen gewerbliche Lizenzen erwerben.
Dies kann verstärkt werden durch die Nutzung einer sog. Copyleft Lizenz, die dazu zwingt, abgeleitete Werke unter die gleiche Lizenz zu stellen und kommerzielle Nutzungen ausschließt – parallel werden kommerzielle Lizenzen angeboten.

Eine Variante ist die sog. Time-Release Open Source. Dabei werden Technologien erst nach einer gewissen Zeit oder bei Vorliegen neuerer Versionen als Open Source verfügbar gemacht.

Dieses Geschäftsmodell ist sowohl bei Forschungseinrichtungen als auch kommerziellen Anbietern möglich und verbreitet. Allerdings ist zu berücksichtigen, dass Erwerber kommerzieller Lizenzen dann auch kommerzielle Bedingungen hinsichtlich Qualität, Service und Haftung erwarten. Duale Lizensierung lässt sich gut mit dem Open Core Modell (siehe Abschnitt 5.6) verbinden.

Time-Release Open Source hingegen wird nahezu ausschließlich von kommerziellen Unternehmen genutzt: durch die zeitlich beschränkte Exklusivität sichern sie sich Marktvorteile und Erlöse, später veröffentlichen sie die Technologien.

5.4 Gemeinwohlfinanzierung

Die ersten Entwicklungs- und Verbreitungsschritte einer OSH erfolgen meist aus anderen Kontexten heraus – einem Entwicklungsvorhaben in einem Unternehmen, einem Forschungsprojekt oder privater Neugierde. Mit zunehmender Komplexität des Projekts sowie steigenden Nutzer- und ggf. auch Entwicklerzahlen steigen auch Koordinations-, Kommunikations- und Supportaufwand. Hier kommt oft der Punkt, an dem dieser Aufwand in bisheriger Form nicht mehr (vollständig) getragen werden kann.

Hier kann der Staat, Stiftungen, oder auch spendende Unternehmen sowie Privatpersonen einspringen und durch einmalige bzw. regelmäßige Zahlungen ohne eine direkte Gegenleistung diesen Aufwand decken helfen. Typische Formen dafür sind

- Spenden
- Sponsoring
- Crowdfunding
- öffentliche Finanzierung (Grund- oder Projektfinanzierung)

Für die Finanziers, ob Staat, Stiftungen oder sonstige Organisationen, erwarten sich verschiedene Arten von Nutzen:

- Technologiepolitische Motive (Verhinderung von marktbeherrschenden Stellungen/Monopolen, Kontrolle über bzw. Verhindern der Abwanderung von Technologiekompetenzen, Förderung von F/E-Kapazitäten, etc.)
- (echter) Altruismus
- Kommunikation mit „Altruismus“ (Werbung, PR)
- Sicherung einer gemeinsamen Technologiebasis (v.a. wenn mehrere Unternehmen gemeinsam finanzieren)

Beim Crowdfunding steht zumindest teilweise auch ein unmittelbarer Nutzen für Geldgeber im Raum: früherer oder anderweitig privilegierter Zugang zum entstehenden, in frühen Phasen knappen Produkt ist eine häufige Gegenleistung.

Dieses Geschäftsmodell ist für verschiedene Organisationsformen in unterschiedlichem Maße geeignet. Öffentliche Mittel sind für eine Forschungseinrichtung viel einfacher zu erlangen als für ein Unternehmen, v.a. wegen Wettbewerbs- und Beihilferecht). Spenden und Sponsoring kann grundsätzlich jeder entgegennehmen, wobei einer öffentlichen oder gemeinnützigen Einrichtung in der Regel mehr gespendet wird als einem kommerziellen Unternehmen. Gemeinnützige Einrichtungen (wie z.B. Forschungseinrichtungen) werden oft steuerlich begünstigt.

Der Aufwand für dieses Geschäftsmodell sollte nicht unterschätzt werden, da viele Zwecke (nicht nur OSH) um die Aufmerksamkeit und das Geld von Spendern, Sponsoren, aber auch des Staates konkurrieren. Für viele kleine oder mittlere Projekte wird der finanzielle Erfolg oft nicht das erhoffte Ausmaß annehmen.

5.5 Mitgliedschaft / Community

Ähnlich wie die Gemeinwohlfinanzierung basiert die Mitgliedschaft auf einem indirekten Nutzenversprechen. Ein Mitglied unterstützt allgemein die weitere Entwicklung und Unterhaltung mit Geld. Im Gegenzug können ihm bestimmte Rechte eingeräumt werden. Die Mitgliedschaft kann formalrechtlich in einem Verein, aber auch lediglich durch vertragliche Vereinbarung erfolgen.

Der Nutzen kann sowohl mittelbar als auch unmittelbar sein. Unmittelbarer Nutzen ist z.B. die Möglichkeit, als Teil der Mitgliedschaft eine bestimmte Zahl oder Menge an Dienstleistungen (Support, Entwicklung) und/oder Zugriff auf bessere Varianten der OSH (früherer Zugriff auf neue Versionen, Zugriff auf zusätzliche Features – siehe auch *Freemium*) zu erhalten. Unternehmen bieten teilweise auch Rabatte auf Ersatzteile und Zubehör an (siehe *Geschäftsmodell 5.7 unten*). Ein mittelbarer Nutzen kann z.B. die Mitsprache bzw. Mitbestimmung darüber sein, wie und in welche Richtung die weitere Entwicklung der OSH erfolgen soll.

Das Mitgliedschaftsmodell ist eher für Forschungseinrichtungen oder ähnliche, gemeinnützig orientierte Organisationen geeignet. Die zu erzielenden Erlöse sind meist überschaubar und

dienen eher der Deckung der notwendigen Kosten für Services oder das Communitymanagement als der Erzielung attraktiver Gewinne.

5.6 Open Core / Freemium

Prinzip des Open Core Modells ist, dass um einen Open Source Kern eine Reihe proprietärer Peripherieelemente angeboten werden. Teilweise ist der Kern ohne diese Peripherie gar nicht nutzbar. Teilweise ist das Geschäftsmodell auch als Freemium bekannt, einem Kofferwort aus Free und Premium. Dabei sind die Basisfunktionen kostenlos, zusätzliche (Premium-) Komponenten kosten Geld.

Eine Sonderform ist, wenn der (möglichst bekannte) Name eines OSH-Projektes markenrechtlich geschützt und nur gegen Bezahlung lizenziert wird. Damit kann zwar jedermann die OSH frei nutzen, darf sie jedoch nicht unter ihrem bekannten Namen verwenden und vermarkten.

Für Anwender sind technische oder organisatorische Zusatzfunktionalitäten attraktiv, die die Herstellung und den Einsatz einer OSH komfortabler oder effizienter gestalten. Teilweise werden diese Peripherieelemente auch zwangsweise dazu verkauft im Sinne eines Lock-in-Effektes: der Kern ist ohne Peripherie für bestimmte Anwendungen sonst gar nicht nutzbar. Nutzer umgehen das teilweise, indem sie notwendige Peripherie selbst entwickeln.

Mit Lizenzierung der Marke können Nutzer vom guten Ruf einer etablierte OSH-Marke profitieren. Die technische Leistungsfähigkeit der OSH Produkte ist zwar die gleiche, die Glaubwürdigkeit bei Endkunden aber mit Marke deutlich höher.

Das Open Core Modell ist vorwiegend für kommerzielle Akteure geeignet, weniger für Forschungseinrichtungen. Die Markenlizenzierung kann auch durch Forschungseinrichtungen effektiv genutzt werden.

5.7 Verkauf von Zubehör oder Merchandise

Parallel zu einer OSH werden begleitend Produkte verkauft. Dies können Merchandise-Artikel sein, aber auch Zubehör aller Art wie Komponenten für den Bau oder Verbrauchsmaterial für den Betrieb einer OSH. Das ist aus verschiedenen Gründen nützlich:

- Komfort: notwendiges Zubehör wie Bauteile, Zusatzkomponenten oder zugehörige Verbrauchsmaterialien wird aus einer Hand (dem OSS/OSH-Anbieter) gekauft, was den Bezug dieser sowieso notwendigen Dinge vereinfacht (Bezug aus einer Quelle, idealerweise geprüfte Qualität oder Kompatibilität).
- Unterstützung: durch den Kauf über den OSH Anbieter wird dieser mittelbar finanziell unterstützt.
- Signaling: mit Merchandise kann der Käufer seine Zugehörigkeit zu einer Community oder die finanzielle Unterstützung dieser in seinem Umfeld demonstrieren.

