**Technische Universität Ilmenau**
Fakultät für Informatik und Automatisierung
Fachgebiet Neuroinformatik und Kognitive Robotik

# Entwicklung einer CrossLab-kompatiblen integrierten Entwicklungsumgebung für das GOLDi-Remotelab

Masterarbeit zur Erlangung des akademischen Grades Master of Science

**Pierre Helbing**

Betreuer: Dr. Detlef Streitferdt

Verantwortlicher Hochschullehrer:
Prof. Dr.-Ing. habil. Daniel Ziener

Die Masterarbeit wurde am DD.MM.YYYY bei der Fakultät für Informatik und Automatisierung der Technischen Universität Ilmenau eingereicht.

Danksagung
Dieser Abschnitt **kann** genutzt werden, um denjenigen Personen Dank auszusprechen, die Sie bei der Erstellung der Arbeit unterstützt haben.

Erklärung: „Hiermit versichere ich, dass ich diese wissenschaftliche Arbeit selbständig verfasst und nur die angegebenen
Quellen und Hilfsmittel verwendet habe. Alle von mir aus anderen Veröffentlichungen übernommenen Passagen sind als solche gekennzeichnet.“

Ilmenau, DD.MM.YYYY .....

Pierre Helbing

# Inhaltsverzeichnis

# Abkürzungsverzeichnis

| | |
|---|---|
| **CNN** | Convolutional Neural Network |
| **SVM** | Support Vector Maschine |

# Kapitel 1

# Einleitung

Das Grid of Online Laboratory Devices Ilmenau (GOLDi) ist ein hybrides Online Labor. Es ermöglicht Studierenden reale Hardwaremodelle mit verschiedenen Kontrolleinheiten zu steuern. Vor dem CrossLab Projekt waren nur Experiment mit einem Hardwaremodell und einer Kontrolleinheit möglich. Aufgrund der neuen CrossLab Architektur ist es allerdings nun möglich Experimente freier zu konfigurieren. Dabei werden sogenannte Laborgeräte definiert, die gewisse Services anbieten. Über diese Services können verschiedene Laborgeräte zu einem Experiment zusammengefügt werden. Nehmen wir zum Beispiel ein 3-Achs-Portal sowie einen Microcontroller. Beide diese Geräte können einen Service anbieten, der ihre Pins definiert. Über diese Services können dann die Pins der beiden Geräte verbunden werden.

WIDE ist eine eigens für das GOLDi Remotelab entwickelte integrierte Entwicklungsumgebung (IDE). Sie ermöglicht es Nutzern direkt im Browser Programme für die Kontrolleinheiten zu schreiben. Weiterhin wird auch die Kompilierung des Quellcodes sowie das Hochladen des Kompilats auf die Kontrolleinheit unterstützt. Allerdings ist WIDE auf die alte Architektur des GOLDi Remotelab ausgelegt, in welcher ein Experiment nur aus jeweils einem Hardwaremodell und einer Kontrolleinheit bestand. In der neuen CrossLab Architektur kann jedoch ein Experiment mehrere Hardwaremodelle und Kontrolleinheiten enthalten.

Um die neuen Anforderungen, die aus der CrossLab Architektur hervorgehen, zu addressieren ist eine Anpassung bzw. Neukonzeption von WIDE nötig. Dafür werden in Kapitel 2 einige grundlegende Begriffe eingeführt. Danach wird in Kapitel 3 eine Systematische Literaturrecherche durchgeführt. In Kapitel 4 werden die Anforderungen

an eine IDE im Kontext des GOLDi Remotelab sowie der CrossLab Architektur gesammelt. In Kapitel 5 werden bestehende Lösungen auf ihre Anwendbarkeit überprüft und verglichen. In Kapitel 6 wird die Lösung konzipiert. In Kapitel 7 wird die Implementierung der konzipierten Lösung beschrieben. In Kapitel 8 wird der Erfüllungsgrad der Anforderungen ausgewertet. Schließlich wird in Kapitel 9 eine Zusammenfassung der Arbeit sowie ein Fazit gegeben.

