Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorthesis

Chris Niklas Lucka

Erstellung eines Steuerprogramms für eine Keyless Entry System Demo auf Basis eines 32-bit Mikrocontrollers

Fakultät Technik und Informatik
Department Informations- und Elektrotechnik

Faculty of Engineering and Computer Science
Department of Information and Electrical Engineering

Chris Niklas Lucka

Erstellung eines Steuerprogramms für eine Keyless Entry System Demo auf Basis eines 32-bit Mikrocontrollers

Bachelorthesis eingereicht im Rahmen der Bachelorprüfung
im Studiengang Informations- und Elektrotechnik
am Department Informations- und Elektrotechnik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. Heike Neumann
Zweitgutachter : Dirk Besenbruch (extern)

Abgegeben am 21. Februar 2019

Thema der Bachelorthesis

Erstellung eines Steuerprogramms für eine Keyless Entry System Demo auf Basis eines 32-bit Mikrocontrollers

Stichworte

Embedded C, Rapid Prototyping, Keyless Entry System, GUI, C#

Kurzzusammenfassung

Diese Bachelorthesis beschreibt die Entwicklung einer Passive Keyless Entry System Demonstration. Die Software des Projektes wird auf dem ARM basierten, für die Automobilindustrie zertifizierten 32-bit Mikrocontroller S32K144 der Firma NXP Semiconductors nach dem Vorgehensmodell Rapid Prototyping entworfen. Abschließend wird eine GUI zur Steuerung und Datensicherung eines Modus der Demonstration entwickelt.

Title of the paper

Development of a control program for a keyless entry system demo based on a 32-bit microcontroller

Keywords

embedded C, rapid prototyping, keyless entry system, GUI, C#

Abstract

This bachelor thesis describe the development of a passive keyless entry system demonstration. The software of the project is designed on the ARM based and automotive certified 32-bit microcontroller S32K144 of the company NXP Semiconductors accoring to the procedure model rapid prototyping. Finally, a GUI for controlling and backing up data for a mode of the demonstration is developed.

Danksagung

Hiermit bedanke ich mich bei meinen Kolleginnen und Kollegen von NXP Semiconductors für die Unterstützung während dieser Bachelorarbeit. Vor allem möchte ich mich bei Dirk bedanken, der mir durch diese Bachelorarbeit die Chance gegeben hat, bei NXP Semiconductors zu arbeiten. Über die hervorragende Betreuung von Dirk während der Bachelorarbeit und darüber hinaus, habe ich mich sehr gefreut.

Bedanken möchte ich mich außerdem bei Frau Neumann für die Betreuung seitens der Hochschule. Für Rückfragen und Hilfestellungen hatten Sie immer Zeit und Rat für mich.

Zuletzt möchte ich mich bei meiner Familie und meinen Freunden für die Unterstützung während der Bachelorthesis und des gesamten Studiums bedanken.

Hamburg, 21. Februar 2019

Inhaltsverzeichnis

Tabellenverzeichnis

Listings

Abbildungsverzeichnis

Abkürzungsverzeichnis

Allgemeine Begriffe

ASCII American Standard Code for Information Interchange

ASK Amplitude-Shift Keying

CPHA Clock Phase

CPOL Clock Polarity

CS Chip Select

EEPROM Electrically Erasable Programmable Read-Only Memory

EMI Elektromagnetic Interference

GPIO General-Purpose Input/Output

GUI Graphical User Interface

HAL Hardware Abstraction Layer

IC Integrated Circuit

IDE Integrated Development Environment

IDE Identifier

ISM Industrial, Scientific and Medical

ISR Interrupt Service Routine

LCD Liquid Crystal Display

LF Low Frequency

LPSPI Low Power Serial Peripheral Interface

LPUART Low Power Universial Asynchronous Receiver and Transmitter

MCAL Microcontroller Abstraction Layer

MCU Microcontroller Unit

MISO Master Input Slave Output

MOSI Master Ouput Slave Input

OEM Original Equipment Manufacturer

RDY Ready

SDK Software Development Kit

SDI Serial Data Input

SDO Serial Data Output

SPI Serial Peripheral Interface

SRD Shrot Range Devices

UART Universial Asynchronous Receiver and Transmitter

UHF Ultra High Frquency

NXP spezifiesche Begriffe

IDE Identifier

JOKER Joint Keyless Entry and Receiver IC

PKE Passive Keyless Entry

PKG Passive Key Go

RCI Remote Control Interface

RKE Remote Key Entry

RSSI Received Signal Strength Indicator

TED-Kit Transponder Evaluation Development - Kit

WUP ID Wake Up Identification

1. Einleitung

Die Ver- und Entriegelung eines PKWs kann auf vielen Wegen realisiert werden. Der bekannte mechanische Schlüssel wurde bei dieser Aufgabe zum großen Teil durch das RKE[1] System erweitert. Die Funktion des RKE Systems lässt sich mit der folgenden Beschreibung sinngemäß nach [9] erklären.