Dieses Geschäftsmodell kommt eher für kommerzielle Anbieter in Frage, da mit dem Handel von Zubehör die normalen kommerziellen Risiken verbunden sind (Vorfinanzierung von Produktion, Haftungsrisiken, einbrechender Abverkauf etc.). Lediglich Merchandise ist auch im Umfeld von Forschungseinrichtungen gebräuchlich.

Während der Verkauf von Zubehör auch bei kleineren Projekten funktioniert, ist das Merchandising nur bei wirklich großen und bekannten Projekten wirtschaftlich lohnend.

6 Technische Umsetzung

6.1 Technologien und Herstellungsverfahren

Im Gegensatz zu Software spielt für Hardware die technologische Ausgestaltung und die damit verbundene Produktionstechnologie eine wesentliche Rolle. Während Software schnell in ein Produkt zu verwandeln ist, muss Hardware teils aufwändig hergestellt werden. Das Herstellverfahren bestimmt viele Eigenschaften:

Die **Leistungsfähigkeit** der OSH hängt wesentlich vom Herstellverfahren ab. Hier spielen Materialfragen, aber auch Fragen der Fertigungspräzision, der Veredlung oder notwendige Umweltbedingungen eine Rolle.

Die **Kosten der Herstellung** hängen neben den Materialkosten auch vom Produktionsverfahren ab. Verfahren der Massenfertigung senken bei großen Stückzahlen die Stückkosten erheblich, wogegen 3D-gedruckte Bauteile bei sehr kleinen Stückzahlen günstiger sind. Viele komplexe Herstellverfahren sind Privatanwendern nicht direkt, sondern nur über kommerzielle Unternehmen, Forschungseinrichtungen oder gemeinschaftlich genutzte Einrichtungen wie Fab Labs oder Makerspaces zugänglich. Auch die Fertigungstiefe, d.h. welche Teile zugekauft oder selbst gefertigt werden müssen, bestimmt mit über Kosten und Einfachheit der Herstellung.

Das gewählte Verfahren hat unmittelbaren Einfluss auf:

- die **Kosten** pro Stück,
- die **Leistungsfähigkeit** der Hardware,
- die **Qualität** und notwendige **Präzision** der Bauteile,
- die **Zeit** bis das erste funktionierende Gerät vorliegt und
- die **Komplexität**, die zukünftige Nutzer beim Zusammenbau bewältigen müssen.

Für die breite Nutzbarkeit von OSH müssen solche Fragen möglichst früh bedacht werden, weil eine Änderung wesentlicher Teile des Herstellverfahrens erst spät im Entwicklungsprozess zu aufwändigen Umplanungen führt. Für diese Entscheidung wiederum sollte man wissen, wer die **Zielgruppe**(n) einer OSH sein sollen:

- Wie fachkundig ist die Zielgruppe?
  Spezialist:innen in der Anwendung oder auch in der Herstellung?
- Wie groß ist die Zielgruppe?
  Wenige Nischennutzer:innen oder viele Privatanwender:innen?
- Wie viele Produkte braucht ein typischer Nutzer:innen?
  Eines oder viele? Einmalig oder regelmäßig wieder?

- Wie wertvoll ist das Produkt für die Zielgruppe?
  Wie viel Geld können bzw. wollen Mitglieder der Zielgruppe ausgeben?

Im Folgenden werden die gängigsten Optionen in Form von drei groben Fertigungsklassen vorgestellt. Anschließend erhalten Sie fünf Leitfragen, die helfen, die passende Klasse zu wählen, und einen kurzen Hinweis zur Fertigungstiefe – also wie viel Sie selbst herstellen wollen.

6.1.1 DIY-Klasse – Do-It-Yourself

In der DIY-Klasse übernimmt das Projekt-Team die gesamte Fertigung. Typische Werkzeuge sind ein Desktop-3-D-Drucker (meist für Kunststoff), ein Laserschneider für Blech oder ein Breadboard-Aufbau für elektronische Prototypen.

Wann ist DIY sinnvoll?

- Wenn nur ein einzelnes Exemplar oder ein kleiner Prototyp (weniger als zehn Stück) benötigt wird.
- Wenn das Budget sehr begrenzt ist – die einzigen Kosten entstehen durch Filament, Schneid- oder Druckmaterial und den Zeitaufwand für den Zusammenbau.
- Wenn die Zielgruppe technisch versiert und bereit ist, Bauteile selbst zu drucken, zu löten und zu montieren.

Vorteile

- Schnellere Iterationen – ein neues Design lässt sich innerhalb von Stunden testen.
- Geringe Anfangsinvestitionen – kein externes Werkzeug muss gemietet oder gekauft werden.

Nachteile

- Eingeschränkte Materialwahl (häufig nur thermoplastische Kunststoffe).
- Geringere Fertigungsgenauigkeit, insbesondere bei feinen Geometrien oder hohen mechanischen Belastungen.

6.1.2 Fab-Lab-Klasse – Der Mittelweg

Hier wird das Design in einem lokalen Fab-Lab, Makerspace oder bei einem kleinen Auftragsfertiger umgesetzt. Die Fertigung kann industriellen 3-D-Druck (z. B. SLS-Verfahren), CNC-Fräsen oder die Bestückung von Leiterplatten umfassen.

Wann passt diese Klasse?

- Wenn Sie zwischen zehn und etwa fünfhundert Stück benötigen – zum Beispiel für eine kleine Serie von Lehr- und Forschungskits.
- Wenn das Bauteil aus einem Material bestehen soll, das im Desktop-Druck nicht verfügbar ist (z. B. Metall-Filament, Hochtemperatur-Kunststoff).
- Wenn die Zielgruppe zwar technisch versiert ist, aber nicht jede einzelne Fertigungsstufe selbst durchführen soll.

Vorteile

- Größere Materialvielfalt und höhere Präzision als beim reinen DIY-Ansatz.
- Relativ kurze Lieferzeiten, weil viele Fab-Labs in Universitäts- oder Forschungsnähe angesiedelt sind.

Nachteile

- Höhere Stückkosten, da Material- und Maschinenkosten bereits anfallen.
- Etwas mehr organisatorischer Aufwand – Sie müssen ein Produktions-Daten-Paket (Design-Dateien, Stückliste und Montage-Anleitung) erstellen und an das Fab-Lab übergeben.

6.1.3 Serien-Klasse – Professionelle Massenfertigung

Für größere Stückzahlen (mehr als 500 Stück) oder wenn das Produkt hohen regulatorischen oder Qualitätsanforderungen genügen muss, ist die Zusammenarbeit mit einem Auftragsfertiger sinnvoll. Typische Verfahren sind Spritzguss für Kunststoffgehäuse, serielle Leiterplattenfertigung und, bei sehr spezialisierten Geräten, Halbleiter- oder Mikrofertigung.

Wann greifen Sie zu dieser Klasse?

- Wenn das Produkt kommerziell vertrieben werden soll und die Stückzahlen die Investition in Werkzeuge (z. B. Spritzgussformen) rechtfertigen.
- Wenn das Produkt besonders komplexe bzw. aufwendige Herstellverfahren erfordert, um die gewünschte Leistung zu erbringen.
- Wenn das Bauteil besonders hohe mechanische Belastungen tragen muss oder Zertifizierungen (z. B. Medizinprodukte) erforderlich sind.
- Wenn die Zielgruppe professionelle Anwender sind, die ein komplett vormontiertes und standardisiertes Produkt erwarten.

Vorteile

- Sehr niedrige Stückkosten bei hohen Stückzahlen – die hohen Anfangsinvestitionen amortisieren sich schnell.
- Hohe Wiederholgenauigkeit und die Möglichkeit, strenge Qualitäts- und Sicherheitsstandards zu erfüllen.

Nachteile

- Hohe Anfangsinvestition (Werkzeug-Kosten, Werkzeug-Einrichtung).
- Längere Vorlaufzeiten (Werkzeugherstellung, Produktionsaufbau).

6.1.4 Entscheidungs-Leitfaden

Die Wahl des Fertigungsverfahrens lässt sich anhand von fünf zentralen Einfluss-Faktoren steuern. Jeder Faktor weist klar darauf hin, ob ein DIY-Ansatz, ein Fab-Lab-Ansatz oder eine professionelle Serienfertigung die sinnvollste Lösung darstellt.

- Materialbedarf
  - Reiner Kunststoff lässt sich problemlos im eigenen 3-D-Drucker oder in einem kleinen Makerspace fertigen – das spricht für die DIY-Klasse.
  - Metall, hochtemperaturbeständige Kunststoffe oder spezielle Legierungen erfordern höhere Präzision und Prozessstabilität. Hier ist ein Fab-Lab notwendig, und bei größeren Stückzahlen wird schnell die Serien-Klasse attraktiv, weil erst dann die Wirtschaftlichkeit von Werkzeugen (z. B. Spritzgussformen) erreicht wird.