# Kapitel 2

# Grundlagen

## 2.1 WebRTC

## 2.2 IndexedDB

## 2.3 CRDT

## 2.4 Debugger

## 2.5 WebAssembly

## 2.6 Language Server

## 2.7 CrossLab

## 2.8 IDE

## 2.9 GOLDi Remotelab

## 2.10 Debug Adapter Protocol

# Kapitel 3

# Systematische Literaturrecherche

1. Defining research questions
    - Was sind Probleme von integrierten Entwicklungsumgebungen in der Lehre?
    - Was sind Vorteile von integrierten Entwicklungsumgebungen in der Lehre?
    - Was sind Nachteile von integrierten Entwicklungsumgebungen in der Lehre?
    - Was sind Anforderungen an integrierte Entwicklungsumgebungen in der Lehre?
    - Lernziele
2. Selecting databases and other research sources
    - ACM Digital Library
    - DBLP Computer Science Bibliography
    - IEEE Xplore / IEEE-IET Electronic Library Online (IEL)
    - Web of Science
    - zbMATH Open
3. Defining search terms
4. Merging hits from different databases
5. Applying inclusion and exclusion criteria
6. Perform the review
7. Synthesizing results

# Kapitel 4

# Anforderungsanalyse

In diesem Kapitel sollen die Anforderung an die zu entwickelnde integrierte Entwicklungsumgebung dargelegt werden. Diese werden in Gesprächen mit Experten des GOLDi-Remotelab sowie Experten des CrossLab Projekts erhoben.

| | |
|---|---|
| **Anforderung Nr.:** | 1 |
| **Beschreibung:** | Die IDE soll komplett im Browser nutzbar sein. |
| **Begründung:** | Studierende haben oftmals Probleme mit der Installation von Software. |
| **Eignungskriterium:** | ? |
| **Kundenzufriedenheit:** | 5 |
| **Kundenunzufriedenheit:** | 5 |

| | |
|---|---|
| **Anforderung Nr.:** | 2 |
| **Beschreibung:** | Die IDE soll erweiterbar sein. |
| **Begründung:** | Durch die Erweiterbarkeit der IDE ... |
| **Eignungskriterium:** | ? |
| **Kundenzufriedenheit:** | 5 |
| **Kundenunzufriedenheit:** | 5 |

| | |
|---|---|
| **Anforderung Nr.:** | 3 |
| **Beschreibung:** | Die IDE soll kollaboratives Editieren von Dateien ermöglichen. |
| **Begründung:** | Teamwork ist ein wichtiges Lernziel. |
| **Eignungskriterium:** | ? |
| **Kundenzufriedenheit:** | 5 |
| **Kundenunzufriedenheit:** | 5 |

| | |
|---|---|
| **Anforderung Nr.:** | 4 |
| **Beschreibung:** | Die IDE soll das Kompilieren von C Programmen für Microcontroller ermöglichen. |
| **Begründung:** | ? |
| **Eignungskriterium:** | ? |
| **Kundenzufriedenheit:** | ? |
| **Kundenunzufriedenheit:** | ? |

| | |
|---|---|
| **Anforderung Nr.:** | 5 |
| **Beschreibung:** | Die IDE soll das Hochladen von kompilierten Programmen auf Microcontroller ermöglichen. |
| **Begründung:** | ? |
| **Eignungskriterium:** | ? |
| **Kundenzufriedenheit:** | ? |
| **Kundenunzufriedenheit:** | ? |

| | |
|---|---|
| **Anforderung Nr.:** | 6 |
| **Beschreibung:** | Die IDE soll das Debuggen von Programmen auf Microcontrollern ermöglichen. |
| **Begründung:** | ? |
| **Eignungskriterium:** | ? |
| **Kundenzufriedenheit:** | ? |
| **Kundenunzufriedenheit:** | ? |