Unter den RKE Systemen versteht man die Ver- und Entriegelung eines Fahrzeugs über einen Funkschlüssel. Der Nutzer des Systems kann durch das Betätigen der Taster auf dem Schlüssel selbstständig bestimmen, wann er das Fahrzeug auf- oder abschließen möchte. Diese Funktionalität ist nur solange gegeben, wie sich der Fahrzeugschlüssel im Aktionsradius des RKE Systems befindet. Der Nutzer führt bei diesem System für die Ver- und Entriegelung eine aktive Handlung aus.

Eine weitere Systemvariante für den Entriegelvorgang des Fahrzeugs bietet sich mit dem PKE[2] System. Im Folgenden wird dieses sinngemäß nach [9] erläutert.

Mit dem PKE System kann die Ver- und Entriegelung des Fahrezuges voll automatisiert, ohne eine aktive Handlung des Nutzers, durchgeführt werden. Um diese Art von Zutritt zu ermöglichen, muss er lediglich den Schlüssel bei sich tragen. Sobald sich der Nutzer im Aktionsradius des Fahrzeugs befindet und sich in Richtung des Fahrzeugs bewegt, wird dieses voll automatisch entriegelt. Ähnlich läuft die Verriegelung ab, welche beim Entfernen des Nutzers vom Fahrzeug einsetzt.

Die Firma NXP Semiconductors bietet in ihrem Produktportfolio die nötigen Komponenten an, mit denen ein PKE System realisiert werden kann. In der Automobilbranche findet man NXP Semiconductors, wie in Abbildung 1.1 zu sehen, als Komponentenlieferant wieder. Zu den Kunden von NXP Semiconductors zählen zahlreiche Modullieferanten, welche die Schnittstelle zwischen den Komponentenlieferanten und den Automobilherstellern, also den OEM[3] bilden. NXP Semiconductors produziert ICs[4] für verschiedenste Module, zu denen unter anderem das eben erwähnte PKE System gehört.

[1] **R**emote **K**ey **E**ntry, engl. für Zugangssystem zum Fahrzeug über eine Fernbedienung
[2] **P**assive **K**eyless **E**ntry, engl. für Automatisiertes Zugangssystem zum Fahrzeug
[3] **O**riginal **E**quipment **M**anufacturer, engl. für Erstausrüster - hier Fahrzeughersteller
[4] **I**ntegrated **C**ircuit, engl. für Integrierter Schaltkreis

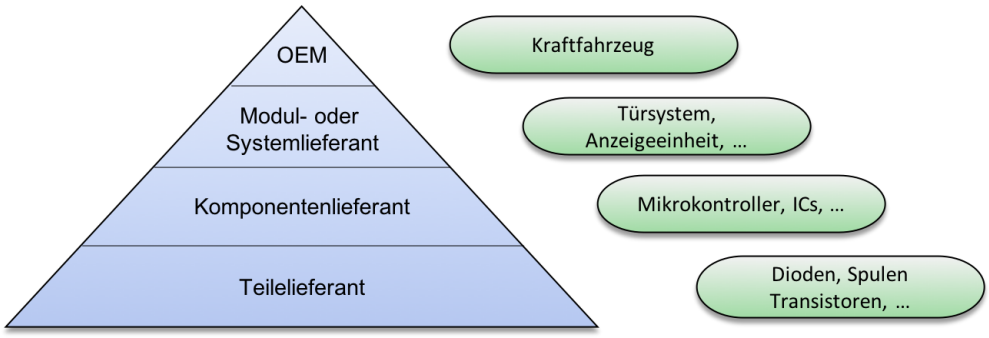


Abbildung 1.1.: Zulieferpyramide der Automobilbranche, modifiziert nach [1]