- Geplanter Produktionsumfang
  - Bis zu ca. 10 Exemplare lassen sich leicht im DIY-Modus umsetzen.
  - Mehrere Dutzend bis einige hundert Stück profitieren vom Fab-Lab-Ansatz, weil dort hochwertige Materialien und präzise Fertigung zur Verfügung stehen, ohne dass bereits erhebliche Werkzeug-Investitionen nötig sind.
  - Überschreitet die Menge etwa 500 Stück, sinken die Stückkosten bei Serienfertigung so stark, dass sich die Anschaffung von Formen und die Einbindung eines Auftragsfertigers rentieren.

- Budget für die Erstproduktion
    - Unter 500 € reicht meist nur die Ausstattung für einen Desktop-3-D-Drucker oder einen kleinen Fab-Lab-Zugang – damit bleibt DIY die einzige praktikable Option.
    - Ein Finanzrahmen von 500 € bis 5 000 € ermöglicht den Zugang zu Fab-Lab-Ressourcen und deckt Material- sowie Maschinenzeit-Kosten ab.
    - Erst ab einem Budget von über 5 000 € wird eine Serien-Klasse lohnenswert, weil die notwendigen Werkzeug-Investitionen dann gedeckt sind.

- Technische Kompetenz der Zielgruppe
    - Laien, die lediglich ein vormontiertes Kit zusammenbauen wollen, benötigen ein Produkt, das bereits weitgehend fertiggestellt ist – also aus dem Fab-Lab oder aus der Serienfertigung.
    - Fachleute, die selbst drucken, löten oder mechanisch bearbeiten können, können problemlos einen DIY-Ansatz nutzen; ein Fab-Lab-Verfahren bietet ihnen zusätzlich komplexere Geometrien, ohne die komplette Eigenproduktion übernehmen zu müssen.

- Leistungs-, Qualitäts- und Zertifizierungsanforderungen
    - Produkte, die in sicherheitskritischen Bereichen eingesetzt werden (z. B. medizinische Geräte, Schutzsysteme), unterliegen häufig strengen Normen und müssen dokumentierte, wiederholbare Fertigungsprozesse nachweisen. Solche Vorgaben können in der Regel nur durch eine professionelle Serienfertigung erfüllt werden.
    - Um bestimmter Leistungsparameter oder Qualitäten zu erreichen, sind manchmal maschinell sehr aufwendige Verfahren notwendig wie z.B. Mikro- bzw. Nanofertigung, Hochtemperatur- oder Hochdruckprozesse. Die dafür notwendigen Anlagen haben viele Nutzer nicht bzw. können sie mit nur kleinen Stückzahlen nicht sinnvoll auslasten. Lohnfertiger können hier gute Konditionen bieten.

Zusammenfassung

Wenn die meisten Einflussfaktoren – also Material, Stückzahl, Budget, Zielgruppenkompetenz und Qualitätsanforderungen – auf hohe Ansprüche hindeuten, ist die Serien-Klasse die vernünftige Wahl. Sind die Anforderungen eher simpel (einfaches Material, geringe Stückzahl, begrenztes Budget), reicht ein DIY-Ansatz völlig aus. Projekte, die zwischen diesen Extremen liegen, profitieren am besten vom Fab-Lab-Ansatz, weil er sowohl Materialvielfalt als auch mittlere Stückzahlen ermöglicht, ohne die hohen Werkzeugkosten einer Serienfertigung zu verlangen.

Fertigungstiefe – Wie viel bauen Sie selbst?

Bereits beim Entwurf sollte festgelegt werden, welche Bauteile **intern** gefertigt und welche **extern** bezogen werden.

- **Komplett-Selbstbau** – Sie drucken das Gehäuse, löten die Elektronik und montieren alles selbst. Das ist die flexibelste Variante, erfordert jedoch umfangreiches technisches Know-how und mehr Zeit für den Endnutzer.
- **Hybrid-Ansatz** – Sie lassen die mechanisch anspruchsvollen Teile (z. B. Metallgehäuse) von einem Dienstleister fertigen, während Standard-Elektronik-Komponenten (Widerstände, Steckverbinder) von den Nutzer*innen selbst bestückt werden. Dieses Modell reduziert den Aufwand für den Endnutzer, behält aber einen gewissen DIY-Charakter.
- **Voll-Auslagerung** – Das komplette Gerät wird von einem Auftragsfertiger produziert und geliefert. Der Nutzer muss nur noch das Gerät auspacken und in Betrieb nehmen. Dieses Modell ist besonders attraktiv, wenn das Produkt an Unternehmen oder Institutionen mit hohen Qualitätsansprüchen verkauft werden soll.

Praktischer Hinweis: Die Entscheidung über die Fertigungstiefe sollte **vor** dem ersten Prototypen getroffen werden. Ein später Wechsel zwischen den Modellen führt häufig zu zusätzlichen Kosten und Verzögerungen, weil Design- und Stücklisten angepasst werden müssen.

6.2 Technische Dokumentation

Eine vollständige, transparente Dokumentation ist das Herzstück jeder Open-Source-Hardware. Sie ermöglicht es anderen, das Projekt zu verstehen, nachzubauen und weiterzuentwickeln. Die nachfolgende Checkliste fasst die unabdingbaren Dokumente zusammen, die in einem öffentlichen Repository liegen sollten.

- Projekt-Übersicht
  - Kurzbeschreibung des Zwecks, Zielgruppe und aktueller Entwicklungsstand (z. B. „Prototype“, „Beta“, „Production“).
  - Kontakt- bzw. Verantwortlichkeitsinformationen.
- Lizenz-Hinweis
  - Für jedes Artefakt (Hardwaredesign, Firmware, Dokumentation) die jeweils geltende Open-Source-Hardware-Lizenz (z. B. CERN-OHL-v2, CC-BY-4.0).

- Optional: Ein kurzer SPDX[7]-Header am Dateianfang, damit die Lizenz maschinell ausgelesen werden kann.

- Original-Design-Dateien
  - Die nativen CAD- oder Schaltplandateien (z. B. KiCad, FreeCAD, SolidWorks).
  - Kommentarzeilen, die kritische Design-Entscheidungen erläutern.
- Export-Dateien für die sofortige Nutzung
  - PDF- oder SVG-Ansichten, STEP- bzw. STL-Modelle, Gerber-Dateien – damit auch Personen ohne das Original-CAD-Tool das Design öffnen können.
- Stückliste (Bill-of-Materials, BOM)
  - CSV-Datei mit allen Bauteilen, Hersteller-/Bestellnummern, Stückzahl, Preishinweis und mindestens einem alternativen Lieferanten.
- Montage-Anleitung
  - Schritt-für-Schritt-Beschreibung, notwendige Werkzeuge und Sicherheitshinweise.
  - Idealerweise ein Explosionsdiagramm, das die Zusammenfügung visualisiert.
- Inbetriebnahme-Handbuch
  - Schnell-Start-Anweisungen, Grundkonfiguration, Betriebsmodi und eine kompakte Fehlersuch-Tabelle.
- Release-Notes / Change-Log
  - Auflistung der wichtigsten Änderungen pro Version, jeweils verlinkt zum entsprechenden Commit-Hash.
  - Optional: Permanente DOI (z. B. Zenodo) für jede veröffentlichte Version.
- Software-/Firmware-Quellcode (falls vorhanden)
  - Vollständiger, kommentierter Code, Build-Anleitung (Makefile, PlatformIO, CMake).
  - Lizenz-Header in jeder Datei.

[7] SPDX (Software Package Data Exchange) ist ein offener ISO/IEC-Standard (5962:2021) der Linux Foundation, der genutzt wird, um Komponenten, Lizenzen, Copyrights und Sicherheitsreferenzen in einer Software-Stückliste (SBOM) zu kommunizieren. Er erleichtert das Lizenzmanagement und die Compliance, indem er ein standardisiertes Format für Open-Source-Software bietet.

- Community-Werkzeuge
    - Link zum Issue-Tracker, zum Projekt-Wiki und zu den Contribution-Guidelines (z. B. Pull-Request-Workflow, Code-of-Conduct).
- Marken- und Trademark-Hinweis
    - Hinweis, welches Logo (z. B. OSHW-Logo) verwendet werden darf und welche Teile des Projekts unter welcher Lizenz stehen.
- Langzeit-Archivierung
    - Jede Release-Version wird zusätzlich in einem dauerhaften Repository (z. B. Zenodo) archiviert und mit einem DOI versehen.

