

Hochschule Karlsruhe  
Technik und Wirtschaft  
UNIVERSITY OF APPLIED SCIENCES

Fakultät Elektro- und Informationstechnik  
Studiengang Elektrotechnik – Elektro- und Informationstechnik

# Masterthesis

VON

David Erb

<b>Referent:</b>	Prof. Dr. Marianne Katz
<b>Korreferent:</b>	Prof. Dr. rer. nat. Klaus Wolfrum
<b>Arbeitsplatz:</b>	
<b>Betreuer am Arbeitsplatz:</b>	
<b>Zeitraum:</b>	18.04.2017 – 18.10.2017

# Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Masterthesis ohne unzulässige fremde Hilfe selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben.

*Stuttgart, den 1. Juni 2017*

# Inhaltsverzeichnis

<b>Inhaltsverzeichnis</b>	<b>II</b>
<b>1 Einleitung</b>	<b>1</b>
1.1 Motivaton . . . . .	1
1.2 Aufgabenstellung . . . . .	1
<b>2 Grundlagen</b>	<b>2</b>
2.1 GIGABOX . . . . .	2
2.1.1 Überblick . . . . .	2
2.1.2 Skriptsprache PAWN . . . . .	4
2.1.3 configurAIDER . . . . .	5
2.2 Softwareentwicklungsprozess . . . . .	5
2.2.1 V-Modell . . . . .	5
2.2.2 Qualitätskriterien an Software . . . . .	5
2.3 Entwicklungsframework . . . . .	6
2.3.1 .NET-Framework . . . . .	6
2.3.2 Windows Presentation Framework(WPF) . . . . .	6
<b>3 Anforderungsanalyse</b>	<b>7</b>
3.1 Übersicht . . . . .	7
3.2 Ermittlung und Bewertung von funktionalen Anforderungen . .	8
3.2.1 Erstellung von Projekten mit Baumstruktur . . . . .	8

3.2.2	Einstellung der Sprache . . . . .	8
3.2.3	Texteditor zur Erstellung von PAWN-Skripten . . . . .	9
3.2.4	Konfigurieren einer GIGABOX ohne PAWN-Kenntnisse .	11
3.2.5	Verbindung zu einer GIGABOX herstellen . . . . .	12
3.2.6	PAWN-Compiler . . . . .	12
3.2.7	Beschreiben/Auslesen des Flash-Speichers der GIGABOX	12
3.2.8	Kommunikation mit einer GIGABOX . . . . .	13
3.2.9	Kommunikation mit dem Benutzer . . . . .	13
3.2.10	Steuern und Beobachten von Ein-/Ausgängen und Timern	13
3.3	Ermittlung von nichtfunktionalen Anforderungen . . . . .	16
3.4	Priorisierung der Anforderungen . . . . .	18
3.4.1	Funktionale Anforderungen . . . . .	19
<b>4</b>	<b>Ist-Zustand</b>	<b>20</b>
4.1	Überblick . . . . .	20
4.2	Elemente des configurAIDER . . . . .	20
4.2.1	Rahmenfenster . . . . .	20
4.2.2	Fenster „Verfügbare Geräte“ . . . . .	20
4.2.3	Fenster „Konsole“ . . . . .	22
4.2.4	Fenster „Programmlog“ . . . . .	23
4.2.5	Fenster „scriptEDITOR“ . . . . .	24
4.2.6	Fenster „multiEDITOR“ . . . . .	25
4.3	Funktionalität . . . . .	27
4.3.1	Einstellung der Sprache . . . . .	30
4.3.2	Bereitstellung von Dokumentation zur GIGABOX, con- figurAIDER und PAWN . . . . .	30
4.3.3	Verbindung zu einer GIGABOX herstellen . . . . .	30
4.3.4	PAWN-Code kompilieren . . . . .	31
4.3.5	PAWN-Code entwickeln . . . . .	31

4.3.6	Übertragen von Skript-Applikationen . . . . .	32
4.3.7	Kommunikation mit verbundener GIGABOX . . . . .	33
4.3.8	Ausgabe von Ereignismeldungen . . . . .	33
4.3.9	Ergebnis . . . . .	33
<b>5</b>	<b>Evaluation</b>	<b>34</b>
5.1	Funktionalität und Usability des Ist-Zustands . . . . .	34
5.1.1	Einstellung der Sprache . . . . .	34
5.1.2	Bereitstellung von Dokumentation zur GIGABOX, configurAIDER und PAWN . . . . .	34
5.1.3	Verbindung zu einer GIGABOX herstellen . . . . .	35
5.1.4	PAWN-Code kompilieren . . . . .	35
5.1.5	PAWN-Code entwickeln . . . . .	35
5.1.6	Übertragen von Skript-Applikationen . . . . .	36
5.1.7	Kommunikation mit verbundener GIGABOX . . . . .	37
5.1.8	Ausgabe von Ereignismeldungen des configurAIDERS . . . . .	37
5.2	Differenz zwischen Anforderungen und Ist-Zustand . . . . .	37
5.3	Quellcode des configurAIDERS . . . . .	41
5.4	Ergebnis . . . . .	41
<b>6</b>	<b>Konzeption</b>	<b>42</b>
6.1	Übersicht . . . . .	42
6.2	Erweiterung des bestehenden configurAIDERS um neue Features . . . . .	42
6.3	Neuentwicklung des configurAIDERS . . . . .	43
<b>7</b>	<b>Design</b>	<b>45</b>
7.1	Entwurf der Benutzeroberfläche . . . . .	45
7.2	Entwurf der Softwarearchitektur . . . . .	45
	<b>Abbildungsverzeichnis</b>	<b>47</b>

<b>Tabellenverzeichnis</b>	<b>48</b>
<b>Literaturverzeichnis</b>	<b>49</b>

# Kapitel 1

## Einleitung

### 1.1 Motivaton

configurAIDER wurde im Rahmen von verschiedenen studentischen Arbeiten erstellt. Aktuell ergeben sich folgende Probleme:

- nicht testbar, da das Sollverhalten nicht immer bekannt ist
- nicht wartbar, da sich keiner im Code auskennt und keine saubere Dokumentation vorhanden ist
- schwer erweiterbar, da sich keiner im Code auskennt und keine saubere Dokumentation vorhanden ist

-> Kommt erst später rein

Einführung in das Thema...

### 1.2 Aufgabenstellung

# Kapitel 2

## Grundlagen

### 2.1 GIGABOX

#### 2.1.1 Überblick

GIGABOX ist eine Produktfamilie von Steuergeräten der Firma GIGATRONIK. GIGABOX-Steuergeräte werden hauptsächlich in der Fahrzeugentwicklung eingesetzt, um prototypische Kommunikationslösungen im Fahrzeug schnell realisieren zu können. Beispielhafte Anwendungen sind der Einsatz als Gateway zur Übersetzung zwischen unterschiedlichen Busprotokollen und zur Ansteuerung von Treiber-ICs im Automobil. Das Use-Case-Diagramm in Abbildung XX zeigt die Einsatzmöglichkeiten von GIGABOXEN.

GIGABOX-Steuergeräte werden in unterschiedlicher Hardwareausstattung und Funktionalität angeboten. Tabelle XX bietet eine Übersicht über die verschiedenen GIGABOX Varianten.

**TODO: Welche GIGABOX-Varianten gibt es? Auf Homepage sind nur wenige gelistet. Warum?** In Confluence: S,FR Degraded, FR Extended, FR Gateway, FR Gateway BLE, XL Standard -> Vermutung: Nur die auf der Homepage angebotenen werden noch verkauft: BEO, flex-i-FlexRay USB Interface, FlexRay Active Star, Gateway, XL. **Warum heißt sense ironCUBE auch GIGABOX? -> ist kein Steuergerät**

Eine GIGABOX besteht aus einer Basisplatine und einer Applikationsplatine. Das GIGABOX gate Grundmodul verfügt über Stromversorgung, USB-Anschluss und einen leistungsfähigen Mikrocontroller mit einer Vielzahl von Standard-Schnittstellen (FlexRay, CAN, LIN) sowie der dazugehörigen Treiber-Software.



TODO: Wenn auf Basisplatine LIN,CAN,Flexray ist, warum haben alle GIGABOXEN diese Bussysteme? Sind grundsätzlich alle Module für LIN,CAN,Flexray vorhanden, aber je nach Applikationsplatine werden nur einzelne Module zur Verfügung gestellt?

Beschreibung Funktionsweise GIGABOX aus Softwaresicht. Skript-Applikation und Bootapplikation. Virtuelle Maschine läuft in Skript-Applikation. Auf virtueller Maschine wird Bytecode ausgeführt. Der Bytecode wurde vom PAWN-Compiler (in configurAIDER) aus einer Skriptdatei erzeugt. Quelle: Bachelorarbeit Christian Eissler Absch. 4.1 (Arbeit über GIGABOX FD -> Funktionsweise bei den alten GIGABOXEN gleich?)

Prozessor Flash-Speicher

Der integrierte Mikrocontroller bietet eine UART-Schnittstelle. Mithilfe eines USB-to-Serial Port Adapters (Korrekt?) kann die GIGABOX per Kabel mit der USB-Schnittstelle eines PCs verbunden werden, um z.B. neu zu flashen.

Abbildung 2.1 zeigt typische Einsatzszenarien der GIGABOX.

1. Use Case: Einsatz als Gateway zur Modifikation von Busbotschaften

Anwendungsbeispiel: Es soll getestet werden, ob Klimaanlage in neuer Fahrzeugserie funktioniert. Busbotschaften in der neuen Serie wurden modifiziert im Vergleich zur alten Serie. Fahrzeug aus neuer Serie steht aber noch nicht zur Verfügung. Deshalb werden Bussignale aus alter Serie mit der GIGABOX so verändert, dass Sie den Signalen der neuen Serie entsprechen.

2. Use Case: Funktionalität des Fahrzeugs erweitern

- Rapid Prototyping: Neues Feature soll möglichst schnell und unkompliziert getestet werden  
Bsp: Blinker-LEDs sollen nicht mehr gleichzeitig angesteuert werden, sondern nacheinander von innen nach außen mit kurzer zeitlicher Verzögerung
- Serieneinsatz: Neue Funktion bei Fahrzeugumbau realisieren  
Bsp: Mercedes E-Klasse wird zu einem Leichenwagen umgebaut. Neue Funktionen wie Rollläden hoch- und runterfahren sollen realisiert werden.

3. Use Case: Restbussimulation:

Nachrichten von nicht real im Netzwerk vorhandenen Steuergeräten werden nachgebildet (von der GIGABOX). Steuergeräte können getestet werden, ohne

dass der komplette Bus aufgebaut werden muss. Quelle: <http://www.samtec.de/hauptmenu/loesfuer/restbussimulation.html>

Kann auch mit Tools wie CANoe gemacht werden. Für CANoe wird aber teure Lizenz benötigt und ein Laptop. GIGABOX ist handlicher und billiger.

Unterschied zu HiL?