| | |
|---|---|
| **Anforderung Nr.:** | 7 |
| **Beschreibung:** | Die IDE soll CrossLab-kompatibel sein. |
| **Begründung:** | ? |
| **Eignungskriterium:** | ? |
| **Kundenzufriedenheit:** | ? |
| **Kundenunzufriedenheit:** | ? |

| | |
|---|---|
| **Anforderung Nr.:** | 8 |
| **Beschreibung:** | Die IDE soll die Anbindung von Language Servern unterstützen. |
| **Begründung:** | ? |
| **Eignungskriterium:** | ? |
| **Kundenzufriedenheit:** | ? |
| **Kundenunzufriedenheit:** | ? |

| | |
|---|---|
| **Anforderung Nr.:** | 9 |
| **Beschreibung:** | Die IDE soll kostenlos nutzbar sein. |
| **Begründung:** | ? |
| **Eignungskriterium:** | ? |
| **Kundenzufriedenheit:** | ? |
| **Kundenunzufriedenheit:** | ? |

| | |
|---|---|
| **Anforderung Nr.:** | 10 |
| **Beschreibung:** | Die IDE soll auch außerhalb eines Experiments nutzbar sein. |
| **Begründung:** | ? |
| **Eignungskriterium:** | ? |
| **Kundenzufriedenheit:** | ? |
| **Kundenunzufriedenheit:** | ? |

| | |
|---|---|
| **Anforderung Nr.:** | 11 |
| **Beschreibung:** | Das erlernte Wissen im Umgang mit der IDE soll auch außerhalb des GOLDi Remotelab anwendbar sein. |
| **Begründung:** | ? |
| **Eignungskriterium:** | ? |
| **Kundenzufriedenheit:** | ? |
| **Kundenunzufriedenheit:** | ? |

| | |
|---|---|
| **Anforderung Nr.:** | 12 |
| **Beschreibung:** | Debug-Sessions sollten zwischen Nutzern teilbar sein. |
| **Begründung:** | ? |
| **Eignungskriterium:** | ? |
| **Kundenzufriedenheit:** | ? |
| **Kundenunzufriedenheit:** | ? |

| | |
|---|---|
| **Anforderung Nr.:** | 13 |
| **Beschreibung:** | Die IDE sollte in Lehrplattformen einbindbar sein (z.B. über LTI) |
| **Begründung:** | ? |
| **Eignungskriterium:** | ? |
| **Kundenzufriedenheit:** | ? |
| **Kundenunzufriedenheit:** | ? |

# Kapitel 5

# Stand der Technik

- Standalone IDEs
  - VSCode
  - Brackets
  - Phoenix
  - IntelliJ IDEs
- IDE Frameworks
  - Theia
  - OpenSumi
- Remote Development Platforms
  - Eclipse Che
  - Coder
  - Gitpod
  - Replit
  - Codeanywhere
  - Stackblitz
- Educational
  - Codeboard
  - Coding Rooms
  - CodeHS
  - JDoodle

  - KIRA
  - Codio
  - Online IDE
  - Codegrade

Im Bereich der IDEs gibt es bereits viele bestehende Softwarelösungen. Für die Zwecke dieser Arbeit ergeben sich drei Kategorien:

- **Standalone IDEs / IDE-Frameworks:**
  Diese Kategorie beinhaltet Softwarelösungen wie VSCode, Theia und OpenSumi.
- **Educational IDE Plattformen:**
  Diese Kategorie beinhaltet Softwarelösungen wie CodeHS, Codegrade und KIRA.
- **Workspace Management:**
  Diese Kategorie beinhaltet Softwarelösungen wie Eclipse Che, Coder und Gitpod.

Im folgenden werden die einzelnen Kategorien genauer betrachtet.

## 5.1 Standalone IDEs / IDE-Frameworks

In der Kategorie Standalone IDEs / IDE-Frameworks befinden sich unter anderem Visual Studio Code, Theia und OpenSumi.