6.3 Betrieb & Wartung (Business-Operations)

Der Betrieb eines Open-Source-Hardware-Produkts umfasst alles, was nötig ist, damit das physische Gerät nach der ersten Auslieferung zuverlässig, sicher und nachhaltig genutzt werden kann. Anders als bei reiner Software endet die Verantwortung nicht mit dem Release des Quellcodes – sie erstreckt sich über die gesamte Lebensdauer des Produkts: von der ersten Inbetriebnahme über regelmäßige Pflege bis hin zu möglicher Weiterentwicklung oder dem Auslaufen des Angebots.

Hinweis zum Umfang:

Der Betrieb und die Wartung von Open-Source-Hardware ist kein starres Pflichtprogramm, sondern ein anpassbares Rahmenwerk. Für ein kleines Lern-Kit genügt meist ein leichtes Set aus Lizenz, CAD-Dateien, Stückliste und kurzer Montageanleitung. Wenn das Projekt jedoch wächst – mehr Stückzahlen, höhere Sicherheitsanforderungen oder ein kommerzieller Vertrieb – können Sie schrittweise weitere Bausteine (Monitoring, Serviceprozesse, Zertifikate) hinzufügen. .

Warum der Betrieb so wichtig ist

- Verfügbarkeit – Das Gerät soll jederzeit funktionsfähig sein; das bedeutet eine verlässliche Uptime und die Möglichkeit, Ersatzteile schnell zu beschaffen.
- Sicherheit – Mechanische, elektrische oder softwareseitige Gefahren müssen durch regelmäßige Prüfungen und Updates kontrolliert werden.
- Nachhaltigkeit – Geplante Wartungs- und Reparaturmaßnahmen verlängern die Lebensdauer und reduzieren den Materialverbrauch.

- Weiterentwicklung – Offene, transparente Prozesse ermöglichen der Community, Fehler zu beheben, neue Funktionen zu integrieren und das Design an veränderte Anforderungen anzupassen.
- Rechtliche Rahmenbedingungen – Zertifizierungen und Lizenz-Compliance müssen über die gesamte Nutzungsdauer hinweg nachgewiesen werden.

Kernelemente eines funktionierenden Betriebs

- Bereitstellungs- und Liefermodell
    - Entscheiden Sie, ob das Gerät als Bausatz, vormontiertes Kit oder komplett ausgeliefertes Produkt angeboten wird.
- Betriebsorganisation
    - Legen Sie Rollen und Verantwortlichkeiten fest (z. B. SRE/Hardware-Support, Qualitätsmanager, Sicherheitsbeauftragter).
    - Definieren Sie einen klaren Eskalations-Workflow für Störfälle.
- Wartungs- und Service-Strategie
    - Planen Sie regelmäßige Inspektionen, Kalibrierungen und den Austausch von Verschleißteilen.
    - Stellen Sie ein Verfahren für Firmware-Updates bereit.
- Monitoring & Fehlerdiagnose
    - Integrieren Sie Embedded-Sensoren und Logging-Mechanismen.
    - Nutzen Sie Remote-Diagnose-Tools, um Zustandsdaten zu sammeln und Frühwarnungen auszulösen.
- Sicherheits- und Patch-Management
    - Scannen Sie systematisch nach Schwachstellen auf Hardware- und Firmware-Ebene.
    - Rollen Sie Sicherheits-Patches zeitnah aus.
- Backup & Wiederherstellung
    - Sichern Sie kritische Konfigurations- und Kalibrierdaten.
    - Definieren Sie ein Disaster-Recovery-Verfahren (RPO/RTO).
- Support- und Community-Service
    - Bieten Sie ein Self-Service-Portal, ein Issue-Tracker-System und eine FAQ-Datenbank.

- Stellen Sie klare Guidelines für Wartungs- bzw. Erweiterungs-Pull-Requests bereit, sodass die Community unkompliziert mitwirken kann.

- Kosten- und Ressourcen-Controlling
  - Kalkulieren Sie laufende Kosten (Ersatzteile, Service-Verträge, Infrastruktur) transparent.
  - Vergleichen Sie diese Kosten mit der geplanten Lebensdauer, um die Wirtschaftlichkeit zu sichern.

- Kontinuierliche Verbesserung
  - Schließen Sie Feedback-Schleifen: Aus Monitoring-Daten, Nutzer-Rückmeldungen und Service-Tickets resultierende Erkenntnisse fließen in die nächste Design-Iteration ein.

7 Publikation, Distribution und Community

7.1 Publikation

Nachdem das Hardwaredesign fertig ist, stellt sich die Frage: Wie kommt das Produkt zu den Menschen, die es nutzen, weiterentwickeln und langfristig erhalten wollen?

- Sichtbarkeit – Ohne einen öffentlichen Ort, an dem Konstruktions- und Fertigungsdaten liegen, bleibt das Projekt unsichtbar. Selbst ein perfekt funktionierendes Design kann nicht von anderen genutzt werden, wenn niemand es finden kann.
- Reproduzierbarkeit – Open-Source-Hardware lebt davon, dass jede:r das Gerät nachbauen kann. Dazu müssen sämtliche CAD-Dateien, Stücklisten, Schaltpläne, Firmware und die Lizenz in einer Form vorliegen, die ohne Rätsel gelöst werden kann.
- Anerkennung und Nachweis – Wissenschaftliche Publikationen, Förderanträge oder industrielle Partner verlangen zitierbare Quellen (DOI). Nur wenn das Projekt archiviert ist, lässt es sich konsequent referenzieren.
- Langfristiger Erhalt – Ein Gerät ist erst dann wirklich Open Source, wenn die zugehörigen Informationen über Jahre hinweg verfügbar bleiben – unabhängig davon, ob das ursprüngliche Team noch existiert.
- Mehrwert durch Gemeinschaft – Sobald das Projekt öffentlich ist, können andere Entwickler, Forschende oder Unternehmen Beiträge leisten: Fehler beheben, neue Features hinzufügen, Tests unter anderen Bedingungen durchführen und das Produkt weiterentwickeln.

Wesentliche Bausteine einer Publikation:

Wesentlicher Ausgangspunkt ist die Bestimmung der Zielgruppe. Entscheiden Sie, ob Ihr Projekt hauptsächlich Maker:innen, Wissenschaftler:innen oder Industriepartner anspricht. Die Zielgruppe bestimmt, welche zusätzlichen Plattformen Sie priorisieren und welche Art von Dokumentation besonders hervorgehoben werden muss (z. B. mehr technische Details für Forschung vs. leicht verständliche Montage-Guides für Maker).

Klare Lizenz und Metadaten

Jede Datei enthält einen kurzen Lizenz-Header (z. B. SPDX-License-Identifier: CERN-OHL-2.0). Zusätzlich erstellen Sie eine „CITATION.cff“-Datei, in der Titel, Autor:innen, Lizenz, DOI-Platzhalter und Schlagwörter hinterlegt sind. Archiv- und Index-Dienste (Zenodo, OSF, Figshare) übernehmen diese Metadaten automatisch, sodass Ihr Projekt zitierfähig wird.

Versionierung

Nutzen Sie das bekannte Semantic Versioning (MAJOR.MINOR.PATCH). Ein kurzer CHANGELOG.md dokumentiert, welche Änderungen in welchem Release enthalten sind – das gibt Nutzer*innen sofort Aufschluss, ob ein Update für sie relevant ist.

Zentrale Ablage

Ein öffentliches Git-Repository (GitHub oder GitLab) ist das Kern-Repository: Hier liegen alle CAD-, Gerber-, Schaltplan- und Firmware-Dateien, das Issue-Tracking und die Pull-Request-Funktion. Durch das integrierte Review-System wird die Qualität der Beiträge gesichert.

Sichtbare Zusatzplattformen

- Wikifactory – veröffentlicht das Projekt als „Produkt", bietet einen webbasierten 3-D-Viewer, Stückliste und einen kurzen Pitch; die Issue-Funktion ist praktisch für Fertigungs-Fragen.
- Hackaday.io – dient als Storytelling-Hub; ein Blog-ähnlicher Beitrag mit Fotos, Entwurfs-Story und QR-Code zum Zenodo-DOI bringt das Projekt schnell in die Maker-Community.
- OSHWA-Product-Page – nach erfolgreicher OSH-Zertifizierung wird das offizielle OSHW-Logo zusammen mit dem Zertifizierungs-Link im README und auf der OSH-HWA-Seite angezeigt – ein vertrauensbildendes Signal für Industrie und Forschung.