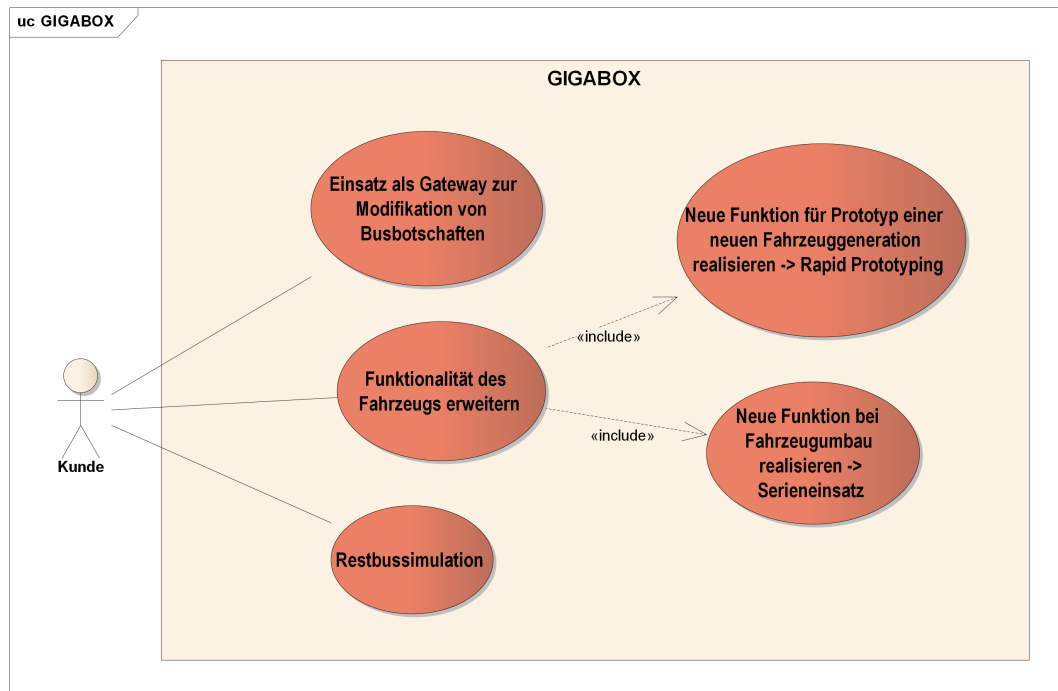


Abbildung 2.1 – Use-Case-Diagramm GIGABOX

## 2.1.2 Skriptsprache PAWN

Siehe Pawn\_Language\_Guide.pdf, Foreword

Pawn\_Implementation\_Guide.pdf, The Compiler, The abstract machine

Warum wurde PAWN gewählt? -frei verfügbar

Vorteile PAWN auf VM gegenüber C? - einfacher zu programmieren -durch VM ist Code hardwareunabhängig und muss nicht an eingesetzten Mikrocontroller angepasst werden -> VM muss aber an Hardware angepasst werden

### 2.1.3 configurAIDER

Der configurAIDER ist eine von GIGATRONIK entwickelte integrierte Entwicklungsumgebung (IDE), die es Benutzern ermöglicht, mit einer GIGABOX zu kommunizieren und diese zu programmieren. Es können Diagnoseinformationen abgerufen werden und Bytecode auf das Steuergerät geflasht werden. Die IDE besteht aus Editoren (skriptEDITOR und multiEDITOR), Interpreter und Linker. Der skriptEDITOR ist ein Texteditor, mit dem PAWN-Skripte geschrieben werden können. Mit dem multiEDITOR können tabellarisch WENN DANN Anweisungen konfiguriert werden, aus denen ein Skript mit PAWN-Code generiert werden kann. Der multiEDITOR stellt für Benutzer ohne Programmierkenntnisse eine Möglichkeit dar, die GIGABOX mit rudimentären Funktionen zu konfigurieren. Der integrierte Interpreter kann Bytecode aus den erstellten PAWN-Skripten erstellen und auf die GIGABOXEN flashen.

## 2.2 Softwareentwicklungsprozess

### 2.2.1 V-Modell

### 2.2.2 Qualitätskriterien an Software

Qualitätskriterien an Software nach ISO 9126 (aus Wiki) besser: SQuaRE-Standard, Product Quality Model (2 Merkmale mehr als ISO 9126) Übernahme S.87 Software Requirements

- Wartbarkeit
  - Analysierbarkeit:
  - Modifizierbarkeit:
  - Testbarkeit:
- Benutzbarkeit (genauer definiert in Norm EN ISO 9241-110)
  - Aufgabenangemessenheit: „Ein Dialog ist aufgabenangemessen, wenn er den Benutzer unterstützt, seine Arbeitsaufgabe effektiv und effizient zu erledigen.“
  - Selbstbeschreibungsfähigkeit: „Ein Dialog ist in dem Maße selbstbeschreibungsfähig, in dem für den Benutzer zu jeder Zeit offensichtlich ist, in welchem Dialog, an welcher Stelle er sich befindet, welche

Handlungen unternommen werden können und wie diese ausgeführt werden können.“

- Lernförderlichkeit
- Steuerbarkeit
- Erwartungskonformität: „Ein Dialog ist erwartungskonform, wenn er konsistent ist und den Merkmalen des Benutzers entspricht, z.B. seinen Kenntnissen aus dem Arbeitsgebiet, seiner Ausbildung und seiner Erfahrung sowie den allgemein anerkannten Konventionen.“
- Individualisierbarkeit
- Fehlertolerant: „Ein Dialog ist fehlertolerant, wenn das beabsichtigte Arbeitsergebnis trotz erkennbar fehlerhafter Eingaben entweder mit keinem oder mit minimalem Korrekturaufwand durch den Benutzer erreicht werden kann.“

Quelle: Wikipedia -> überprüfen ob andere Quelle verfügbar ist, am Besten die Norm direkt

- Effizienz
- Übertragbarkeit
  - Anpassbarkeit: Möglichkeit, Software an verschiedene Umgebungen anzupassen  
Verschiedene Betriebssysteme? 32/64Bit?
- Zuverlässigkeit
  - Reife: Geringe Versagenshäufigkeit durch Fehlerzustände
  - Wiederherstellbarkeit: Fähigkeit, bei einem Versagen das Leistungsniveau wiederherzustellen und die direkt betroffenen Daten wiederzugewinnen. Zu berücksichtigen sind die dafür benötigte Zeit und der benötigte Aufwand.

## 2.3 Entwicklungsframework

### 2.3.1 .NET-Framework

### 2.3.2 Windows Presentation Framework(WPF)

MVVM-Pattern

# Kapitel 3

## Anforderungsanalyse

### 3.1 Übersicht

In diesem Kapitel soll eine Anforderungsanalyse für eine Entwicklungsumgebung für GIGABOX-Steuergeräte durchgeführt werden. In Abbildung 2.1 wurde gezeigt, für welche Anwendungen GIGABOXEN eingesetzt werden. Die Anforderungen an die Entwicklungsumgebung wurden so gestellt, dass sie Softwareentwicklern für GIGABOX-Funktionen die Umsetzung dieser Anwendungen ermöglicht und möglichst große Unterstützung bei der Codeentwicklung bietet.

Zur Ermittlung der funktionalen Anforderungen wurden Gespräche mit Entwicklern geführt, vorhandener Programmcode aus realisierten Projekten analysiert und selbstständig Codeentwicklung mit dem configurAIDER betrieben. Aus diesen Eindrücken heraus entstanden die im Folgenden beschriebenen funktionalen Anforderungen.

Jede funktionale Anforderung soll bewertet werden nach der Priorität der Umsetzung. Die Bewertung der Anforderungen soll qualitativ erfolgen anhand von Meinungen befragter Entwickler.

Auf Basis der Qualitätsmerkmale für die Anforderungsspezifikation im Standard IEEE 830-1998 werden die funktionalen Anforderungen eingeteilt in drei Prioritätsklassen.

- Essential: Das Software-System kann nicht akzeptiert werden, wenn diese Anforderung nicht in der vereinbarten Form geliefert wird.
- Conditional: Diese Anforderungen erweitern das Software-System in geeigneter Form. Wenn sie fehlen, würden sie den Einsatz des Systems jedoch nicht gefährden.

- Optional: Diese Anforderungen sind nicht unbedingt notwendig, für die Anwender jedoch eine angenehme Erweiterung (nice to have)

Zur Ermittlung der nichtfunktionalen Anforderungen wurden bekannte Vorgehensmodelle und Normen der Softwareentwicklung herangezogen. Die hergeleiteten nichtfunktionalen Anforderungen sollen bei der Entwicklung beachtet werden, um eine hohe Qualität der zu entwickelnden Software sicherzustellen. Auf eine Bewertung wird verzichtet.

Quelle: [1]

## 3.2 Ermittlung und Bewertung von funktionalen Anforderungen

### 3.2.1 Erstellung von Projekten mit Baumstruktur

*Beschreibung:*

Innerhalb eines Projektes können verschiedene GIGABOX-Modelle angelegt werden. Für jedes angelegte GIGABOX-Modell können PAWN-Skripte erstellt werden mit der zum Modell passenden Dateiendung `.gt**.p`. Für jedes angelegte GIGABOX-Modell können Include-Files hinzugefügt werden

*Bewertung:*

Der Benutzer erhält einen strukturierten Überblick über alle seine GIGABOX-Projekte und alle für ein Projekt entwickelten Skripte. Das Öffnen eines Skriptes erfolgt komfortabel über Doppelklick auf ein Skript in der Projektstruktur. Lästiges navigieren durch die Dateistruktur des PCs beim Öffnen eines Skriptes entfällt + der Benutzer muss den Ablageort im Dateisystem nicht wissen. Erhöht den Benutzerkomfort signifikant, deshalb Einstufung in Prioritätsklasse „Conditional“.

### 3.2.2 Einstellung der Sprache

*Beschreibung:*

Die Sprache der Benutzeroberfläche kann angepasst werden. Es soll Deutsch und Englisch zur Verfügung stehen.

*Bewertung:*

Die Darstellung der Benutzeroberfläche in der Muttersprache des Benutzers erhöht die Usability, da es die Selbstbeschreibungsfähigkeit und Individualisierbarkeit fördert. Englisch muss auf jeden Fall zur Verfügung stehen, da es die international am weitesten verbreitete Sprache ist und auch in Deutschland von einem Großteil der Menschen verstanden wird. Die Darstellung der Benutzeroberfläche in Englisch wird deshalb in Prioritätsklasse „Essential“ eingeteilt. Die Darstellung in Deutsch wird in Klasse „Optional“ eingeteilt, da die Usability dadurch nur mäßig erhöht wird. Das liegt daran, dass der potenzielle Benutzerkreis des Tools Entwickler sind, denen größtenteils der Umgang mit englischen GUIs vertraut ist.

### 3.2.3 Texteditor zur Erstellung von PAWN-Skripten

- Schreiben von PAWN-Code

*Beschreibung:*

Es steht ein Textfeld zur Verfügung, in dem PAWN-Code editiert werden kann.

*Bewertung:*

Klasse „Essential“, da als Grundfunktionalität einer IDE einzustufen.

- Copy/Paste

*Beschreibung:*

Codefragmente können kopiert und an anderer Stelle eingefügt werden.

*Bewertung:*

Macht die Codeentwicklung bedeutend effektiver, da dem Benutzer Tipparbeit erspart bleibt. Weiterer Vorteil ist die Fehlervermeidung durch Tippfehler. Deshalb Einstufung in Klasse „Conditional“.

- Suchen/Ersetzen

*Beschreibung:*

Ein Skript kann nach Zeichenketten durchsucht werden. Die gefundenen Ergebnisse werden farblich hervorgehoben und es kann zu den Ergebnissen gesprungen werden. Alle gefundenen Ergebnisse können ersetzt werden durch eine neue Zeichenkette.

*Bewertung:*

Das Auffinden von Variablen und Methoden im Skript wird erheblich erleichtert. Bei Umbenennung von Variablen oder Methoden wird viel

Zeit eingespart und es werden Tippfehler vermieden. Einteilung in Klasse „Conditional“.

- Letzte Schritte rückgängig machen

*Beschreibung:*

Die letzten vom Benutzer vorgenommenen Anweisungen können rückgängig gemacht werden.

*Bewertung:*

Fördert die Fehlertoleranz. Einteilung in Klasse „Conditional“

- Anzeige aller instanziierten Variablen im Skript

*Beschreibung:*

Alle instanziierten Variablen werden aufgelistet. Doppelklick auf einen Variablennamen in der Liste markiert im Skript alle Verwendungen der Variable. Per Drag&Drop kann ein Variablenname aus der Liste in das Skript eingefügt werden.