Für die Präsentation des PKE Systems vor den Modullieferanten, benötigt NXP Semiconductors einen Demonstrationsaufbau. Dieser soll dazu dienen, das PKE System von NXP Semiconductors vorzustellen und technische Informationen über die Produkte zu vermitteln. Ebenfalls sollen unterschiedlichste Schlüssel-ICs, aus dem Produktportfolio von NXP Semiconductors, im direkten Vergleich zueinander vorgestellt werden können. Beispielhaft soll der Demonstrationsaufbau auf einem für die Automobilindustrie zertifizierten Mikrocontroller implementiert werden. Dem Modullieferanten/ Kunden soll durch die Demonstration bewusst werden, dass das System von NXP Semiconductors direkt auf einem Mikrocontroller im Fahrzeug implementiert werden kann. Das Gesamtmodul Zutrittskontrolle wird vom Modullieferanten entworfen, daher ist das implementierte PKE System auf einem Mikrocontroller eine geeignete Demonstration für Präsentationen vor dem Kunden. Der Aufbau soll ebenfalls die Zuverlässigkeit und Reichweite des Systems repräsentieren.

Das Thema dieser Bachelorarbeit beschreibt sich daher über den Titel *Erstellung eines Steuerprogramms für eine Keyless Entry System Demo auf Basis eines 32-Bit Mikrocontrollers*. Thematisch ist diese Bachelorarbeit in sieben Kapitel unterteilt. Beginnend werden die technischen Grundlagen des PKE Systems von NXP Semiconductros erläutert. Weitergehend wird die Aufgabenstellung dieser Bachelorarbeit definiert. Es folgt ein Überblick über die Projektplanung und die Projektdurchführung, in der unter anderem die Strukturen des Quelltextes vom Mikrocontroller erklärt werden. Für die zuverlässige Funktionalität der Demonstration wird das Testdokument für ein erfolgreiches Softwareprodukt beschrieben. Abschließend folgt ein Fazit mit einem Einblick in die weiteren Modifikations- und Einsatzmöglichkeiten des Aufbaus.

2. Technische Grundlagen

Im folgenden Kapitel wird die Funktionsweise des PKE Systems von NXP Semiconductors technisch erläutert. Zunächst wird die Funktion des Gesamtsystems betrachtet. Im Anschluss daran werden die einzelnen Module detallierter beschrieben. Ebenfalls wird ein Mikrocontroller und das sogenannte RSSI-Measurement Setup vorgestellt.

2.1. Das PKE System

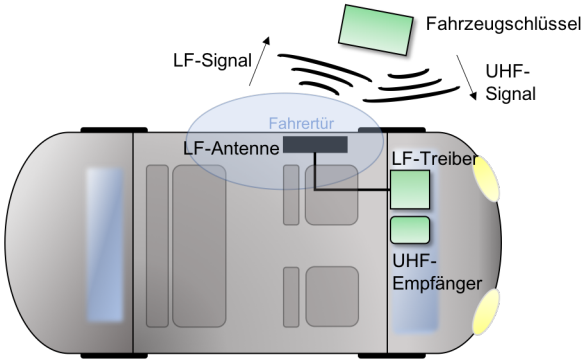


Abbildung 2.1.: Funktionsschema eines PKE Systems, modifiziert nach [2]

Mit dem PKE System wird einem Fahrzeuginhaber ein automatisiertes Ver- und Entriegelsystem angeboten. Der Nutzer des PKE Systems kann sich in Richtung seines Fahrzeugs bewegen, während dieses voll automatisch entriegelt wird, sobald der Schlüssel in einem vordefinierten Bereich des Fahrzeuges befindet[5]. Das PKE System kann selbstständig die Entfernung des Schlüssels zum Fahrzeug durch eine sogenannte RSSI-Messung[6] ermitteln, welche im kommenden Unterabschnitt 2.1.1 erklärt wird. Sobald sich der Schlüssel im vordefinierten Bereich befindet und der Nutzer am Türgriff zieht, wird das Fahrzeug entriegelt. Analog zum Entriegelvorgang wird das Fahrzeug automatisch verriegelt, sobald der Schlüssel den vordefinierten Bereich verlässt. In Abbildung 2.1 ist der Fahrzeugschlüssel zu erkennen. Auf diesem wird die Feldstärkemessung des 125kHz niederfrequenten Feldes durchgeführt.

[5] nach Thatcham Anforderungsprofil
[6] **R**eceived **S**ignal **S**trength **I**ndikator Measurement, engl. für Feldstärkemessung

2.1.1. Die Grundlagen der RSSI-Messung

Um die Entfernung des Schlüssels zum Fahrzeug festzustellen, verwendet NXP Semiconductors ein Messverfahren über die magnetische Feldstärke. Für dieses Messverfahren sind die Gesetze von Biot-Savart ausschlaggebend.