Langzeit-Archiv

Für jede stabile Release-Version erzeugen Sie ein ZIP-Archiv und laden es bei Zenodo hoch. Zenodo vergibt einen permanenten DOI und übernimmt automatisch die Metadaten aus der Datei „CITATION.cff". Als sekundäres Backup können Sie das gleiche Paket bei OSF oder Figshare ablegen – besonders sinnvoll, wenn ergänzende Mess- oder Videodaten hinzugefügt werden sollen.

Begleitpublikationen

Die reine Bereitstellung im Repository reicht für die wissenschaftliche Sichtbarkeit nicht aus. Ergänzen Sie daher:

- Journal-Artikel (z. B. Journal of Open Hardware oder HardwareX) – im Manuskript werden Projektlink, Zenodo-DOI und ein kurzer Lizenz-Hinweis angegeben.

- Konferenz-Beiträge – Poster oder Kurzpräsentationen enthalten QR-Codes, die direkt zum aktuellen Release führen.
- Pre-Prints (arXiv, HAL) – vor dem Peer-Review veröffentlichte Manuskripte verlinken bereits den Zenodo-DOI.

Durch diese mehrgleisige Strategie wird Ihr Projekt sowohl in der Open-Source-Community als auch in der wissenschaftlichen Literatur auffindbar und zitierfähig.

Kommunikation und Marketing

Um die Zielgruppen der OSH zu erreichen, ist eine gezielte Kommunikation über zusätzliche Kanäle unerlässlich. Dazu zählen institutionelle Kanäle wie Pressemitteilungen und Newsletter des Technologietransferbüros, die lokale Stakeholder erreichen, sowie wissenschaftliche Netzwerke wie ResearchGate oder ORCID zur Sicherung der Sichtbarkeit in der Fachcommunity. Über LinkedIn als professionelles Netzwerk können gezielt Industriepartner und Förderorganisationen angesprochen werden. Die Vorstellung auf Konferenzen und Workshops mit direkten Links zum Repository fördert den persönlichen Austausch, und eine dedizierte Projektseite auf der institutionellen Website dient als vertrauenswürdige Anlaufstelle für langfristige Erreichbarkeit.

7.2 Community–Aufbau & Management

Open-Source-Hardware lebt von der Mitwirkung Dritter. Die Veröffentlichung von Konstruktions- und Fertigungsdaten ermöglicht es Interessierten nicht nur, das Gerät nachzubauen, sondern aktiv an seiner Weiterentwicklung teilzunehmen oder durch Tests die Zuverlässigkeit zu erhöhen.

Entwicklercommunities sind nützlich auf verschiedenen Ebenen:

- Mehr Ressourcen: zusätzliche Entwickler:innen, Labor- und Fertigungsgeräte.
- Neue Ideen: Einbringung frischer Konzepte, Materialien und Fertigungsverfahren.
- Vielfältige Tests: Prüfung unter unterschiedlichsten Anwendungsbedingungen.
- Nachhaltigkeit: geringere Abhängigkeit von einer einzelnen Organisation.
- Reichweite: besser Verbreitung über die Netzwerke einer Vielzahl an Mitwirkenden.

Allerdings entstehen Communities in den allerwenigsten Fällen von allein. Der Aufbau und die Unterhaltung einer Community erfordern teils erheblichen Aufwand und ggf. auch Kosten. Dies ist notwendig u.a. für

- Management der Community und interne Kommunikation
- Koordination von Entwicklungspfaden, Co-Entwickler:innen und Ressourcen als auch Governance zum Gesamtprojekt

Die Bedingungen sind im Hardwarebereich schwieriger als in der Softwarewelt: Potenzielle Co-Entwickler:innen benötigen meist nicht nur einen PC, sondern auch Zugang zu Fertigungs- und Testequipment. Zudem sind modulare Hardware-Entwicklungen komplexer. Der Aufbau und die Unterhaltung einer Community erfordern daher nicht unerheblichen Aufwand und ggf. auch Ressourcen.

Besonderheiten von Hardware-Communities

Im Vergleich zu Software ist die Einstiegshürde bei Hardware höher. Dies muss bei der Planung der Community-Strategie berücksichtigt werden:

- Ressourcen: Mitwirkende benötigen oft Bauteile, Werkzeuge und Testequipment, die nicht jeder privat besitzt.
- Dokumentation: Lückenhafte Stücklisten oder undokumentierte Fehlermuster erschweren den Einstieg erheblich. Eine klare, nachvollziehbare Dokumentation (siehe Kapitel 6.2) ist die Grundvoraussetzung.
- Motivation: Eine Community bildet sich nicht aus Pflicht, sondern aus Interesse. Klare Mehrwerte (z. B. Zugang zu Wissen, gemeinsame Problemlösung, wissenschaftliche Anerkennung) sind notwendig, um Beteiligung zu stimulieren.

Onboarding & Erreichbarkeit

Damit Interessierte sofort wissen, wo sie anfangen können, sollten einfache Zugangswege bereitstehen. Dies senkt die Hemmschwelle für den ersten Beitrag:

- Einrichtungshilfen: Dokumente wie CONTRIBUTING.md (Schritte zum Einreichen von Beiträgen) und CODE_OF_CONDUCT.md (respektvolles Miteinander) definieren die Spielregeln von Beginn an.
- Strukturierte Anfragen: Vordefinierte Issue-Templates für Fehlerberichte, Feature-Wünsche oder Dokumentationsverbesserungen sorgen für einheitliche Informationen und erleichtern das Sortieren von Anfragen.

- **Erste Schritte Guide:** Ein einfaches Dokument (z. B. PDF oder Markdown), das den Ablauf (z.B. „Zielgruppe bestimmen → Repository klonen → erstes Issue anlegen") Schritt für Schritt beschreibt.

Alle diese Dokumente sollten bereits im **README.md** unter einem Hinweis wie „Wie man mitmachen kann" verlinkt werden.

Kommunikationskanäle & Interaktion

Der Austausch innerhalb der Community sollte auf verschiedenen Kanälen stattfinden, je nach Art der Fragestellung:

- **Technische Diskussionen:** Issue-Tracker und Pull-Request-Kommentare sind für konkrete Änderungen am Design geeignet.
- **Allgemeine Fragen:** Diskussionsforen (z. B. GitHub Discussions) oder Mailinglisten eignen sich für allgemeine Fragen, die nicht direkt einen Code-Change erfordern.
- **Echtzeit-Kommunikation:** Chat-Tools (z. B. Matrix, Discord) können den persönlichen Austausch fördern, sollten aber für Forschungseinrichtungen datenschutzkonform eingesetzt werden.

Wichtig ist, dass die Kanäle nicht zu fragmentieren. Ein zentraler Punkt (z. B. GitHub Repository) sollte als „Single Source of Truth" für alle technischen Entscheidungen dienen.

Governance & Entscheidungsfindung

Ein funktionierendes Governance-Modell ist unverzichtbar, um Chaos und unkoordinierte Abweichungen zu verhindern. Es regelt, wer welche Entscheidungen trifft:

- **Rollenverteilung:** Klare Aufgaben wie Community Lead (Gesamtstrategie), Review Manager (Qualitätssicherung), Documentation Owner (Aktualität der Anleitungen) und Release Manager (Veröffentlichung) sorgen für Übersicht.
- **Entscheidungsprozesse:** Transparente Wege (z. B. Review von Pull Requests durch mehrere Personen) verhindern unkoordinierte Forks und stellen sicher, dass jede Änderung geprüft wird.
- **Transparenz:** Entscheidungen und deren Begründungen sollten dokumentiert werden (z. B. in docs/meeting-notes.md), um Nachvollziehbarkeit zu gewährleisten.

In Forschungseinrichtungen sind zusätzlich klare Verantwortlichkeiten für Lizenzfragen, Dokumentationsanforderungen und Publikationsprozesse festzulegen (z. B. Projektleitung, Technologietransfer).

Wachstum & Nachhaltigkeit

Um beurteilen zu können, ob die Community wächst und effektiv arbeitet, sollten einige Kennzahlen regelmäßig erfasst werden. Dies dient nicht der Kontrolle, sondern der Selbstreflexion:

- Aktivität: Geschlossene vs. offene Issues geben Aufschluss über die Reaktionsgeschwindigkeit des Teams.
- Nutzung: Zenodo-Downloads pro Release messen, wie häufig das veröffentlichte Paket tatsächlich verwendet wird.
- Beiträge: Der Anteil der Nutzer, die mindestens einen Pull Request eingereicht haben, zeigt die Tiefe der Beteiligung.