*Bewertung:*

Benutzer hat alle instanziierten Variablen im Blick und kann leicht zu Verwendungsstellen im Skript navigieren. Das Einfügen per Drag&Drop bietet eine komfortable Möglichkeit der Variablenverwendung und verhindert Fehleingaben durch Tippfehler. Einteilung in Klasse „Optional“.

- Anzeige aller implementierten Funktionen zur Navigation im Skript

*Beschreibung:*

Alle implementierten Funktionen werden gelistet. Doppelklick auf eine Funktion führt zu einem Sprung zur Funktionsimplementierung. Per Drag&Drop kann ein Funktionsname in das Skript gezogen werden, ein Aufruf der Funktion wird in das Skript eingefügt. Es können alle Stellen im Skript markiert werden, an denen die gewählte Funktion aufgerufen wird.

*Bewertung:*

Benutzer hat alle implementierten Funktionen im Blick und kann leicht zu Verwendungsstellen im Skript navigieren. Komfortable Möglichkeit der Funktionsverwendung, Verhindert Fehleingaben durch Tippfehler. Einteilung in Klasse „Optional“.

- Bei der Codeentwicklung unterstützende Funktionen

*Beschreibung:*



Es sollen Funktionen integriert werden, die schnelleres und effektiveres Schreiben von Code ermöglichen. Bsp: Autovervollständigung von Variablen, Funktionen und Anweisungen bei der Tastatureingabe

*Bewertung:*

Einteilung in Klasse „Optional“.

- Routingeditor

*Beschreibung:*

**Beschreibung für CAN** Aus einer Datenbankdatei (.dbc/.arxml) können dort definierte Bussignale geladen werden. Die Signale können in verschiedenen CAN-Botschaften enthalten sein. Aus den Bussignalen kann im Routingeditor eine eigene CAN-Busbotschaft konfiguriert werden mit einer vom Benutzer definierten ID. Dazu können die Signale innerhalb der Nutzdatenfeldes einer PDU frei angeordnet werden.

Aus der im Routing-Editor konfigurierten Botschaft kann PAWN-Code generiert werden und in die Funktion OnCanRxEvent() eines geöffneten Skriptes im Texteditor eingefügt werden.

*Bewertung:* Beim Programmieren der GIGABOX tritt des öfteren der Use-Case auf, dass aus Signalen, die in verschiedenen auf dem Bus ankommenden Botschaften enthalten sind, eine neue Botschaft erstellt und verschickt werden soll.

Dabei werden oft Bitverschiebungen benötigt, deren Programmierung viel Zeit- und Denkaufwand bedeuten. Der Routingeditor ermöglicht die grafische Konfiguration von Busbotschaften und erleichtern die Umsetzung dieser Routingaufgaben. Einteilung in Klasse „Optional“.

### 3.2.4 Konfigurieren einer GIGABOX ohne PAWN-Kenntnisse

*Beschreibung:* Grundlegende Funktionen einer GIGABOX sollen ohne Kenntnisse von PAWN konfiguriert werden können. Dazu bietet sich die Funktionsbausteinsprache an.

*Bewertung:*

Erleichtert es Kunden ohne Programmierkenntnisse, selbständig GIGABOX-Funktionalitäten zu konfigurieren. Erweitert möglicherweise den Kreis an potentiellen Käufern der GIGABOX. Im Vergleich zur direkten Codeentwicklung können allerdings nur eingeschränkte Funktionalitäten realisiert werden. Erfahrene Softwareentwickler werden aufgrund dieser Tatsache PAWN-Code direkt entwickeln. Einteilung in Klasse „Optional“.

### 3.2.5 Verbindung zu einer GIGABOX herstellen

#### *Beschreibung:*

Alle GIGABOXEN, die mit dem PC per Kabel oder Bluetooth verbunden sind, werden aufgelistet. Es werden Geräteinformationen wie z.B. Modellname, Seriennummer bereitgestellt. Der Benutzer kann aus der Liste eine GIGABOX anwählen, mit der kommuniziert werden soll.

#### *Bewertung:*

Wird als Grundfunktionalität eingestuft, Klasse „Essential“.

### 3.2.6 PAWN-Compiler

#### *Beschreibung:*

PAWN-Skripte können in Bytecode übersetzt werden. Wenn Include-Files verwendet werden, soll ein Linker den Bytecode des Skriptes und des Include-Files nach dem Übersetzen zusammenfügen.

#### *Bewertung:*

Das Kompilieren ist eine Grundfunktionalität und wird deshalb in die Klasse „Essential“ eingestuft.

### 3.2.7 Beschreiben/Auslesen des Flash-Speichers der GIGABOX

#### *Beschreibung:*

Skript-Applikationen können vom PC auf den Flash-Speicher einer GIGABOX übertragen werden. Sich bereits auf dem Flash-Speicher befindliche Skripte können auf den PC geladen und angezeigt werden. Die Übertragung soll jeweils per Kabel über die USB-Schnittstelle und per Bluetooth möglich sein.

#### *Bewertung:*

Die Übertragung von Skripten vom PC auf eine GIGABOX über die USB-Schnittstelle wird als Grundfunktionalität bewertet und in Klasse „Essential“ eingeteilt. Die Übertragung von der GIGABOX auf den PC über die USB-Schnittstelle ist für den Entwickler sehr nützlich, aber nicht unbedingt notwendig und wird deshalb als Klasse „Conditional“ eingestuft. Bluetooth-Übertragung wird in Klasse „Optional“ eingestuft, da es wenige Use-Cases gibt, bei denen

eine Funkübertragung Vorteile gegenüber der Kabelübertragung bietet. Ein möglicher Use-Case wäre eine schwer zugängliche GIGABOX im Fahrzeug, auf die ein neues Skript aufgespielt werden soll. Eine Übertragung per Bluetooth würde einen aufwändigen Ausbau der GIGABOX aus dem Fahrzeug ersparen.

### 3.2.8 Kommunikation mit einer GIGABOX

#### *Beschreibung:*

Benutzer soll Befehle an eine GIGABOX senden können, z. B. einen Reset der GIGABOX veranlassen, den Bootloader aufrufen. Außerdem sollen Diagnose- und Geräteinformationen abgerufen werden können. Die Datenübertragung soll per Kabel über die USB-Schnittstelle und per Bluetooth möglich sein.

#### *Bewertung:*

Das Senden von Befehlen und Abrufen von Diagnose- und Geräteinformationen per Kabel über die USB-Schnittstelle wird in Klasse „Essential“ eingestuft. Die Datenübertragung per Bluetooth wird in Klasse „Optional“ eingestuft, da es wenige Use-Cases gibt, bei denen eine Funkübertragung Vorteile gegenüber der Kabelübertragung bietet. Ein möglicher Use-Case wäre eine schwer zugängliche GIGABOX im Fahrzeug, auf die ein neues Skript aufgespielt werden soll. Eine Übertragung per Bluetooth würde einen aufwändigen Ausbau der GIGABOX aus dem Fahrzeug ersparen.

### 3.2.9 Kommunikation mit dem Benutzer

#### *Beschreibung:*

Benutzer soll Befehle an configurAIDER senden können, z. B. das Kompilieren eines Skriptes veranlassen. Außerdem sollen Diagnose- und Programminformationen des configurAIDERS abgerufen werden können.

#### *Bewertung:*

Einstufung in Klasse „Essential“.

### 3.2.10 Steuern und Beobachten von Ein-/Ausgängen und Timern

#### *Beschreibung:*

Die Zustände der digitalen und analogen Ausgänge sowie der internen Timer sollen gesteuert und beobachtet werden können. CAN- und LIN-Botschaften sollen gesendet und empfangen werden können. Dabei soll einmaliges und zyklisches Senden von Botschaften möglich sein. Zusätzlich soll die Möglichkeit bestehen, Buskommunikation innerhalb eines Zeitfensters mitzuloggen. Alle Busbotschaften, die innerhalb dieses Zeitfensters auf dem Bus liegen, sollen dabei mit Zeitstempel gelistet werden.

*Bewertung:*

- Texteditor zur Erstellung von PAWN-Skripten
  - Schreiben von PAWN-Code
  - mehrere Skripte lassen sich parallel über Tabs öffnen
  - Copy/Paste
  - Suchen/Ersetzen
  - Möglichkeit, letzte Schritte rückgängig zu machen
  - Anzeige aller implementierten Funktionen zur Navigation im Skript
  - Anzeige aller instanziierten Variablen. Einfügen des Variablennamens per Drag&Drop.
  - Einfügen von Variablennamen aus Datenbankdateien (.dbc/.arxml)
  - Routing-Editor. Erleichtert Codeentwicklung für GIGABOX Use-Case „Einsatz als Gateway“.
    - \* Laden und Anzeigen von Botschaften/Signalen aus Datenbank (.dbc/.arxml)
    - \* Eigene Botschaft kann definiert werden mit ID sowie Channel, auf die sie gelegt werden soll
    - \* Inhalt der eigenen Botschaft kann aus Signalen, die in Datenbank definiert sind, zusammengestellt werden
    - \* Aus der im Routing-Editor zusammengestellten Botschaft kann PAWN-Code generiert werden und in die Funktion OnCanRxEvent() eines geöffneten Skriptes im Texteditor eingefügt werden
  - Bei der Codeentwicklung unterstützende Funktionen, z.B. Autovervollständigung. Ziel: Schnelles und effektives Schreiben von fehlerfreiem Code
- Konfigurieren einer GIGABOX ohne PAWN-Kenntnisse
  - Realisierung von Funktionen ohne PAWN-Kenntnisse