Ein stromdurchflossener Leiter hat ein magnetisches Feld zur Folge. Eine stromdurchflossene Spule wiederum, bildet einen Magneten mit Nord und Südpol aus. Bei der Spule summieren sich die Magnetfeldlinien der einzelnen stromdurchflossenen Leiter auf, wodurch ein resultierendes Magnetfeld die Folge des Stromflusses in der Spule ist, sinngemäß nach [10].

Über das Verfahren einer stromdurchflossenen Spule lässt sich ein Messaufbau für die Ermittlung der Entfernung realisieren. Für den Messaufbau benötigt man eine Sende- und eine Messspule.

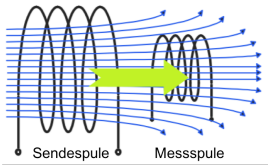


Abbildung 2.2.: Funktionsschema der RSSI Messung, modifiziert nach [3]

Durch die Sendespule (links in Abbildung 2.2 zu sehen) fließt ein kontinuierlicher Strom, welcher ein gleichbleibendes Magnetfeld zur Folge hat. Das Magnetfeld der Sendespule hat, wie bei einem Transformator, einen Stromfluss in der Messspule zur Folge, solange diese an einem geschlossenen Stromkreis angeschlossen ist. Durch den linearen Zusammenhang zwischen magnetischer Feldstärke und der Spannung über der Messspule, kann die magnetische Feldstärke bestimmt werden. Nach den Gesetzen von Biot-Savart lässt sich ein exponentieller Abfall der magnetischen Feldstärke bei steigender Entfernung der Messspule zur Sendespule herleiten. Es ergibt sich $H(x) = \frac{1}{x^3}$, wobei x für die Entfernung der Messspule zur Sendespule steht, sinngemäß nach [10, S. 25]. Bei der RSSI-Messung werden diese Grundlagen zur Entfernungsmessung verwendet. Da im Rahmen dieser Bachelorarbeit die weiteren Details der Messwertermittlung als gegeben angenommen werden, muss die RSSI-Messung nicht detallierter betrachtet werden. Zusammenfassend kann mit dieser Methodik über eine Sendespule mit konstantem Stromfluss eine Spannung in einer Messspule induziert werden, welche Informationen zum Abstand der beiden Spulen zueinander liefert.

NXP Semiconductors verwendet hierfür ein 125kHz- Sendesignal, auch LF[7]-Signal genannt. In Analogie zu diesem Messaufbau steht Abbildung 2.1. Die Sendespule aus Abbildung 2.2 wird durch die LF-Antenne im PKE System von NXP Semiconductors realisiert. Die Messspule aus Abbildung 2.2 wird auf dem Fahrzeugschlüssel des PKE Systems umgesetzt, welcher im folgenden Unterkapitel detallierter vorgestellt wird.

[7] **L**ow **F**requency, engl. für Niederfrequenz

2.1.2. Der Fahrzeugschlüssel

Der Fahrzeugschlüssel ist aus modularer Sicht ein Authentifizierungs- und Messwerkzeug für das PKE System.

Abbildung 2.3.: Der Fahrzeugschlüssel als Beispielanwendung

Er besteht aus einer Batterie für die Energieversorgung, einem LF-Teilbereich für den Datenempfang und die messtechnische Ermittlung der Entfernung und einem UHF[8]-Teilbereich, mit dem der Fahrzeugschlüssel die Messdaten zur Steuereinheit zurücksenden kann.

Im LF-Teilbereich wird über eine 3D-Spule das Magnetfeld am Standort des Schlüssels bestimmt. Die Spule ist dreidimensional ausgelegt, damit das Magnetfeld auf jeder Achse im Raum vermessen und lageunabhängig vom Fahrzeugschlüssel ermittelt werden kann. Dadurch kann die Messung in jeder Orientierungslage des Schlüssels durchgeführt werden. Jedoch soll der Fahrzeugschlüssel diese Messung nicht kontinuierlich durchführen, um energiesparend zu arbeiten.