Die Prinzipien (Transparenz, Lizenzklarheit, Versions- und Langzeitarchivierung) sind universell und gelten für jedes OSH-Projekt. Die Ausprägung der unterstützenden Maßnahmen (Onboarding-Material, Governance, KPIs, zusätzliche Plattformen, Finanzierungsplan) richtet sich nach Größe, Zielgruppe und verfügbaren Ressourcen. Durch ein gestuftes Vorgehen kann ein Projekt klein starten und mit wachsendem Aufwand und zunehmender Community-Größe schrittweise professioneller werden, ohne die Grundidee des offenen Zugangs zu verlieren.

8 Glossar

Rechtliche Grundlagen & geistiges Eigentum

Intellectual Property (IP) – Geistiges Eigentum

Der Oberbegriff für alle rechtlichen Schutzinstrumente, die immaterielle Schöpfungen schützen. Dazu gehören das Urheber-, Patent-, Marken- und Designrecht sowie das Datenbank-Recht.

Markenrecht

Schützt Zeichen – Namen, Logos, Slogans – die Waren oder Dienstleistungen eindeutig kennzeichnen und von anderen Unternehmen unterscheiden. Das Markenrecht gewährt keinen Schutz für die technische Umsetzung eines Produkts.

Patent

Ein aktiv beantragtes Schutzrecht, das dem Inhaber ein zeitlich befristetes Monopol (in der Regel 20 Jahre) auf Herstellung, Nutzung und Verkauf einer technischen Erfindung gewährt.

Produkthaftung

Gesetzliche Verpflichtung des Herstellers, für Personen- und Sachschäden zu haften, die durch ein fehlerhaftes Produkt entstehen. Die Haftung besteht unabhängig davon, ob das Produkt als Open-Source veröffentlicht ist.

Produkthaftungsgesetz (ProdHaftG)

Deutsches Gesetz, das die Haftung von Herstellern für Schäden regelt, die durch fehlerhafte Produkte verursacht werden.

Rechteinhaber

Natürliche oder juristische Person, die das Verwertungsrecht an einem Werk besitzt. Der Rechteinhaber muss nicht zwingend der Urheber sein; bei einem „Work-for-Hire“-Verhältnis kann das Unternehmen oder die Forschungseinrichtung die Rechte besitzen.

Verwertungsrecht

Vom Urheber eingeräumtes Recht, ein Werk zu vervielfältigen, zu verbreiten, öffentlich

wiederzugeben usw. In Deutschland geht das Verwertungsrecht bei einem „Work-for-Hire“-Vertrag automatisch auf den Arbeitgeber über, sofern vertraglich nichts anderes geregelt ist.

Werk-für-Hire / Dienst-/Werkvertrag

Rechtlicher Rahmen, bei dem die vom Arbeitnehmer oder Auftragnehmer geschaffenen Rechte automatisch auf den Auftraggeber (Arbeitgeber, Forschungseinrichtung) übergehen, wenn dies vertraglich vereinbart ist. Entscheidend für die Frage, wer die Lizenz erteilen darf.

Dual-Use

Güter oder Technologien, die sowohl zivil- als auch militärisch eingesetzt werden können. Der Export solcher Waren unterliegt speziellen Genehmigungsverfahren (z. B. EU-Dual-Use-Verordnung) zur Verhinderung von Rüstungsexporten.

Export-Kontroll-Bestimmungen

Rechtliche Vorgaben (z. B. US EAR, EU-Dual-Use-Verordnung), die den internationalen Versand und die Nutzung von Gütern mit militärischem oder sicherheitsrelevantem Potential regeln. OSH-Projekte, die dual-use-fähige Bauteile enthalten, müssen diese Bestimmungen prüfen und ggf. eine Exportgenehmigung einholen.

Lizenz & Lizenz-Typen

CC-Lizenzen (Creative Commons)

Standardisierte Urheber-Lizenzen für kreative Werke wie Texte, Bilder, Videos und Datensätze. Sie reichen von völlig offen (CC0 = Public Domain, CC BY = Namensnennung) bis zu restriktiver (z. B. CC BY-NC-ND = keine kommerzielle Nutzung, keine Bearbeitung).

Duale Lizenzierung

Ein Werk wird gleichzeitig unter einer Open-Source-Lizenz (für die Community) und einer kommerziellen Lizenz (gegen Gebühr, mit zusätzlichen exklusiven Rechten) angeboten. Die Open-Source-Lizenz erlaubt kostenlose Nutzung und Weiterentwicklung; die kommerzielle Lizenz ermöglicht proprietäre Nutzung ohne die Auflagen der Open-Source-Lizenz.

Lizenz

Vertragliches Rechtsinstrument, mit dem der Rechteinhaber festlegt, welche Nutzungen (z. B.

Nutzung, Weitergabe, Bearbeitung) erlaubt sind und welche Auflagen (Namensnennung, Copyleft, Patent-Grant usw.) gelten.

Lizenz-Typen

Permissive-Lizenzen stellen nur geringe Auflagen; Nutzung, Veränderung und kommerzielle Weitergabe sind ohne Pflicht zur Weitergabe von Änderungen erlaubt (Beispiele: MIT, Apache 2.0, BSD 3, CERN OHL P, Solderpad 2.1). Stark reziproke (starkes Copyleft) Lizenzen verlangen, dass alle abgeleiteten Werke unter derselben Lizenz veröffentlicht werden (Beispiele: GPL 3.0, CERN OHL S, TAPR OHL). Schwach reziproke (leichtes Copyleft) Lizenzen behalten das Copyleft nur für den Kern bzw. die Bibliothek, während Erweiterungen permissiv lizenziert werden können (Beispiele: LGPL 3.0, MPL 2.0, CERN OHL W).

License Facts Label (Lizenz-Fakten-Badge)

Ein kurzer, maschinenlesbarer Hinweis – häufig als Markdown-Badge – der Lizenz, Autor*innen und Jahr in jede Datei (Quell-, Design-, Dokumentationsdatei) integriert. Meist wird dazu ein SPDX-Header verwendet.

Nutzungsrecht

Die im Lizenzvertrag festgelegte Erlaubnis, ein urheberrechtlich geschütztes Werk zu nutzen, zu verändern und weiterzugeben.

Open-Source-Hardware (OSH)

Physische Hardware, deren komplette Baupläne (Schaltpläne, Layout-Dateien, Stücklisten, Firmware-Quellcode usw.) öffentlich zugänglich und unter einer Open-Source-Lizenz stehen, sodass jeder das Werk studieren, modifizieren, herstellen und verkaufen kann.

Projekt- und Dokumentations-Infrastruktur

Auxiliary Design Files

Export- bzw. Ansicht-Formate wie PDF, SVG, DXF, STL oder Gerber, die ohne das Original-CAD-Programm gelesen werden können. Sie dienen ausschließlich zur Ansicht bzw. Schnell-Fertigung und ersetzen die Original-Design-Files nicht.

Bill of Materials (BOM)

Stückliste, die alle Bauteile eines Projekts tabellarisch auflistet – inkl. Referenz-Designator, Hersteller, Bestell-Nummer, Menge, Preis und Link zum Datenblatt.

CI-Pipeline

Kontinuierliche-Integrations-Kette (z. B. GitHub Actions, GitLab CI), die bei jedem Commit automatisierte Prüfungen wie Design-Rule-Checks, Lint-Durchläufe, Firmware-Builds oder PDF-Export ausführt und die Ergebnisse sichtbar macht.

Issue-Tracker

System zur Erfassung, Priorisierung und Diskussion von Bugs, Feature-Wünschen und offenen Fragen (z. B. GitHub Issues, GitLab Issues).

Original Design Files

Die nativen CAD- bzw. Quell-Dateien (z. B. .kicad_sch, .kicad_pcb, .step, .sldprt), die ohne weitere Konvertierung zur vollständigen Änderung des Projekts erforderlich sind.

Reifegrad-Badge

Visuelle Kennzeichnung des Entwicklungsstands eines Projekts (Prototype, Beta, Production), meist als Shield-Badge im README angezeigt.

Repository

Strukturiertes, versioniertes digitales Archiv (etwa ein Git-Repository auf GitHub oder ein Zenodo-Datensatz), das alle für das Verständnis, den Nachbau, die Weiterentwicklung und die Verbreitung einer Open-Source-Hardware-Lösung notwendigen Dateien und Metadaten (README, LICENSE, AUTHORS, Baupläne, …) enthält.

Wiki

Im Repository integriertes, leicht editierbares Dokumentations-Portal (GitHub-Wiki, MkDocs, Docusaurus) für ergänzende Informationen, FAQs und langfristige Wissenssammlung.