## 5.2 Educational IDE Plattormen

## 5.3 Workspace Management

# Kapitel 6

# Konzeption

Im Fokus dieser Arbeit liegt die Programmierung von Mikrocontrollern im Rahmen des GOLDi Remotelab. Bei den verwendeten Microcontrollern handelt es sich um ATmega2560-16AU. Damit diese mit der CrossLab Infrastruktur kommunizieren können sind sie über einen FPGA mit einem Raspberry Pi Compute Module 4 verbunden. Dabei übernimmt der FPGA die Kommunikation zwischen dem CM und dem Microcontroller, während das CM die Kommunikationsschnittstelle zur CrossLab Infrastruktur übernimmt.

## 6.1 Debugging

Das Debuggen von Programmen auf den vorhandenen Microcontrollern gestaltet sich schwierig. Eine Möglichkeit ist die Nutzung der Bibliothek avr_debug. Diese wird zusammen mit dem Programm kompiliert und auf den Microcontroller hochgeladen. Dort erstellt sie ein Interface für den Debugger gdb. Dieses Interface nutzt die Serielle Schnittstelle des Microcontrollers zur Kommunikation mit gdb. Das CM agiert in diesem Szenario als Schnittstelle zwischen gdb und unserer IDE. Ein Nachteil dieses Vorgehens ist der hohe Speicherverbrauch der Bibliothek, welcher die Anzahl möglicher Programme einschränkt. Allerdings ist ein Vorteil dieses Ansatzes, dass keine zusätzlichen Kosten durch die Anschaffung externe Debugger entstehen.
Ein weiteres Problem, was beim Debuggen eines laufenden Experimentes beachtet werden muss, ist die fortlaufende Ansteuerung von weiteren Geräten. Nehmen wir als Beispiel ein einfaches Experiment bestehend aus einem Microcontroller und einem 3-

Achs-Portal. Wenn wir das Program des Microcontrollers unterbrechen, während dieser den Portalkran aktiv nach rechts bewegt, so wird diese Bewegung nicht unterbrochen. Um sicherzustellen, dass die Signale von Aktoren während eines Breakpoints nicht an andere Geräte weitergeleitet werden müssen die anderen Geräte entsprechend benachrichtigt werden.

## 6.2 Language Server

Eine Möglichkeit zur Einbindung von Language Servern ist es, diese auf dem CM zu installieren und sich dann zu diesen zu verbinden. Allerdings ist das Language Server Protokoll nur auf einen einzigen Client pro Server ausgelegt. Daher müsste in einem Experiment mit mehreren Nutzern für jeden dieser Nutzer eine eigene Instanz des Language Server erstellt werden. Eine spezielle Alternative könnte die Verwendung des zu WebAssembly kompilerten Language Servers clangd darstellen. Dieser könnte im Browser des jeweiligen Nutzers instanziiert werden, wodurch keine zusätzliche Last auf dem CM entsteht. Allerdings muss hierbei die neu entstandene Last auf dem Rechner des Nutzers in Betracht gezogen werden. Eine weitere Möglichkeit ist die Bereitstellung weiterer Server zur Bereitstellung der Language Server.

## 6.3 Kompilierung

Die Kompilierung des Quellcodes kann vom CM übernommen werden. Alternativ steht auch hier die Möglichkeit eines externen Servers zur Verfügung. Weiterhin kann auch die Möglichkeit der Kompilierung innerhalb des Browsers des Nutzers erforscht werden. Hierbei gibt es bereits eine Version des C-Compilers clang in WebAssembly. Diese ermöglicht jedoch in der aktuellen Form nur die Kompilierung von C/C++ Code zu WebAssembly. Im Falle des GOLDi Remotelab müsste allerdings eine Kompilierung des Quellcodes für die verwendeten Microcontroller erfolgen. Diese wird normalerweise mit Hilfe des Compilers avr-gcc durchgeführt. Allerdings bietet auch der clang Compiler ein avr Ziel an, welches jedoch einen experimentellen Status hat.