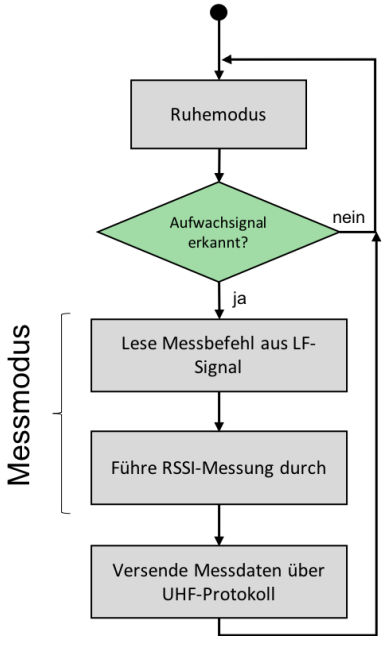


Abbildung 2.4.: Ablaufdiagramm des Fahrzeugschlüssels

[8] **U**ltra **H**igh **F**requency, engl. für Ultrahochfrequenz - ISM Frequenzband SRD (433,05 bis 434,79MHz)

Um ein erfolgreiches Energiemanagement umzusetzen, ist auf dem Schlüssel ein Protokoll festgelegt, mit dem dieser aus dem Ruhemodus in den Messmodus überführt werden kann. Sobald ein LF-Signal mit passendem Protokoll über die 3D-Spule auf der Auswerteeinheit des Schlüssels eintrifft, wechselt der Schlüssel seinen Modus und führt eine Vermessung des Magnetfeldes durch. Der LF-Teilbereich, der im folgenden Unterabschnitt beschrieben wird, stellt während der Messung des Fahrzeugschlüssels ein konstantes Magnetfeld als Messwerkzeug zur Verfügung. Die Vermessung dieses konstanten Magnetfeldes erfolgt dann nacheinander in jeder Teilspule der 3D-Spule. Für die Bestimmung des gesamten Magnetfeldes, wird die geometrische Summe über die Messwerte der drei Teilspulen gebildet. Die ermittelten Daten werden nach einer erfolgreichen Messung zusammengestellt und über ein UHF-Signal versendet.

2.1.3. Der LF-Teilbereich

Mit der LF-Antenne aus Abbildung 2.1 wird eine Sendespule für die RSSI-Messung realisiert. Mit ihr können Informationen auf der physikalischen Übertragungsebene versendet werden. Um alle Daten über die LF-Antenne zu versenden, müssen diese über den Antennenstrom modelliert werden. Für diese Modellierung wurde der LF-Treiber JOKER[9] entworfen.

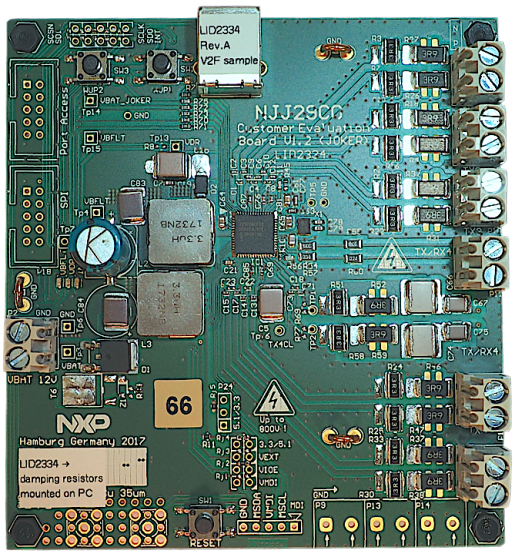


Abbildung 2.5.: Der LF-Treiber JOKER

Die folgende Komponentenbeschreibung ist sinngemäß aus [2] übernommen.
Der JOKER wird über 12V Gleichspannung betrieben. Er bietet sechs Vollbrücken-LF-Treiberkanäle, welche über die angeschlossenen Antennen das 125kHz-Signal aussen-

[9] **JO**int **K**eyless **E**ntry and **R**eceiver Integrated Circuit

den. Die Antennen müssen so im Fahrzeug platziert sein, dass der gesamte Fahrzeuginnenraum mit dem 125kHz-Signal ausgeleuchtet wird. Beispielsweise können diese in der Fahrertür, der Kofferraumtür oder auch im Innenraum des Fahrzeuges untergebracht sein. Der Antennenstrom und somit die Signalstärke des LF-Signals[10] kann über die Parametrierung des JOKERs variiert werden, wobei ein maximaler Antennenstrom von 1 Ampere-Spitze eingestellt werden kann. Die Modellierung der Daten auf den Antennenstrom erfolgt über eine ASK[11].

Eine Teilaufgabe des JOKERs ist die Überführung des Schlüssels in den Messmodus. Hierbei wird ein definiertes Protokoll verwendet, welches sowohl beim JOKER als auch beim Schlüssel durch die Ansteuerung in der Software implementiert sein muss.