Brand-Logo (OSHW-Logo)

Offizielles Kennzeichen, das anzeigt, dass ein Produkt (oder ein Teil davon) unter einer OSH-Lizenz veröffentlicht ist. Es muss unverändert und in korrekter Größe verwendet werden.

Geschäfts- und Vertriebsmodelle

Geschäftsmodelle

Beschreiben, wie ein Unternehmen mit einem Produkt langfristig Wert schafft. Dazu gehören Erlösströme (Hardware-Verkauf, Service-Verträge, duale Lizenzierung, Beratung usw.), Kostenstrukturen, eingesetzte Ressourcen und Vertriebskanäle.

Duale Lizenzierung (siehe Abschnitt „Lizenz & Lizenz-Typen“)
stellt ein typisches Geschäftsmodell dar, bei dem das gleiche Werk sowohl Open-Source- als auch kommerziell lizenziert wird.

Sonstige häufig genutzte Begriffe

Open Hardware Repository (OHR)

Ein zentrales Index- und Aggregations-Portal, das OSH-Projekte aus verschiedenen Quellen (GitHub, Wikifactory, Zenodo u. a.) zusammenfasst, Metadaten standardisiert und durchsuchbar macht.

Open-Source-Hardware-Definition (OSHWA)

Die offizielle Definition von Open-Source-Hardware, die Mindestanforderungen an offene Baupläne, Lizenzierung und Dokumentation festlegt.

Reproduzierbarkeit

Die Fähigkeit, ein Produkt exakt gemäß den veröffentlichten Dokumenten (Baupläne, Stückliste, Build-Guide) nachzubauen. In OSH-Projekten wird dies häufig durch CI-Ergebnisse und öffentlich zugängliche Rohdateien sichergestellt.

9 Anhang

9.1 Informationsportale zu Lizenzen

9.1.1 Lizenz-Picker & allgemeine Informationsportale

- Choose a License – Interaktive Hilfestellung, die anhand Ihrer Projekt-Ziele passende Open-Source-Software-Lizenzen vorschlägt.
  https://choosealicense
- GitHub – Choose a License – Offizielle GitHub-Seite, gleiches Angebot wie *choosealicense.com*, aber direkt im Git-Workflow integriert.
  https://github.com/github/choosealicense.com
- Open Source Initiative (OSI) – Lizenz-Liste – Vollständige Auflistung aller von der OSI anerkannten Software-Lizenzen inkl. Kurz-Infos und Links zum Volltext.
  https://opensource.org/licenses
- SPDX License List – Maschinell lesbare Lizenz-IDs, die in Build- und CI-Tools verwendet werden können – praktisch für automatisierte Lizenz-Reports.
  https://spdx.org/licenses/
- FSF – Guide zu freien Lizenzen – Ausführliche Erläuterungen zu GPL-Familie, LGPL, AGPL und deren rechtlichen Konsequenzen.
  https://www.fsf.org/licensing
- Linux Foundation – Open-Source-Licensing Guide – White-Paper mit Überblick über gängige Lizenzen, Kompatibilitätsfragen und Best-Practices für Unternehmen.
  https://www.linuxfoundation.org/resources/open-source-guides/
- Debian Free Software Guidelines (DFSG) – Kriterien, nach denen Debian Software als „frei“ klassifiziert – nützlich, um die Freiheit einer Lizenz zu prüfen.
  https://www.debian.org/social_contract#guidelines

9.1.2 Kompatibilitäts- und Analyse-Tools

- Interoperable Europe – Lizenz-Finder – EU-Projekt, das Lizenzen filterbar macht, Vergleiche erlaubt und nach Anwendungsbereich sowie Rechtsrahmen sortiert.
  https://interoperable-europe.ec.europa.eu/collection/eupl/solution/licensing-assistant/find-and-compare-software-licenses abgerufen

- Interoperable Europe – Kompatibilitäts-Check – Online-Tool, das prüft, ob zwei gewählte Lizenzen (Software + Hardware) miteinander vereinbar sind.
  https://interoperable-europe.ec.europa.eu/collection/eupl/solution/licensing-assistant/compatibility-checker
- tldrlegal – Kurz-Zusammenfassungen aller gängigen OSS-Lizenzen in leicht verständlicher Sprache (Hinweis: gelegentlich fehlt juristische Präzision).
  https://www.tldrlegal.com abgerufen
- OSS Review Toolkit (ORT) – Kommandozeilen-Tool, das Projekte scannt, Lizenz- und Sicherheits-Risiken automatisiert erkennt und SBOMs erstellt.
  https://oss-review-toolkit.org/ort/
- Licensee (GitHub-Action) – Automatischer Lizenz-Scanner für Git-Repos, der im CI-Workflow unbekannte oder unzulässige Lizenzen meldet.
  https://github.com/licensee/licensee
- Tidelift – Lizenz-Kompatibilitäts-Matrix – Interaktive Matrix, die anzeigt, welche Lizenz-Kombinationen miteinander kompatibel sind – praktisch bei gemischten OSS/OSH-Stacks.
  https://tidelift.com/license-compliance
- OSSA – Open-Source-Compliance-Plattform – Cloud-basiertes Tool, das sowohl Lizenz- als auch Sicherheits-Compliance prüft; kosten-frei für reine Open-Source-Projekte.
  https://fossa.com

9.1.3 Open-Source-Hardware (OSH) – Lizenzen und Ressourcen

- CERN Open Hardware Licence (CERN-OHL) – Der de-Facto-Standard für OSH, verfügbar in drei Varianten (v1, v2, v2-with-exception). Copyleft-Optionen für Hardware.
  https://cern-ohl.web.cern.ch
- Übersicht der drei großen OSH-Lizenzen – Gegenüberstellung von CERN-OHL, TAPR-HA und anderen verbreiteten Hardware-Lizenzen, inkl. Vor- und Nachteilen.
  https://curriculum.openhardware.space/articles/06-licenses-and-standards/oh-licenses/
- OSHW Certification – FAQ – Offizielle Fragen-und-Antwort-Sammlung der Open-Source-Hardware-Association zu Lizenzwahl, Marken- und

Zertifizierungsanforderungen.
https://certification.oshwa.org/faq/

- Open-Source-Hardware – Lizenz-FAQ der Open Hardware Community – Community-Erfahrungen zu Lizenz-Interpretationen und Best-Practices beim Publizieren von Hardware-Designs.
  https://forum.oshwa.org/t/license-faq

9.2 Zertifizierung von Open-Source Hardware

Zweck

Eine Zertifizierung gibt einem Open-Source-Hardware-Projekt (OSH) ein glaubwürdiges Vertrauenssignal. Sie belegt, dass das Design vollständig offen, reproduzierbar und rechtlich eindeutig geklärt ist – ein wichtiges Kriterium für Fördermittel, industrielle Partner und Nutzer, die das Produkt in ihre eigenen Anwendungen integrieren wollen.

Zertifizierungswege

Für die meisten OSH-Entwicklungen reicht die Selbst-Zertifizierung über die *OSHWA-Open-Source-Hardware-Declaration* aus. Hier legt das Projekt im öffentlichen Repository Lizenz- und Metadaten-Informationen offen, beantragt das offizielle OSHW-Badge und erhält sofort ein sichtbares Qualitätssiegel.

Wenn das Projekt darüber hinaus in den regulierten Markt (z. B. Medizintechnik, industrielle Elektronik) eingeführt werden soll, ist eine formale Zertifizierung sinnvoll. Dazu gehören das OSHWA-*Certified Product*-Siegel (gegen eine moderate Jahresgebühr), sowie klassische Sicherheits- und Konformitätsprüfungen wie TÜV-EMC-Tests, CE-Markierung nach Low-Voltage-Directive und ggf. ein ISO 9001-Qualitäts-Management-System.

Kurzüberblick über relevante Programme

Programm	Ziel	wichtigste Anforderungen	Kosten / Prüfungsmodus
OSHWA-Declaration	Alle OSH-Projekte	Offenlegung von Design-Files, Lizenz (SPDX-Header), vollständige Metadaten, Nachweis der Reproduzierbarkeit	kostenlos, Selbst-Deklaration (optional Review)
OSHWA-Certified Product	Projekte mit breiter öffentlicher Nutzung	Vollständige Metadaten, Lizenz-Konformität, Dokumentations-Vollständigkeit, reproduzierbarer Build-Prozess	150 – 300 €/Jahr, Review durch OSHWA-Expert*innen, Badge-Ausstellung
TÜV-Sicherheits/EMC	Elektronische Geräte für den EU-Markt	Einhaltung IEC/EN-Normen, EMV-Tests, Sicherheits-Check	abhängig vom Umfang, Labor-Prüfung + Audit
CE-Markierung (LVD, RoHS)	Geräte, die in der EU verkauft werden	Grenzwerte für Spannung, Schadstoffe, EMV	Tester + technische Dokumentation, Konformitätserklärung
ISO 9001	Unternehmen, die Serienfertigung planen	Qualitäts-Management-System, Dokumentations- und Audit-Pflicht	Jahresgebühr + externer Audit

Für die meisten Community-Entwicklungen ist die OSHWA-Declaration bereits ausreichend; die übrigen Zertifikate kommen erst zum Tragen, wenn ein Produkt in den regulierten Markt überführt wird.