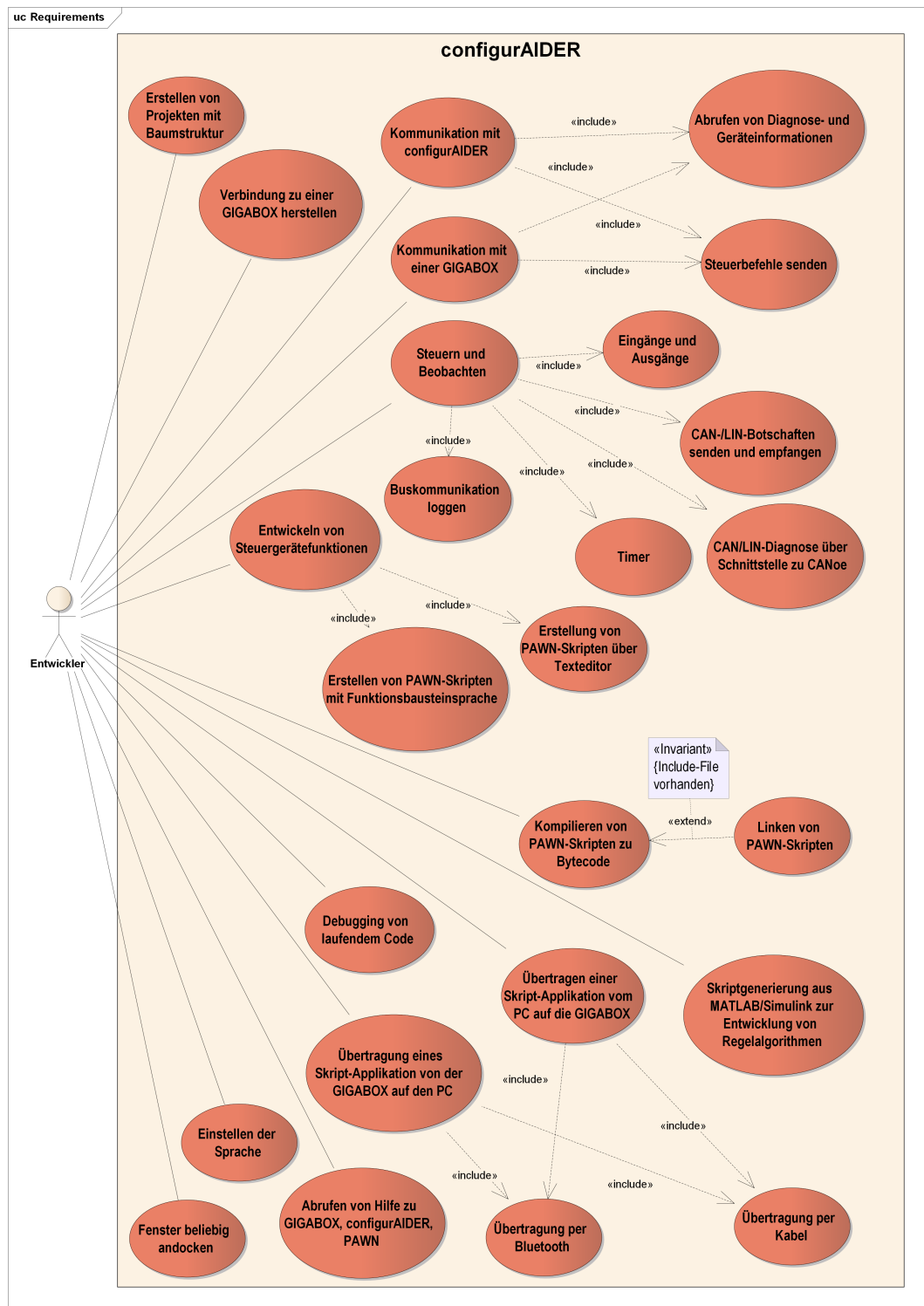
- Realisierung von Funktionen mit Funktionsbausteinsprache
- Frei andockbare Fenster
- Verbindung zu einer GIGABOX herstellen
  - Anzeige aller mit dem PC per Kabel oder Bluetooth verbundenen GIGABOXEN
  - Benutzer kann GIGABOX auswählen, mit der kommuniziert werden soll
- PAWN-Compiler
  - Compiler zum Übersetzen von PAWN-Code in Bytecode
  - Linker, um Bytecode zusammenzufügen
- Beschreiben/Auslesen des Flash-Speichers der GIGABOX
  - Übertragen der Skript-Applikation vom PC auf den Flash-Speicher der GIGABOX per Kabel/Bluetooth
  - Übertragen der Skript-Applikation von einer GIGABOX auf den PC per Kabel/Bluetooth
- Kommunikation mit einer GIGABOX
  - Benutzer kann Befehle an eine GIGABOX senden (per Kabel/Bluetooth). Bsp: Reset der GIGABOX veranlassen, Bootloader aufrufen, Skript starten/stoppen
  - Benutzer kann Diagnose- und Geräteinformationen von GIGABOX abrufen (per Kabel/Bluetooth)
- Kommunikation mit dem Benutzer
  - Benutzer kann Konsolenbefehle an configurAIDER senden
  - Benutzer kann Diagnose- und Programminformationen von configurAIDER abrufen
- Steuern und Beobachten von Ein-/Ausgängen und Timer
  - Steuern und Beobachten von DIN, AIN, DOUT, SWITCH
  - CAN-/LIN-Botschaften senden (einmalig und zyklisch) und empfangen

- CAN/LIN beobachten über Schnittstelle zu CANoe
  - Steuern und Beobachten der internen Timer
  - Buskommunikation loggen
- Debugging
  - Setzen von Breakpoints
  - Zeilenweise Steuerung der Anweisungsausführung
- Skriptgenerierung aus MATLAB/Simulink
  - Modellbasierte Entwicklung von Regelalgorithmen in MATLAB/Simulink. Generierung eines PAWN-Skriptes aus Simulink-Modell.
- Bereitstellung von Dokumentationen
  - Dokumentation zur GIGABOX
  - Dokumentation zum configurAIDER
  - Dokumentation zu PAWN

Das Use-Case-Diagramm in Abbildung 3.1 veranschaulicht die grundlegenden Funktionen, die an das Tool gestellt werden.

### 3.3 Ermittlung von nichtfunktionalen Anforderungen

- Entwicklung des Tools mit WPF unter Verwendung MVVM-Pattern
- Entwicklung für Windows 32bit + 64bit als Zielbetriebssystem
- Entwicklung nach V-Modell
- Qualitätskriterien an Software nach ISO 9126 (aus Wiki)/ siehe auch Kapitel 6 Moderne Softwarearchitektur
  - Wartbarkeit
    - \* Analysierbarkeit:  
Coding-Richtlinien sollen eingehalten werden: Verständliche Kommentare, einheitliche Namensgebung etc



### Abbildung 3.1 – Use-Case-Diagramm funktionale Anforderungen

- \* Modifizierbarkeit:  
SOLID-Prinzipien einhalten, um möglichst wenig Abhängigkeiten unter den Klassen zu erreichen. Dadurch treten bei Modifikationen innerhalb einer Klasse weniger schwer vorhersehbare Fehler auf
- \* Testbarkeit:  
Keine Ahnung auf was bei der Entwicklung geachtet werden soll um gute Testbarkeit zu erreichen
- Benutzbarkeit (genauer definiert in Norm EN ISO 9241-110)
- Effizienz
  - \* Akzeptable Zeit bis Tool gestartet ist und verwendbar ist
  - \* Unmittelbare und flüssige Reaktion auf Benutzereingaben
- Übertragbarkeit
  - \* Anpassbarkeit: Möglichkeit, Software an verschiedene Umgebungen anzupassen  
Verschiedene Betriebssysteme? 32/64Bit?
- Zuverlässigkeit
  - \* Reife: Geringe Versagenshäufigkeit durch Fehlerzustände: Ausreichendes Testing
  - \* Wiederherstellbarkeit: Fähigkeit, bei einem Versagen das Leistungsniveau wiederherzustellen und die direkt betroffenen Daten wiederzugewinnen. Zu berücksichtigen sind die dafür benötigte Zeit und der benötigte Aufwand.

## 3.4 Priorisierung der Anforderungen

Wie soll Bewertung vorgenommen werden?? Wie begründen? Anforderungen müssen quantifiziert werden

Die Priorisierung der Anforderungen soll qualitativ erfolgen anhand von Meinungen befragter Entwickler. Die Anforderungen werden eingeordnet in vier Klassen:

- A: Anforderung, die essenziell wichtige Grundfunktionalitäten bereitstellt
- B: Anforderung, die wichtige Funktionalitäten bereitstellt



- C: Anforderung, die mäßig wichtige Funktionalitäten bereitstellt
- D: Anforderung, die wenige wichtige Funktionalitäten bereitstellt

### **3.4.1 Funktionale Anforderungen**

# Kapitel 4

## Ist-Zustand

### 4.1 Überblick

In diesem Kapitel wird der Ist-Zustand des configurAIDERS beschrieben. Es wird die Benutzeroberfläche vorgestellt und die Funktionalität beschrieben. Im Anschluss wird die Funktionalität und die Usability bewertet. Abbildung 4.1 veranschaulicht die Module, die im configureAIDER enthalten sind.

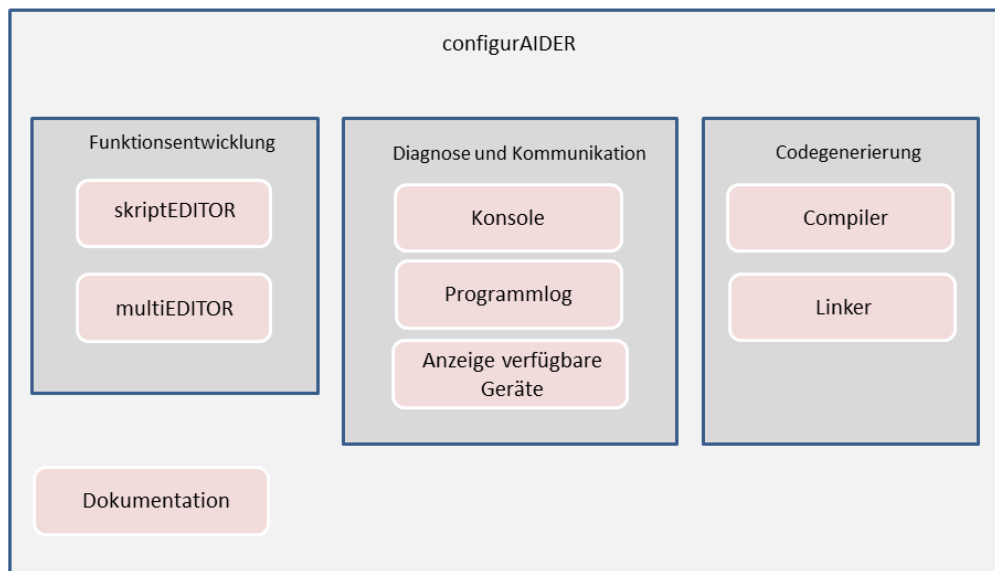
### 4.2 Elemente des configurAIDER

#### 4.2.1 Rahmenfenster

Der configurAIDER besteht aus einem Rahmenfenster, in das weitere Fenster eingebettet werden können. Das Rahmenfenster besitzt im oberen Fensterbereich eine Menüleiste in horizontaler Ausrichtung und ist in einem Hintergrund gehalten, der Firmenlogo und -farbe zeigt. Die Menüeinträge ändern sich dynamisch in Abhängigkeit der vom Benutzer geöffneten Fenster. Beim Start des configurAIDERS ist das Fenster „Verfügbare Geräte“ standardmäßig in das Rahmenfenster eingebettet. Eingebettete Fenster können nicht über die Grenzen des Rahmenfensters hinausgezogen werden.

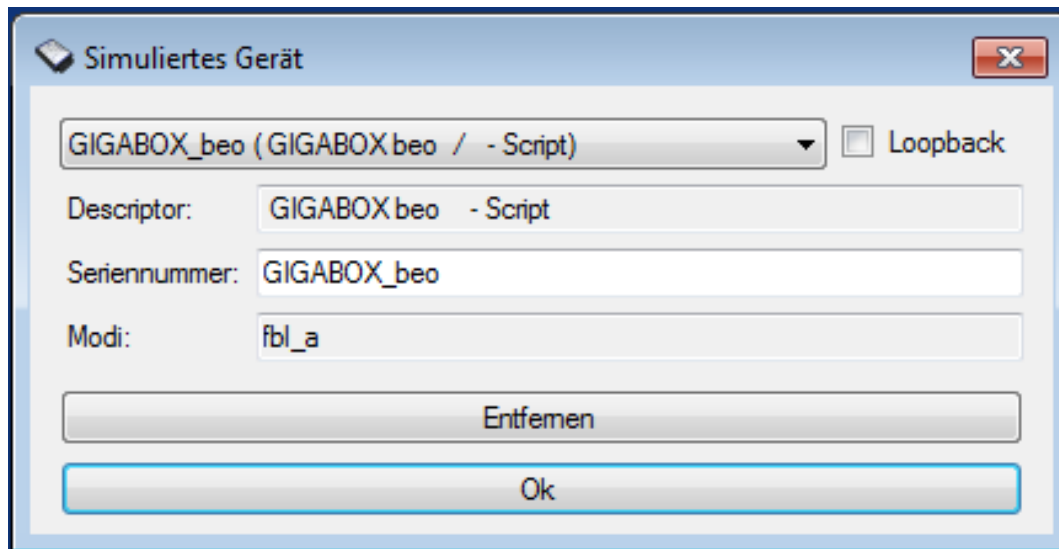
#### 4.2.2 Fenster „Verfügbare Geräte“

Im Fenster „Verfügbare Geräte“ werden alle GIGABOXen angezeigt, die an der USB-Schnittstelle des PC angeschlossen sind. Es wird der Modellname, die



**Abbildung 4.1** – Übersicht configurAIDER

Seriennummer und der Verbindungsstatus von GIGABOXen ausgegeben, die am PC erkannt wurden. Über dieses Fenster kann eine GIGABOX angewählt werden, mit der eine Kommunikation gestartet werden soll. Wenn keine reale GIGABOX zur Verfügung steht, kann eine GIGABOX simuliert werden. Die Auswahl eines zu simulierenden Gerätes erfolgt über ein eigenes Dialogfenster „Simuliertes Gerät“ (siehe Abbildung 4.2).