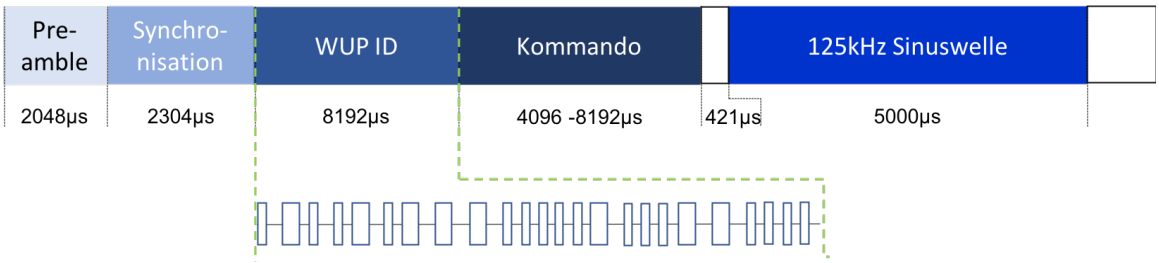


Abbildung 2.6.: LF-Signal , übersetzt und modifiziert nach [4]

In Abbildung 2.6 findet man zwei wichtige Informationen zum LF-Protokoll. Zum Einen sind im unteren Teil die modellierten Daten auf binärer Ebene beispielhaft dargestellt. Die Daten sind in der Manchestercodierung modelliert. Zum Anderen ist im oberen Bereich das Protokoll zu erkennen, welches aus fünf Abschnitten besteht. Die ersten drei Abschnitte beschreiben das Protokoll bis zur Überführung des Schlüssels in den Messmodus. Es beginnt mit der *Preamble*, welche dem Schlüssel Informationen zur Signalstärke liefert. Der Dekoder des Schlüssels kann sich auf diese Signalstärke automatisch einstellen. Darauffolgend wird die Sychronisation für die Taktung des Emfängers auf das LF-Signal übertragen. Mit der *WUP ID*[12] wird das im Unterabschnitt 2.1.2 erwähnte Weckmuster übermittelt. Empfängt der Schlüssel bis zu diesem Zeitpunkt das LF-Signal, wird er in den Messmodus überführt und verarbeitet alle weiteren Daten.
Im zweiten Teilbereich des JOKERs findet man die Übermittlung der Kommandos zum Schlüssel. Mit dem Kommando können dem Schlüssel unterschiedliche Aufgaben aufgetragen werden.
Im letzten Bereich des LF-Signals findet man den Begriff *Sinuswelle*, welche das konstante Magnetfeld für die RSSI-Messung auf dem Schlüssel beschreibt. Das konstante

[10] Das LF-Signal wird im weiteren Verlauf teilweise als PKE-Loop bezeichnet.
[11] **A**mplitude-**S**hift **K**eying - digitale Modulationsart bei der unterschiedliche Spannungspegel ein binäres Signal repräsentieren
[12] **W**ake **U**p **P**attern **ID**entification, engl. für Aufweckmuster Identifikation

Magnetfeld hat eine definierte Zeitspanne. In dieser Zeitspanne muss der Schlüssel die RSSI-Messung durchgeführt haben.

Der JOKER dient aus der Systemsicht als Übersetzer. Er ist über das Bussystem SPI[13] an einem Hostsystem, wie beispielswiese einem PC oder einem Mikrocontroller, angeschlossen. Über dieses Bussystem erhält das Modul Kommandos, welche unter anderem die Parametrierung oder Anweisungen für das Modul enthalten. Nach der Parametrierung des JOKERs weiß dieser, in welchem Rahmen er die Daten verpacken, modellieren und versenden muss. Aus einer SPI-Anweisung formt der JOKER in Kombination mit einer LF-Antenne schlicht gesagt ein übersetzes LF-Signal.

2.1.4. Der UHF-Teilbereich

Aufgenommene Messdaten kann der Fahrzeugschlüssel beispielsweise über eine Frequenz von 433,92MHz versenden. Um der Fahrzeugelektronik diese Messdaten zur Verfügung zu stellen, benötigt das PKE System einen UHF-Empfänger. Der Lizard ist einer dieser UHF-Empfänger aus dem Produktportfolio von NXP Semiconductors.