Check-Liste für die OSHWA-Declaration (Kurzfassung)

1. Lizenz – Jede Datei (Schaltplan, CAD, Firmware, Dokumentation) enthält einen gültigen SPDX-Header, der auf die gewählte Open-Hardware-Lizenz (z. B. CERN-OHL-v2) verweist.
2. Metadaten – Im Repository-Root liegt eine metadata.yaml nach dem OSHWA-Schema vor (Projekt-Name, Version, Reifegrad, Kontakt, DOI, Lizenz).
3. Design-Files – Alle Quell-CAD-Dateien sind im Ordner hardware/ abgelegt und kommentiert.

4. Fertigungs-Daten – Gerber- bzw. STEP-Export, Stückliste (CSV) und eine Fertigungs-Anleitung existieren.
5. Build- und Prüf-Instruktionen – Schritt-für-Schritt-Anleitungen, DRC/DFT-Ergebnisse und Validierungs-Berichte liegen im docs/- bzw. validation/-Verzeichnis bereit.
6. Reproduzierbarkeit – Der komplette Build-Prozess kann über ein CI-Log (GitHub Actions) nachgeprüft werden.
7. Versionierung – Jeder Release wird mit einem SemVer-Tag versehen und automatisch über Zenodo mit einem DOI versehen.
8. Badge & Hinweis – Das offizielle OSHW-Badge wird im README.md eingebunden.

Sind diese Punkte erfüllt, kann das Projekt die Declaration online einreichen und erhält das OSHW-Badge.

Ablauf einer formalen OSHWA-Zertifizierung

1. Vorbereitung – Die oben genannte Check-Liste wird abgearbeitet, fehlende Dokumente ergänzt und ein Ordner certification/ angelegt, in dem das ausgefüllte Zertifizierungs-Formular, das Badge und das offizielle Zertifikat versioniert werden.
2. Einreichung – Das öffentliche Repository-URL sowie das PDF-Formular werden auf der OSHWA-Zertifizierungsplattform hochgeladen.
3. Review – Das OSHWA-Board prüft Lizenz-Konformität, Vollständigkeit der Metadaten und die Nachvollziehbarkeit des Builds.
4. Feedback – Eventuelle Korrekturen werden nachgebessert; das Projektteam erhält Rückmeldungen innerhalb von etwa zwei Wochen.
5. Zertifikat – Nach erfolgreichem Review wird das *Certified-Product-Badge* (SVG/PNG) sowie ein offizielles Zertifikat (PDF) ausgestellt.
6. Publikation – Das Badge wird im README.md, auf der Projekt-Webseite und im Release-Tag eingebunden; der zugehörige Zenodo-DOI bleibt dauerhaft erhalten.
7. Wartung – Bei wesentlichen Änderungen (neue Lizenz, neue Hardware-Revision) muss das Zertifikat erneuert oder das Projekt erneut zertifiziert werden.

Weiterführende Links

- OSHWA – Definition & FAQ: https://www.oshwa.org/definition/
- OSHWA – Zertifizierungs-Prozess: https://certificates.oshwa.org/
- SPDX-Lizenzliste: https://spdx.org/licenses/
- Zenodo – DOI-Minting: https://zenodo.org/help/quickstart

- TÜV – Elektronik-Zertifikate: https://www.tuv.com/de/de/elektronik.html
- EU-CE-Markierungs-Guide: https://ec.europa.eu/growth/single-market/ce-marking_en

9.3 Lizenzleitfäden und Entscheidungshilfen

FOSSA Guide to Open Source Licences
https://fossa.com/learn/open-source-licenses/

Choose a license
https://choosealicense.com
https://github.com/github/choosealicense.com

Interoperable Europe – Lizenzen finden und vergleichen & Kompatibilitätscheck
https://interoperable-europe.ec.europa.eu/collection/eupl/solution/licensing-assistant/find-and-compare-software-licenses
https://interoperable-europe.ec.europa.eu/collection/eupl/solution/licensing-assistant/compatibility-checker

tl:dr legal – Software Lizenzen in einfacher Sprache
(sehr ausführlich, aber manchmal fehlerbehaftet)
https://www.tldrlegal.com

OSS Review Toolkit
https://oss-review-toolkit.org/ort/

CERN Open Hardware Lizenzen – der Quasi-Standard unter den Hardwarelizenzen
https://cern-ohl.web.cern.ch

Übersicht über die drei „großen“ Hardwarelizenzen
https://curriculum.openhardware.space/articles/06-licenses-and-standards/oh-licenses/
https://ospo.docs.cern.ch/

9.4 Quellen & Leitfäden Open Source Communities

Software Communities

einfache Tipps zum Aufbau einer Community
https://zammad.com/de/blog/5-tipps-zum-aufbau-und-pflege-einer-open-source-community

Kurze Grundlagen für Communities
https://government.github.io/best-practices/community-building/

Open Source Community aus einer öffentlichen Einrichtungen heraus
https://interoperable-europe.ec.europa.eu/collection/open-source-observatory-osor/3-building-your-own-public-sector-oss-community

Leitfaden von Bitkom e.V. (S. 45-60) – Fokus auf Governance und Tools
https://www.bitkom.org/sites/main/files/2022-06/220624-Bitkom-Leitfaden-Open%20Source-3.0_0.pdf

Hinweise zum Aufbau einer Community
https://github.blog/open-source/maintainers/four-steps-toward-building-an-open-source-community

Best Practices zu OSS Communities
https://opensource.guide/best-practices/
https://opensource.guide/de/building-community

Ausführliches Handbuch zu Communities
https://guidebook.theopensourceway.org

Älterer Leitfaden zu Communities aus dem Umfeld der HS Augsburg
https://hhoegl.informatik.hs-augsburg.de/oss/StuermerMyrach_OpenSourceCommunityBuilding.pdf

Dokumentation

Best Practice zu OSH Dokumentation (2017)

https://depositonce.tu-berlin.de/items/a2831304-6a7a-436c-8048-66e2df432830

https://github.com/jbon/Best-Practices-of-Open-Source-Mechanical-Hardware

Impressum

Datum der Veröffentlichung: September 2026
Version 1.0

Herausgeber

Technologietransfer, GSI Helmholtzzentrum für Schwerionenforschung GmbH
www.gsi.de/innovation

OpenTransfer

Im Rahmen von OpenTransfer wurden Strategien und Geschäftsmodelle für den Transfer im Kontext von Open Science entwickelt, Werkzeuge für die konkrete Unterstützung von Transferprozessen geschaffen sowie Organisationsstrukturen für OpenTransfer in den Forschungseinrichtungen eingeführt.

Alle Informationen, Guidelines und Werkzeuge finden Sie im Wiki:

www.OpenTransfer.eu

Das OpenTransfer-Team

Dr. Kathrin Göbel, Albrecht Haag, Dr. Tobias Engert
GSI Helmholtzzentrum für Schwerionenforschung GmbH, Darmstadt

Dr. Thomas Jahn, Dr. Ivonne Bieber
IPHT Leibniz-Institut für Photonische Technologien e.V., Jena

Dr. Vladimir Voroshnin, Dr. Björn Wolf
HZDR Helmholtz-Zentrum Dresden-Rossendorf e.V., Dresden

Peter Häfner
INNOcentric GmbH, Leipzig

Haftungsausschluss

Dieses Werk gibt allgemeine Empfehlungen. Es ersetzt keine Rechts- oder Fachberatung. Für konkrete Projekte sind ggf. zusätzliche Prüfungen nötig.

Copyright

Lizenz

Das zugrundeliegende Verbundvorhaben »OpenTransfer – Strategien für den Transfer von Forschungsergebnissen im Open-Science-Kontext« wurde mit Mitteln des Bundesministeriums für Forschung, Technologie und Raumfahrt (BMFTR) unter Förderkennzeichen 03IO2204 gefördert. Die Verantwortung für den Inhalt liegt bei den Autor:innen.

Gefördert durch:

Bundesministerium
für Forschung, Technologie
und Raumfahrt