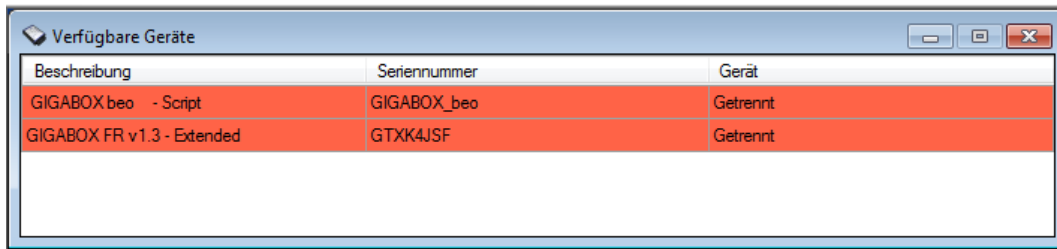
**Abbildung 4.2** – Fenster „Simuliertes Gerät“

Die simulierte GIGABOX wird wie eine reale GIGABOX gelistet.

Nach der Anwahl eines GIGABOX-Modelles wird versucht, eine Verbindung zu dem Gerät herzustellen (wird das tatsächlich hardwaremäßig versucht oder ist die Anzeige nur dazu da, dem Benutzer zu signalisieren, welches Gerät er konfiguriert). Bei erfolgreicher Verbindungsherstellung wird das Gerät grün hinterlegt und es öffnet sich ein Fenster zur Editorauswahl. Dort kann ausgewählt werden, ob die Konsole, der scriptEDITOR oder der multiEDITOR geöffnet werden soll. Bei einigen GIGABOX-Modellen steht kein multiEDITOR zur Verfügung.

#### 4.2.3 Fenster „Konsole“

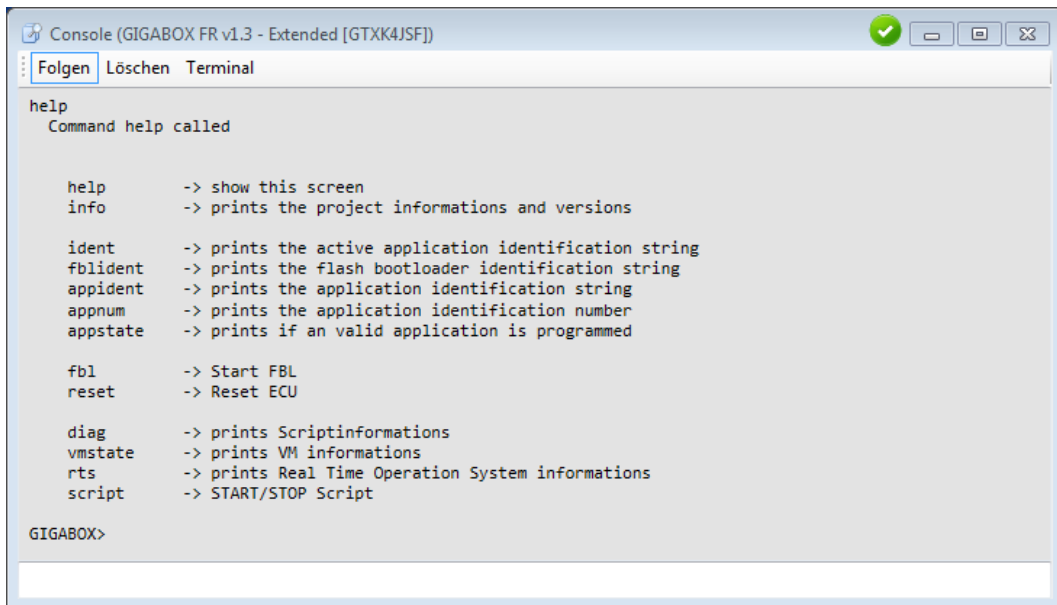
Die Konsole besteht aus einer Textbox zur Eingabe von Befehlen und einer Textbox zur Ausgabe von Informationen. Sie ermöglicht Kommunikation mit einer GIGABOX. Die Konsole kann z.B eingesetzt werden zur Abfrage von Diagnoseinformationen und um einen Reset zu veranlassen.



Beschreibung	Seriennummer	Gerät
GIGABOX beo - Script	GIGABOX_beo	Getrennt
GIGABOX FR v1.3 - Extended	GTXXK4JSF	Getrennt

Abbildung 4.3 – Fenster „Verfügbare Geräte“

Über die Schaltflächen „Folgen“ und „Terminal“ kann gewählt werden, wie die Textein- und ausgabe erfolgen soll. Wenn „Folgen“ ausgewählt wird, erfolgt die Texteingabe in einem von der Textausgabe getrennten Fenster. Wenn „Terminal“ ausgewählt wird, erfolgt die Textein- und ausgabe in einem gemeinsamen Fenster.



```

help
  Command help called

help      -> show this screen
info      -> prints the project informations and versions

ident     -> prints the active application identification string
fblident  -> prints the flash bootloader identification string
appident  -> prints the application identification string
appnum    -> prints the application identification number
appstate  -> prints if an valid application is programmed

fbl       -> Start FBL
reset     -> Reset ECU

diag      -> prints Scriptinformations
vmstate   -> prints VM informations
rts       -> prints Real Time Operation System informations
script    -> START/STOP Script

GIGABOX>
  
```

Abbildung 4.4 – Fenster „Konsole“

#### 4.2.4 Fenster „Programmlog“

Das „Programmlog“-Fenster besteht aus einer Textbox, über die tabellarisch Erfolgs- und Fehlermeldungen über die ausgeführten Aktionen des configurAI-DERs ausgegeben werden.

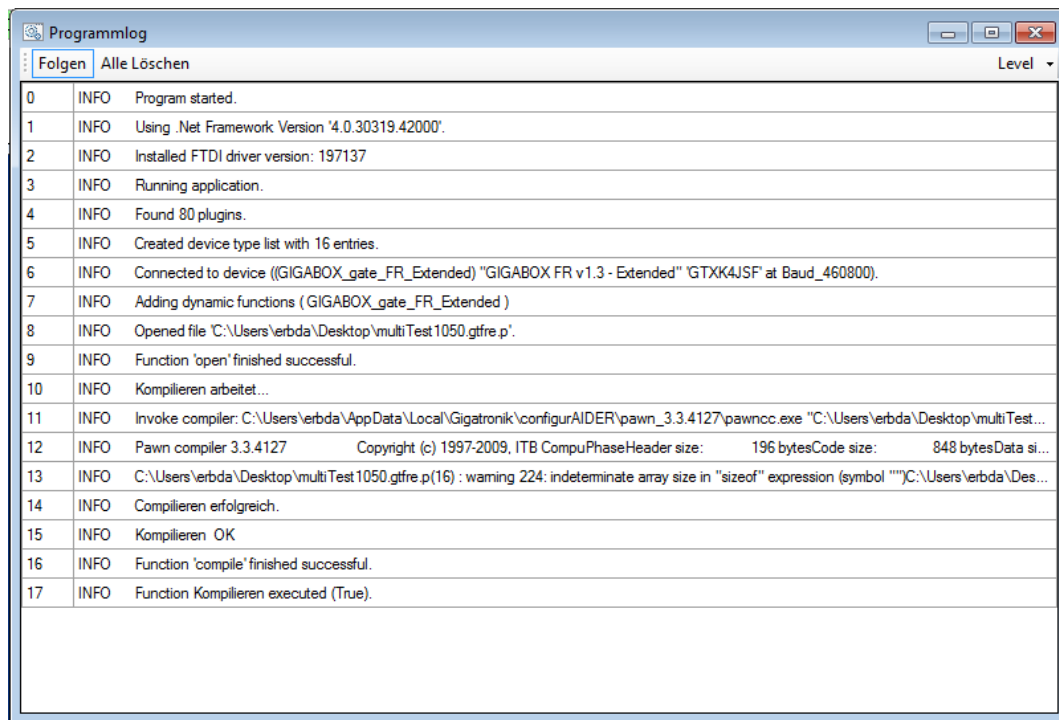


Abbildung 4.5 – Fenster „Programmlog“

#### 4.2.5 Fenster „scriptEDITOR“

Der scriptEDITOR ist ein Texteditor zum Erstellen und Editieren von Funktionen in PAWN. Um den scriptEDITOR aufzurufen, muss zuerst eine Verbindung zu einer realen oder simulierten GIGABOX hergestellt werden über das Fenster „Verfügbare Geräte“. Anschließend öffnet sich automatisch ein Fenster zur Editorauswahl. Dort kann der scriptEDITOR angewählt werden. Mit dem scriptEDITOR erstellte Skriptdateien haben die Dateiendung .gt\*\*.p. Die Sterne sind zu ersetzen mit einem Kürzel, das abhängig ist vom verbundenen GIGABOX-Modell. Für die unterschiedlichen GIGABOX-Modelle werden also Skripte mit verschiedenen Dateiendungen erstellt, da die Modelle unterschiedliche Funktionen implementieren. Bsp: GIGABOX gate FR extended: .gtfre.p GIGABOX gate XL: .gtxl.p GIGABOX gate gateway: .gtfrg.p

In der obersten Fensterzeile ist eine Buttonleiste angeordnet mit den folgenden Buttons:

- Neu: Öffnet ein neues Skript
- Öffnen: Öffnet ein vorhandenes Skript

- Speichern: Speichert das geöffnete Skript unter einem bereits festgelegt Dateinamen und Pfad
- Speichern unter: Speichert das geöffnete Skript unter einem anzugebenden Dateinamen und Pfad
- Kompilieren: Interpreter erzeugt Bytecode aus dem PAWN-Quellcode
- Herunterladen: Interpreter erzeugt Bytecode aus dem PAWN-Quellcode. Die PAWN-Skriptdatei wird als ZIP-Datei komprimiert. Anschließend wird der Bytecode und das gezippte Skript auf die virtuelle Maschine der GIGABOX übertragen.
- Heraufladen: Die als ZIP-Datei komprimierte Skriptdatei auf dem Steuergerät wird auf den PC übertragen, entzippt und im Texteditor geöffnet.
- Reset: Führt einen Reset der GIGABOX durch
- Log: Öffnet Konsole

**Aufzählung der Buttons zu detailreich** Mittig ist der Texteditor angeordnet. Hier kann ein Skript angezeigt und bearbeitet werden. Es stehen bei der Entwicklung typische Unterstützungswerkzeuge wie Codefolding zur Verfügung. **(Aufzählung der unterstützenden Werkzeuge wie Autovervollständigung etc.)**

Unten befindet sich ein Fenster zur Ausgabe von Fehlern, Warnungen und Erfolgsmeldungen.

#### 4.2.6 Fenster „multiEDITOR“

Der „multiEDITOR“ ist ein Editor zur Funktionskonfiguration mit WENN-DANN-Anweisungen in Tabellen. Er soll es Benutzern mit geringer Programmiererfahrung ermöglichen, einfache Funktionen für die GIGABOX zu entwickeln. Aus den konfigurierten Funktionstabellen kann anschließend automatisch ein PAWN-Skript generiert werden. Das erstellte PAWN-Skript kann dann kompiliert und der Bytecode auf den Flash-Speicher der GIGABOX übertragen werden.