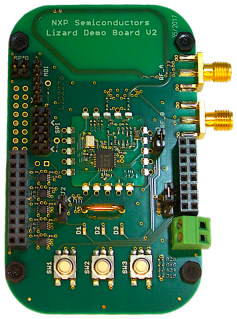


Abbildung 2.7.: Lizard - NCK2910

Mit dem Lizard können die beschriebenen Messdaten aus Unterabschnitt 2.1.2 empfangen werden. Ähnlich wie der JOKER, gilt auch der Lizard als Übersetzungsglied, nur formt dieser SPI-Signale aus den über UHF empfangenen Daten für ein Hostsystem.

Der Empfänger muss für eine erfolgreiche UHF-Kommunikation die gleiche UHF-Parametrierung wie der Fahrzeugschlüssel erhalten. Parametrierbar ist das Gerät ebenfalls über die SPI-Kommunikation mit einem Hostsystem. Parametereigenschaften sind unter anderem die maximale Datenlänge, die Datenrate, das Modulationsverfahren oder auch die Sendefrequenz (hier 433,92MHz).

[13] **S**erial **P**eripheral **I**nterface, engl. für Serielle Schnittstelle

2.1.5. Zusammenfassung des PKE Systems

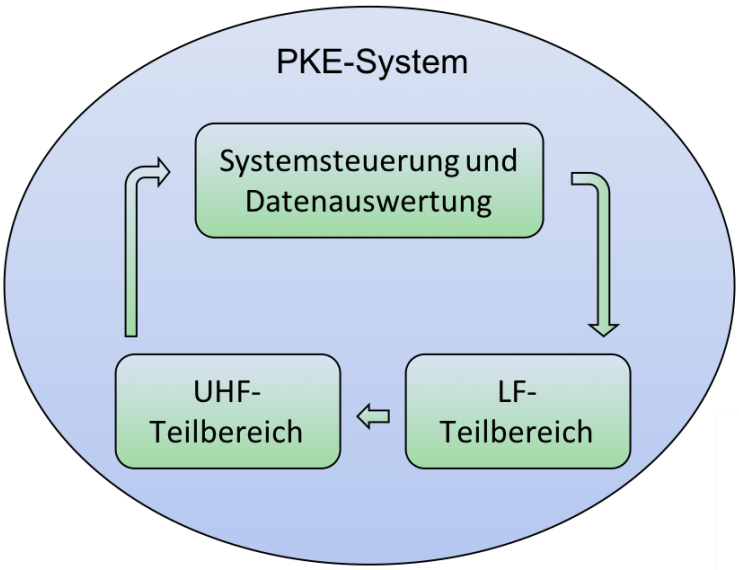


Abbildung 2.8.: Die Teilbereiche eines PKE Systems

Zurück zur Vogelperspektive ist die Aufteilung der Teilbereiche wie in Abbildung 2.8 zu sehen. Mit dem LF-Teilbereich werden Messdaten zur Entfernungsermittlung aufgenommen. Außerdem kann der Fahrzeugschlüssel durch den Ruhemodus energieeffizient betrieben werden. Über den UHF-Teilbereich können die Messdaten an die Auswerteeinheit übertragen werden. Im dritten Teilbereich findet sich das Hostsystem wieder, auf dem die Systemsteuerung und Entfernungsermittlung/ Datenauswertung implementiert ist. Das PKE-System kann als zyklisches System betrachtet werden. Das konstante Magnetfeld wird zyklisch bereitgestellt. Sobald der Schlüssel in Reichweite des Fahrzeugs ist, wird die RSSI-Messung durchgeführt. Anschließend werden die Daten zum Hostsystem versendet. Durch die Messwertsammlung der einzelnen Zyklen kann der Fahrzeugzugang gesteuert werden.
Im folgenden Unterabschnitt wird ein Hostsystem vorgestellt, auf dem der Teilbereich *Systemsteuerung und Datenauswertung* umgesetzt werden kann.

2.2. Der Mikrocontroller S32K144

Mit dem S32K144 Mikrocontroller bietet sich ein Hostsystem zur Steuerung eines PKE Systems.