## 6.4 Kollaboratives Zusammenarbeiten

Die Studenten sollen in der Lage sein gemeinsam an Dateien arbeiten zu können. Hierbei könnte auf bereits bestehende Lösungen zurückgegriffen werden. Allerdings muss hierbei beachtet werden, dass Studenten auch in der Lage sein sollten Debug-Sessions zu teilen. Die Extension Live Share von Microsoft ermöglicht das kollaborative Editieren von Dateien und das Teilen von Debug-Sessions. Allerdings wird für die Nutzung des Services ein Microsoft bzw. ein Github Account benötigt. Es wäre vom Vorteil eine Lösung zu finden, die es den Studenten ermöglicht zusammen an einem Projekt zu arbeiten, ohne dafür einen neuen Account anlegen zu müssen. Für eine derartige Implementierung können bereits vorhandene Software-Bibliotheken wie z.B. YJS genutzt werden.

## 6.5 Datenspeicherung

Um sicherzustellen, dass die IDE komplett im Browser verwendet werden kann, müssen auch die Dateien der Nutzer im Browser gespeichert werden können. Dafür kann die IndexedDB verwendet werden. Diese ermöglicht die persistente Speicherung von größeren Datenmengen im Browser. Um die Speicherung der Daten in der IndexedDB zu ermöglichen muss eine entsprechende Erweiterung entwickelt werden. Für derartige Erweiterungen steht die FileSystemProvider API zur Verfügung. Um allerdings auch die Durchsuchbarkeit der Dateien sowie deren Inhalts zu ermöglichen müssen die FileSearchProvider und TextSearchProvider APIs eingesetzt werden. Diese befinden sich allerdings noch in einem proposed Stadium. Dies bedeutet, dass sie sich noch verändern können. Erweiterung, die derartige APIs nutzen können nicht auf dem Marketplace angeboten werden. Jedoch sollte dies für die Zwecke des GOLDi Remotelab kein Problem darstellen.

## 6.6 CrossLab Kompatibilität

Zur Herstellung der CrossLab Kompatibilität müssen zunächst die benötigten Schnittstellen definiert werden. Darauf aufbauend können dann entsprechende Services definiert werden. Hierbei gilt es auch zu beachten, dass diese Schnittstellen möglichst wiederver-

wendbar gestaltet sind.

# Kapitel 7

# Implementierung

## 7.1 Datenspeicherung

- FileSystemProvider
- FileSearchProvider (Proposed API)
- TextSearchProvider (Proposed API)
- Speicherung in indexeddb an Domain gebunden →

## 7.2 Code Kompilierung

- Kompilierungsserver
- Kompilierung auf Raspberry Pi

## 7.3 Language Server

- Server wahrscheinlich benötigt
- clangd WebAssembly? (https://github.com/Guyutongxue/clangd-in-browser)
- Handhabung kollaboratives Arbeiten an Code

## 7.4 Debugging

- avr_debug (https://github.com/jdolinay/avr_debug)
- Anbindung des Debuggers an das Frontend

## 7.5 Kollaboratives Bearbeiten

- Yjs kann genutzt werden (https://github.com/yjs/yjs)
- wahrscheinlich Anpassung des webrtc-connectors erforderlich (https://github.com/yjs/y-webrtc)

## 7.6 CrossLab Kompatibilität

- Definition und Implementierung der Services
- Anbindung an IDE

# Kapitel 8

# Auswertung

# Kapitel 9

# Zusammenfassung und Fazit

# Abbildungsverzeichnis

# Tabellenverzeichnis

# Liste der Algorithmen

[Badrinarayanan et al., 2015]

# Literaturverzeichnis

[Badrinarayanan et al., 2015] Badrinarayanan, Vijay, A. Kendall und R. Cipolla (2015). *SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation*. arXiv preprint arXiv:1511.00561. *(Seite: 29)*