In der obersten Fensterzeile ist eine Buttonleiste angeordnet mit den folgenden Buttons:

- Neu: Öffnet ein neues Skript
- Öffnen: Öffnet ein vorhandenes Skript

```

1 // Declare variables.
2 new ida_CAN_1[] = { 1, 2 }
3 new ida_CAN_2[] = { }
4 new timeout[] = { }
5 new timeout_counter[] = { }
6 static #undef can_0_1_0 = 0
7 static #undef can_0_2_1 = 0
8
9
10
11 // Handle startup event.
12 public OnStartup()
13 {
14     TimerSet( 0, 1 )
15     CAN_Init(CAN_1, CAN_BAUD_500KBIT_65PERC, CAN_GetArbitrationMask(ida_CAN_1, sizeof(ida_CAN_1)), CAN_GetAcceptanceMask(ida_CAN_1, sizeof(ida_CAN_1)));
16     CAN_Init(CAN_2, CAN_BAUD_500KBIT_65PERC, CAN_GetArbitrationMask(ida_CAN_2, sizeof(ida_CAN_2)), CAN_GetAcceptanceMask(ida_CAN_2, sizeof(ida_CAN_2)));
17 }
18
19 // Handle timeouts.
20 OnTimeoutTimer()
21 {
22     for ( new i_var = 0; i_var < sizeof ( timeout_counter ); i_var++ )
23     {
24         --timeout_counter[i_var];
25         if ( timeout_counter[i_var] == 0 )
26         {
27             // Reset Timeout Counter
28             timeout_counter[i_var] = timeout[i_var]
29             switch (i_var)
30             {
31                 // Timeout Action Code
32
33                 default:
34                 {
35                     break
36                 }
37             }
38         }
39     }
40 }
41
42 // Handle single timer event.
43 public OnTimerFromTimer1 timerNumber 1
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

Pawn compiler 3.3.4127 Copyright (c) 1997-2009, ITS CompuPhase  
 Header size: 196 bytes  
 Code size: 848 bytes  
 Data size: 16 bytes  
 Stack/heap size: 256 bytes; estimated max. use=14 cells (56 bytes)  
 Total requirements: 1316 bytes  
 3 Warnings.  
 C:\Users\verba\Desktop\multitest1050.gtfre.p(16) : warning 224: indeterminate array size in "sizeof" expression (symbol "")  
 C:\Users\verba\Desktop\multitest1050.gtfre.p(16) : warning 224: indeterminate array size in "sizeof" expression (symbol "")  
 C:\Users\verba\Desktop\multitest1050.gtfre.p(23) : warning 224: indeterminate array size in "sizeof" expression (symbol "")  
 Compilieren erfolgreich.  
 Kompilieren OK

Abbildung 4.6 – Fenster „scriptEDITOR“



- Speichern: Speichert das geöffnete Skript unter einem bereits festgelegt Dateinamen und Pfad
- Speichern unter: Speichert das geöffnete Skript unter einem anzugebenden Dateinamen und Pfad
- Generieren: Aus den mittels Tabelle konfigurierten Funktionen wird ein PAWN-Skript generiert
- Kompilieren: Interpreter erzeugt Bytecode aus dem PAWN-Quellcode
- Herunterladen: Aus den mittels Tabelle konfigurierten Funktionen wird ein PAWN-Skript generiert. Interpreter erzeugt Bytecode aus dem PAWN-Quellcode. Die PAWN-Skriptdatei wird als ZIP-Datei komprimiert. Anschließend wird der Bytecode und das gezippte Skript auf die virtuelle Maschine der GIGABOX übertragen.
- Heraufladen: Die als ZIP-Datei komprimierte Skriptdatei auf dem Steuergerät wird auf den PC übertragen und entzippt. PAWN-Code wird rückgewandelt in Tabelle mit WENN-DANN-SONST-Anweisungen. Die funktioniert nur, wenn der von der GIGABOX hochgeladenen Code ursprünglich mithilfe des multiEDITORS erzeugt wurde.
- DBC: Es können Vector-DBC Dateien oder AUTOSAR .arxml Dateien erstellt oder geöffnet werden. Dabei handelt es sich um Datenbanken, in denen CAN-Botschaften und Signale definiert sind
- Reset: Führt einen Reset der GIGABOX durch
- Log: Öffnet Konsole

#### Aufzählung der Buttons zu detailreich

Unter der Buttonleiste befindet sich eine Tabelle, in der CAN-Botschaften und Signale erstellt und bearbeitet werden können. Botschaften und Signale, die aus Datenbank-Dateien (.dbc oder .arxml) geladen wurden, werden hier angezeigt.

Darunter befindet sich eine Tabelle zur Konfiguration von Funktionen mithilfe von WENN-DANN-SONST-Anweisungen.

## 4.3 Funktionalität

Abbildung 4.8 gibt einen Überblick über die verfügbaren Funktionen des configurAIDERS.

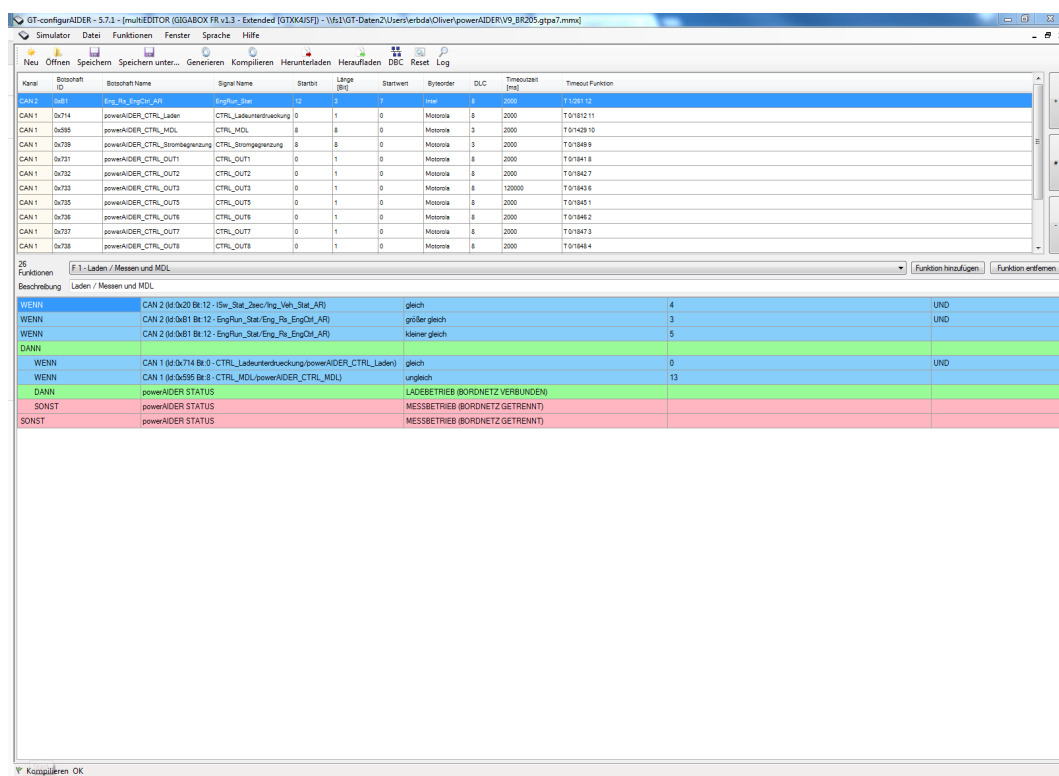


Abbildung 4.7 – Fenster „multiEDITOR“

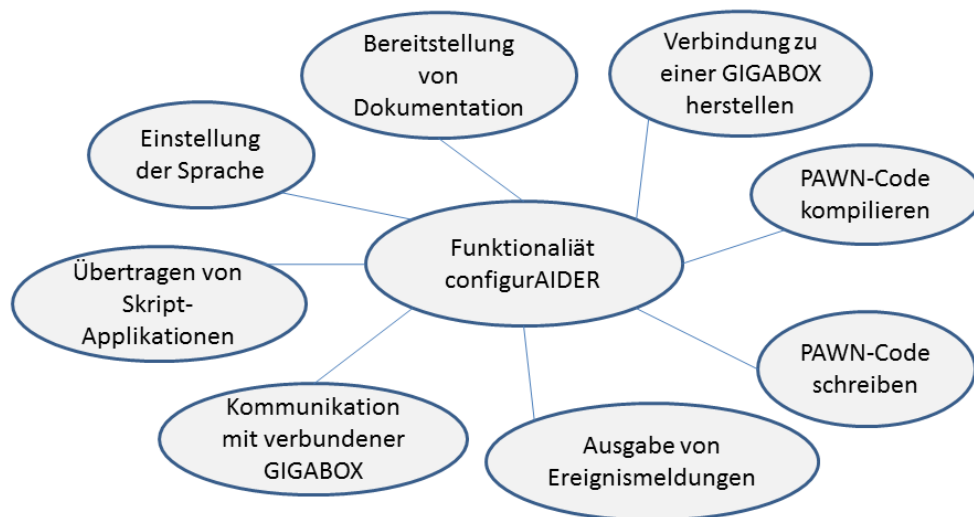


Abbildung 4.8 – Überblick Funktionen des configurAIDERS

### 4.3.1 Einstellung der Sprache

Die Sprache der Fensterdialoge des configurAIDERS kann in der Menüleiste des Rahmenfenster unter dem Eintrag „Sprache“ eingestellt werden. Es steht Deutsch und Englisch als Sprache zur Verfügung. Nach der Sprache von Deutsch auf Englisch wird ein Dialog eingeblendet, dass die Sprachänderung erst nach einem Neustart übernommen wird. Nach ausgeführtem Neustart werden nicht alle Menüeinträge auf Englisch dargestellt. So werden beispielsweise im scriptEDITOR die Menüeinträge unter „Edit“ weiterhin komplett in Deutsch dargestellt. Die Funktionstabellen im multiEDITOR werden auch weiterhin in Deutsch dargestellt.

### 4.3.2 Bereitstellung von Dokumentation zur GIGABOX, configurAIDER und PAWN

Über den Menüeintrag „Hilfe“ können PDF-Dokumente zu den verschiedenen GIGABOX-Modellen, configurAIDER und zur Skriptsprache PAWN abgerufen werden. Für den configurAIDER steht ein allgemeines Handbuch, Hinweise zur Installation und jeweils für scriptEDITOR und multiEDITOR ein eigenes Handbuch zur Verfügung. Der Menüeintrag „linEDITOR“ enthält nur einen ausgegrauten Eintrag. Zu den GIGABOX-Modellen stehen Handbücher, Flyer und Informationen zu Firmwareupdates zur Verfügung. Die Dokumentationen werden nicht durchgängig für alle GIGABOX-Modelle zur Verfügung gestellt. So wird für die GIGABOX gate xl kein Handbuch angeboten, für die GIGABOX gate powerAIDER kein Flyer.

### 4.3.3 Verbindung zu einer GIGABOX herstellen

GIGABOXEN, die an der USB-Schnittstelle des PCs erkannt werden sowie simulierte GIGABOXEN werden im Fenster „Verfügbare Geräte“ (Abbildung 4.3) gelistet. Hier kann die Verbindung zu einem Gerät hergestellt werden. Falls keine reale GIGABOX zur Verfügung steht, kann eine GIGABOX simuliert werden.

Die verschiedenen Modelle besitzen unterschiedliche Hardwarefunktionen (unterschiedliche Anzahl von Ein-/Ausgängen, Bussysteme), die zu entwickeln den Funktionen müssen daran angepasst werden. Dem configurAIDER muss deshalb vor der Funktionsentwicklung bekannt sein, für welches Modell Code entwickelt werden soll.

Es wurden verschiedene Fehler entdeckt, die im Folgenden aufgelistet werden.

1. Das Fenster „Verfügbare Geräte“ wird vom Benutzer geschlossen. Bei einem Wiederaufruf des Fensters kann kein neues Gerät mehr simuliert werden. Lösung: configurAIDER neu starten
2. Im Fenster „Verfügbare Geräte“ sind mehrere Geräte gelistet. Bei dem Versuch, die Verbindung zu einem Gerät herzustellen über Rechtsklick auf das Gerät und „Verbinden“, wird keine Verbindung zum angewählten Gerät hergestellt sondern zum ersten Gerät in der Liste. Lösung: Verbindung herstellen über Doppelklick auf Gerät
3. Die Simulation von Geräten schlägt gelegentlich fehl, es wird über das Programmlog-Fenster ein Fehler ausgegeben (ERROR: Internal device table obscured!) Lösung: Neuer Versuch der Herstellung einer Verbindung

#### 4.3.4 PAWN-Code kompilieren

Aus scriptEDITOR und multiEDITOR heraus kann mithilfe des Compilers PAWN-Code in Bytecode übersetzt werden. Der erzeugte Bytecode kann auf der virtuellen Maschine der GIGABOX ausgeführt werden.

#### 4.3.5 PAWN-Code entwickeln

##### PAWN-Skripte schreiben

PAWN-Skripte können im scriptEDITOR geschrieben werden. Erstellte Skriptdateien besitzen die Dateiendung .gt\*.p. Die Sterne sind zu ersetzen mit einem Kürzel, das abhängig ist vom verbundenen GIGABOX-Modell. Für die unterschiedlichen GIGABOX-Modelle werden also Skripte mit verschiedenen Dateiendungen erstellt, da die Modelle unterschiedliche Funktionen implementieren. Bsp: GIGABOX gate FR extended: .gtfre.p GIGABOX gate XL: .gtxl.p GIGABOX gate gateway: .gtfrg.p

Um den Benutzer bei der Codeentwicklung zu unterstützen, sind aus anderen IDEs bekannte Funktionen integriert. Einige ausgewählte Funktionen sind hier beispielhaft angeführt:

- Codeabschnitte sind auf- und zuklappbar
- Unterschiedliche Farbgebung (Anweisungen in blau, Kommentare in grün etc)

- Eventfunktionen sind als Rumpf vorimplementiert
- Markierung der Zeile, in der sich der Cursor aktuell befindet

Im Vergleich zu weit verbreiteten IDEs wie Visual Studio ist der Umfang an unterstützenden Funktionen aber klein gehalten.

### **PAWN-Skripte aus Funktionstabellen generieren**

Im multiEDITOR können tabellarisch Funktionen konfiguriert werden mit WENN-DANN-SONST-Anweisungen. Aus den konfigurierten Funktionstabellen kann anschließend automatisch ein PAWN-Skript generiert werden. Das erstellte PAWN-Skript kann dann kompiliert und der Bytecode auf den Flash-Speicher der GIGABOX übertragen werden.

Für jede Funktion wird eine neue Tabelle erstellt. Jede Tabelle besteht mindestens aus 3 Zeilen mit den Anweisungen WENN, DANN und SONST. Es können mehrere WENN-Anweisungszeilen erstellt werden, die mit dem Bedingungsoperator UND oder ODER verknüpft werden müssen. Außerdem können mehrere DANN und SONST-Anweisungen erstellt werden, die jeweils miteinander über den UND-Operator verknüpft werden. Unter jeweils jede DANN und SONST-Anweisung kann eine neue, untergeordnete WENN-DANN-SONST-Anweisung eingefügt werden. Damit sind verschachtelte Anweisungen möglich.

In den Spalten der WENN-Anweisungszeile können digitale und analoge Eingänge, digitale Ausgänge sowie eingehende Busbotschaften auf definierte Werte geprüft werden. Wenn die dort definierten Werte angenommen werden, wird die DANN-Anweisungszeile ausgeführt. Hier können digitale Ausgänge durchgeschaltet oder ausgeschaltet werden. Das Absetzen von Busbotschaften ist nicht möglich.

Wenn die in der WENN-Anweisungszeile definierten Werte nicht angenommen werden, wird die SONST-Anweisungszeile ausgeführt. Auch hier können digitale Ausgänge durchgeschaltet oder ausgeschaltet werden.

### **Bewertung**

#### **4.3.6 Übertragen von Skript-Applikationen**

Es kann eine Skript-Applikation auf die GIGABOX übertragen werden. Die Skriptdatei wird als ZIP-Datei komprimiert und zusammen mit dem Bytecode auf den Flash-Speicher geschrieben. Wenn ein Skript mithilfe des multiEDITORS generiert wurde, wird zusätzlich zu der Skriptdatei (.gt\*\*.p) auch

die multiEDITOR-Konfigurationsdatei (.gt\*\*.mmx) (+.tmp -> was steht da drin?) mit in die ZIP-Datei gepackt.

Ein auf der GIGABOX ausgeführtes Skript kann auch von der GIGABOX auf den PC geladen werden. Dabei wird die auf der GIGABOX gespeicherte ZIP-Datei auf den PC geladen, entpackt und im scriptEDITOR angezeigt. Wenn das Skript mit dem multiEDITOR erzeugt wurde, liegt der ZIP-Datei zusätzlich die multiEDITOR-Konfigurationsdatei (.gt\*\*.mmx) bei und kann folglich auch im multiEDITOR geöffnet werden.

#### 4.3.7 Kommunikaton mit verbundener GIGABOX

Über die Konsole können Befehle an eine reale GIGABOX gesendet werden, z.B. Befehl zum Reset, Aufrufen des Bootloaders. Dort können auch Informationen der GIGABOX abgerufen und ausgegeben werden, z.B. Diagnoseinformationen über Bussysteme oder Timer.

Wenn die Schaltfläche „Terminal“ aktiviert ist, kann eine erfolgte Texteingabe nicht mehr korrigiert werden. Bei deaktivierter Schaltfläche ist die Korrektur möglich.

#### 4.3.8 Ausgabe von Ereignismeldungen

Es kann über das Programmlog-Fenster nachverfolgt werden, welche Aktionen vom configurAIDER durchgeführt wurden. Wichtige Erfolgs- und Fehlermeldungen werden dort ausgegeben. Bsp: Erfolgreich kompiliert.

#### 4.3.9 Ergebnis

# Kapitel 5

## Evaluation

### 5.1 Funktionalität und Usability des Ist-Zustands

Es sollen die Funktionen des configurAIDERS bewertet werden hinsichtlich Umfang, Umsetzung und Fehlerzuständen. Zusätzlich soll die Usability nach EN ISO 9241-110 (siehe Abschnitt 2.2.2) bewertet werden.

#### 5.1.1 Einstellung der Sprache

Es kann Deutsch und Englisch als Sprache eingestellt werden. Hier wäre zu überlegen, ob noch weitere Spracheinstellungen zur Verfügung gestellt werden sollten, falls es eine größere Anzahl an Benutzern aus nicht deutsch- oder englischsprachigen Ländern gibt. Die englische Spracheinstellung ist nicht konsistent umgesetzt, hier sollte darauf geachtet werden, dass durchgängig alle Bedienelemente englisch beschriftet werden.

#### 5.1.2 Bereitstellung von Dokumentation zur GIGABOX, configurAIDER und PAWN

Über den Menüeintrag „Hilfe“ können PDF-Dokumente zu den verschiedenen GIGABOX-Modellen, configurAIDER und zur Skriptsprache PAWN abgerufen werden. Für den configurAIDER steht ein allgemeines Handbuch, Hinweise zur Installation und jeweils für scriptEDITOR und multiEDITOR ein eigenes Handbuch zur Verfügung. Hier ist zu kritisieren, dass der Menüeintrag „linEDITOR“ nur einen ausgegrauten Eintrag enthält und somit unnötig



ist. Zu den GIGABOX-Modellen stehen Handbücher, Flyer und Informationen zu Firmwareupdates zur Verfügung. Leider werden die Dokumentationen nicht durchgängig für alle GIGABOX-Modelle zur Verfügung gestellt. So wird für die GIGABOX gate xl kein Handbuch angeboten, für die GIGABOX gate powerAIDER kein Flyer.

### 5.1.3 Verbindung zu einer GIGABOX herstellen

Für den Benutzer ist nicht zu unterscheiden, ob ein Gerät in der Liste real oder simuliert ist. Simulierte Geräte werden in der Liste nicht hervorgehoben oder markiert. Die gewünschte Aufgabenangemessenheit ist für dieses Fenster damit nicht gegeben, da der Benutzer die Auswahl einer GIGABOX nicht effizient erledigen kann.

Die Auswahl eines zu simulierenden Gerätes erfolgt über ein eigenes Dialogfenster „Simuliertes Gerät“ (siehe Abbildung 4.2). Für den Benutzer ist in diesem Dialog nicht ersichtlich, was die Checkbox „Loopback“ bewirkt. Die Selbsterklärungsfähigkeit des Dialogs ist nicht ausreichend.

Bei der Auswahl eines zu simulierenden Gerätes im Fenster „Simuliertes Gerät“ wird das angewählte Gerät der Liste im Fenster „Verfügbare Geräte“ hinzugefügt, bevor die Auswahl mit „Ok“ bestätigt wird. Diese Reaktion entspricht nicht den üblichen, dem Benutzer bekannten Konventionen von Software und ist damit nicht erwartungskonform.

Aufgrund von diverse Fehlerzuständen, die beobachtet wurden, kann die Verbindung zu einem Gerät nicht zuverlässig und fehlerfrei hergestellt werden. Das Tool produziert Ergebnisse, die vom Benutzer so nicht erwartet werden und erfordert lästiges Neustarten des Programmes.

### 5.1.4 PAWN-Code kompilieren

Die Kompilierung funktioniert ohne Probleme.

### 5.1.5 PAWN-Code entwickeln

#### PAWN-Skripte schreiben

Keine Möglichkeit, letzte Aktion rückgängig zu machen -> schlechte Steuerbarkeit

Keine Erkennung von Syntaxfehlern -> schlechte Selbstbeschreibungsfähigkeit

Buttons Herunterladen/Heraufladen sind missverständlich -> schlechte Selbstbeschreibungsfähigkeit

Nur ein Skript kann geöffnet werden, nicht mehrere gleichzeitig -> Aufgabenangemessenheit

Autovervollständigung nur teilweise umgesetzt -> Aufgabenangemessenheit

Codeblöcke können nicht auskommentiert werden über Buttons/Tastenkombination -> Aufgabenangemessenheit

Die Buttons „Herunterladen“ und „Heraufladen“ in der oberen Buttonleiste sind nicht klar verständlich benannt. Dem Benutzer wird nicht klar, in welche Richtung der Datenaustausch erfolgt. Der Button „Log“ öffnet die Konsole, die Erwartung wäre, dass sich das Programmlog-Fenster öffnet. Die Selbstbeschreibungsfähigkeit und Erwartungskonformität der Buttonleiste ist negativ zu bewerten.

### **PAWN-Skripte aus Funktionstabellen generieren**

Im Signaleditor ist Bit/CAN-Bit Unterschied nicht klar ersichtlich -> Selbstbeschreibungsfähigkeit

Es können keine CAN-Nachrichten gesendet werden -> eingeschränkte Funktionalität

Keine Reaktion auf LIN-Nachrichten möglich/ kein Senden von LIN-Nachrichten möglich

Keine Möglichkeit, SWITCH CASE-Anweisung einzufügen. Deshalb müssen verschachtelte WENN DANN-Anweisungen erstellt werden, die schnell unübersichtlich werden

Es ist nicht ersichtlich, dass über den DBC-Button auf .arxml-Dateien geladen werden können

Bei Klick auf DBC-Button öffnet sich Fenster zum Laden von .dbc und .armxl-Dateien. In diesem Fenster befindet sich eine Tabelle zur Anzeige der geladenen Signale. Bei Auswahl einer .dbc-Datei bleibt die Tabelle allerdings leer. Die im ausgewählten DBC definierten Signale werden dort nicht angezeigt

### **5.1.6 Übertragen von Skript-Applikationen**

Das Übertragen von Skript-Applikationen funktioniert ohne Probleme.

### **5.1.7 Kommunikaton mit verbundener GIGABOX**

Wenn die Schaltfläche „Terminal“ aktiviert ist, kann eine erfolgte Texteingabe nicht mehr mit der Rücktaste korrigiert werden. Bei deaktivierter Schaltfläche ist die Korrektur möglich. Die Konsole mangelt es folglich an Fehlertoleranz, was einer schlechten Usability entspricht.

### **5.1.8 Ausgabe von Ereignismeldungen des configurAIDERS**

Die Ausgabe von Ereignismeldungen über das Programmlog-Fenster ist zufriedenstellend.

## **5.2 Differenz zwischen Anforderungen und Ist-Zustand**

	Anforderungen	Wichtigkeit	Ist-Zustand
Erstellung von Projekten	Innerhalb eines Projektes können verschiedene GIGABOX-Modelle angelegt werden	B	Nicht realisiert
	Für jedes angelegte GIGABOX-Modell können PAWN-Skripte erstellt werden mit der zum Modell passenden Dateiendung	B	Nicht realisiert
	Für jedes angelegte GIGABOX-Modell können Include-Files hinzugefügt werden	B	Nicht realisiert
Einstellung der Sprache	Deutsch	B	Realisiert
	Englisch	A	Realisiert, aber nicht konsistent umgesetzt
Oberfläche individualisierbar	Frei andockbare Fenster	C	Fenster sind frei verschiebbar, aber nicht andockbar
Verbindung zu einer GIGABOX herstellen	Anzeige aller mit dem PC per Kabel oder Bluetooth verbundenen GIGABOXEN	A	Realisiert im Fenster „Verfügbare Geräte“
	Benutzer kann Verbindung zu GIGABOX herstellen, mit der kommuniziert werden soll	A	Realisiert im Fenster „Verfügbare Geräte“. Es treten Fehler auf.
PAWN-Compiler	Compiler zum Übersetzen von PAWN-Code in Bytecode	A	Realisiert
	Linker um Bytecode zusammenzufügen	A	Realisiert
Beschreiben/Auslesen des Flash-Speichers der GIGABOX	Übertragen der Skript-Applikation vom PC auf den Flash-Speicher der GIGABOX per Kabel/Bluetooth	A	Realisiert. Keine Übertragung per Bluetooth
	Übertragen der Skript-Applikation von einer GIGABOX auf den PC per Kabel/Bluetooth	B	Realisiert. Keine Übertragung per Bluetooth

Kommunikation mit einer GIGABOX	Benutzer kann Befehle an eine GIGABOX senden	A	Realisiert im Fenster "Konsole"
	Benutzer kann Diagnose- und Geräteinformationen der GIGABOX abrufen	A	Realisiert im Fenster "Konsole"
Kommunikation mit dem Benutzer	Benutzer kann Konsolenbefehle an configurAIDER senden	C	Nicht realisiert
	Benutzer kann Diagnose- und Programminformationen abrufen	A	Realisiert im Fenster "Programmlog" sowie im Ausgabefenster des scriptEDITOR und multiEDITOR
Steuern- und Beobachten von Ein- und Ausgängen und Timern	Steuern und Beobachten von DIN, AIN, DOUT, SWITCH	C	Nicht realisiert
	CAN-/LIN-Botschaften senden (einmalig und zyklisch) und empfangen	A	Realisiert im scriptEDITOR. Im multiEDITOR können keine Botschaften gesendet werden
	CAN/LIN beobachten über Schnittstelle zu CANoe	D	Nicht realisiert
	Steuern und Beobachten der internen Timer	C	Nicht realisiert
Debugging	Setzen von Breakpoints	C	Nicht realisiert
	Zeilenweise Steuerung der Anweisungsausführung	C	Nicht realisiert
Skriptgenerierung aus MATLAB/Simulink	Modellbasierte Entwicklung von Regelalgorithmen in MATLAB/Simulink. Generierung eines PAWN-Skriptes aus Simulink-Modell.	D	Nicht realisiert
Bereitstellung von Dokumentationen	Dokumentation zur GIGABOX	B	Realisiert, aber nicht konsistent umgesetzt
	Dokumentation zum configurAIDER	B	Realisiert
	Dokumentation zu PAWN	B	Realisiert

Texteditor zur Erstellung von PAWN-Skripten	Schreiben von PAWN-Code	A	Realisiert
	Mehrere Skripte lassen sich parallel über Tabs öffnen	C	Nicht realisiert
	Copy/Paste	B	Realisiert
	Suchen/Ersetzen	B	Realisiert
	Möglichkeit, letzte Schritte rückgängig zu machen	B	Nicht realisiert
	Anzeige aller implementierten Funktionen zur Navigation im Skript	C	Nicht realisiert
	Anzeige aller instanziierten Variablen. Einfügen des Variablennamens per Drag&Drop	C	Nicht realisiert
	Routing-Editor	C	Nicht realisiert
	Einfügen von Variablen aus Datenbankdateien (.dbc/.arxml)	D	Nicht realisiert
	Funktionen, die bei der Codeentwicklung unterstützen, z.B. Autovollständigung	C	Realisiert, aber Funktionsumfang ist mäßig
Konfigurieren einer GIGABOX ohne PAWN-Kenntnisse	Realisierung von Funktionen mit Funktionsbausteinsprache	C	Nicht mit Funktionsbausteinsprache realisiert sondern mit Anweisungstabellen

## 5.3 Quellcode des configurAIDERS

Was wurde mit WinForms gemacht, was mit WPF?

Entstand aus mehreren studentischen Arbeiten

Diagramm mit bestehender Architektur erstellen

## 5.4 Ergebnis

# Kapitel 6

## Konzeption

### 6.1 Übersicht

Ziel des Kapitels ist es, Konzepte aufzuzeigen, wie die in der Anforderungsanalyse erarbeiteten Anforderungen umgesetzt werden können.

### 6.2 Erweiterung des bestehenden configurAIDERS um neue Features

Der bestehende Code des configurAIDERS wird erweitert um neue Features, die bei der Bewertung der Anforderungen in Abschnitt 3.4 mit (+) oder (++) bewertet wurden. Die an bereits bestehenden Features in Abschnitt 4.3 kritisierten Punkte sollen verbessert werden.

Neue Funktionen:

- Projektverwaltung: Es können Projekte angelegt werden, in denen unterschiedliche Modelle der GIGABOX projiziert sind und die zugehörigen Skripte+Include-Files
- Texteditor wird erweitert um:
  1. Möglichkeit, letzte Schritte rückgängig zu machen
  2. Anzeige aller implementierten Funktionen zur Navigation im Skript
  3. Bei der Codeentwicklung unterstützende Funktionen



#### 4. Routing-Editor

- Frei andockbare Fenster
- Steuern und Beobachten von DIN, AIN, DOUT, SWITCH

Vorteil:

- Das Tool erhält damit einen erhöhten Funktionsumfang, als wichtig bewertete funktionale Anforderungen können umgesetzt werden.

Nachteil:

- Nichtfunktionale Anforderungen wie Qualitätskriterien an Software nach ISO9126 (Wartbarkeit, Usability, Effizienz, Übertragbarkeit, Zuverlässigkeit) können nicht erfüllt werden, sondern werden sich verschlechtern durch eine weitere Funktionserweiterung des Tools
- Viel Einarbeitung in bestehenden Code nötig

### 6.3 Neuentwicklung des configurAIDERS

Das Tool wird von Grund auf neu entwickelt. Es sollen die funktionalen und nichtfunktionalen Anforderungen aus Kapitel 3 umgesetzt werden. Im Rahmen dieser Arbeit sollen vorerst nur die wichtigsten funktionalen Anforderungen umgesetzt werden (mit ++ bewertet). Allerdings sollen alle nichtfunktionalen Anforderungen erfüllt werden.

Funktionen:

- Projektverwaltung: Es können Projekte angelegt werden, in denen unterschiedliche Modelle der GIGABOX projektiert sind und die zugehörigen Skripte+Include-Files
- Texteditor
  - mehrere Skripte lassen sich parallel über Tabs öffnen
  - Copy/Paste
  - Suchen/Ersetzen
  - Möglichkeit, letzte Schritte rückgängig zu machen

- Bei der Codeentwicklung unterstützende Funktionen. Ziel: Schnelles und effektives Schreiben von fehlerfreiem Code
- Erkennung von GIGABOXEN, die mit dem PC verbunden sind
- PAWN-Compiler
- Beschreiben/Auslesen des Flash-Speichers der GIGABOX
- Konsole
- Bereitstellen von Dokus

Vorteil:

- Tool wird einfacher erweiterbar als bisheriger Stand. Dies ermöglicht es, zukünftig einfacher neue Features zu integrieren.
- Besser wartbar: Bugs können in Zukunft besser behoben werden, da Verhalten des neuen Codes besser bekannt ist
- Usability kann verbessert werden
- Tool konsistent mit WPF realisiert unter aktueller .NET Version 4.6
- Keine lange Einarbeitung in alten Code nötig

Nachteil:

- In Entwurf der neuen Softwarearchitektur muss Zeit investiert werden
- Tool bietet keinen direkten Mehrwert in Form von höherem Funktionsumfang
- Insgesamt höherer Aufwand um die gestellten funktionalen Anforderungen zu erreichen

# Kapitel 7

## Design

### 7.1 Entwurf der Benutzeroberfläche

Entwurf mit Pencil hier einfügen

Erläuterung des Entwurfes mit Bezugnahme auf ISO, die gute Usability definiert

### 7.2 Entwurf der Softwarearchitektur

[2]

Eine Softwarearchitektur definiert die Komponenten eines Systems, beschreibt deren wesentliche Merkmale und charakterisiert die Beziehungen dieser Komponenten. Sie beschreibt den statischen Aufbau einer Software im Sinne eines Bauplans und den dynamischen Ablauf einer Software im Sinne eines Ablaufplans.

Die in diesem Kapitel vorgestellte Software-Architektur des configurAIDERS wurde entworfen auf Basis der funktionalen und nichtfunktionalen Anforderungen an Software in Kapitel 3.

In Abbildung XX wird der configurAIDER in Kontext zu seiner Umgebung dargestellt. Der configurAIDER wird als Blackbox dargestellt und zeigt die Schnittstellen zu seiner Umgebung. **Im Moment wird er nicht als Blackbox sondern Whitebox dargestellt**

Abbildung

Was ist Softwarearchitektur? IEEE 1471-2000

Architekturmuster: Layer-Architektur: MVVM Bild mit MVVM Model, 3 Schichten

Architektur des configurAIDERS

Module: Rahmenfenster Projektverwaltung

# Abbildungsverzeichnis

2.1	Use-Case-Diagramm GIGABOX . . . . .	4
3.1	Use-Case-Diagramm funktionale Anforderungen . . . . .	17
4.1	Übersicht configurAIDER . . . . .	21
4.2	Fenster „Simuliertes Gerät“ . . . . .	22
4.3	Fenster „Verfügbare Geräte“ . . . . .	23
4.4	Fenster „Konsole“ . . . . .	23
4.5	Fenster „Programmlog“ . . . . .	24
4.6	Fenster „scriptEDITOR“ . . . . .	26
4.7	Fenster „multiEDITOR“ . . . . .	28
4.8	Überblick Funktionen des configurAIDERS . . . . .	29

# Tabellenverzeichnis

# Literaturverzeichnis

- [1] Gerd Beneken Ulrike Hammerschall. *Software Requirements*. Pearson Deutschland GmbH, 2013.
- [2] Gernot Starke. *Effektive Software Architekturen*. Hanser Verlag, 2009.