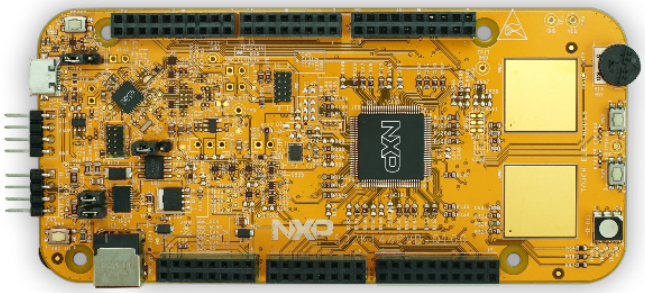


Abbildung 2.9.: Das *S32K144* EVB-0100 - S32K144 Evaluierungsboard,[5]

Das S32K144 Evaluation Board beinhaltet den 32-Bit Mikrocontroller S32K144 der Firma NXP Semiconductors aus der Produktreihe S32K. Die Produktreihe basiert auf der Arm Cortex Technologie. Das für die Automobilindustrie zertifizierte S32K144EVB-0100 aus Abbildung 2.9 verfügt über zwei Taster, eine RGB-LED, zwei Berührungssensoren und einem Potentiometer. Der Mikrocontroller beinhaltet eine OpenSDA[14]-USB Schnittstelle und eine OpenSDA MCU[15]. Diese Systemteile stellen eine Brücke zwischen Hostcomputer und Zielprozessor dar. Somit wird das Debuggen, die Flash-Programmierung und die serielle Kommunikation mit dem Mikrocontroller ermöglicht. Mit dem S32 Design Studio bietet sich eine IDE[16] über die sich der Mikrocontroller programmieren, kompilieren und debuggen lässt. Die Entwicklungsumgebung ist ähnlich wie eine Eclipse IDE aufgebaut.

[14] **Open** Standard **S**erial and **D**ebug **A**dapter
[15] **M**icro**C**ontroller **U**nit, engl. für Mikrocontroller Einheit
[16] **I**ntegrated **D**evelopment **E**nvironment, engl. für Integrierte Entwicklungsumgebung

2.3. Das RSSI-Measurement Setup

NXP Semiconductors bietet unterschiedliche Schlüssel-ICs in ihrem Produktportfolio an. Um dem Modullieferanten einen detallierten Vergleich der einzelnen Produkte zu ermöglichen, wurde das RSSI-Measurement Setup entwickelt. Dieses beinhaltet schon bekannte Komponenten. Unter anderem ist hier der LF-Treiber JOKER zu erkennen und die Fahrzeugschlüssel aus Unterabschnitt 2.1.2 sind zu sehen.

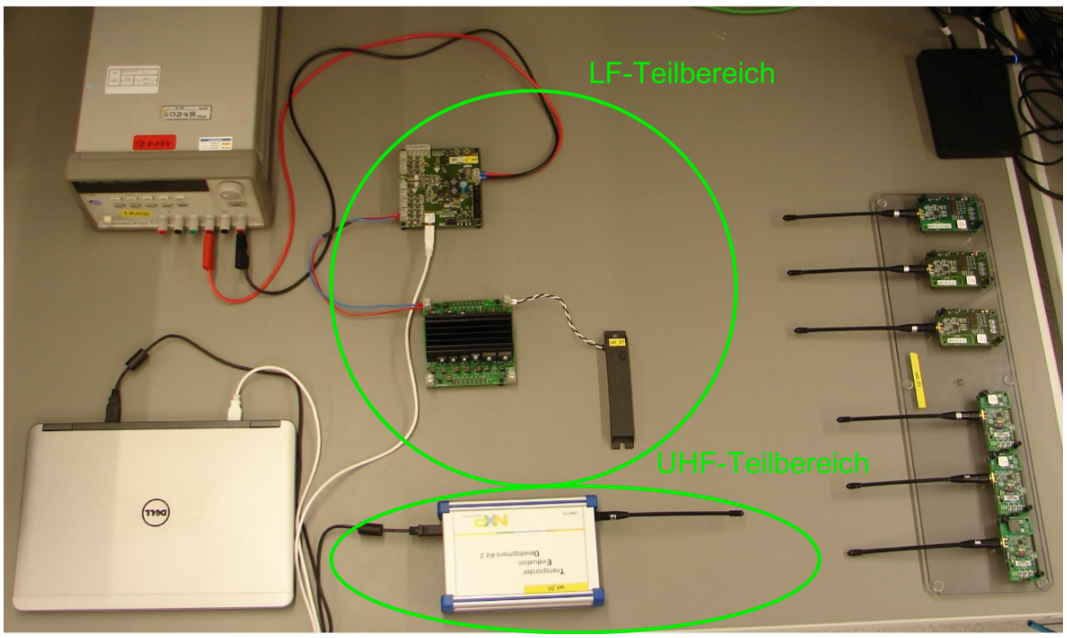


Abbildung 2.10.: RSSI-Measurement Setup, modifiziert nach [4]

Bei diesem Aufbau sind auf der rechten Seite der Abbildung 2.10 sechs Fahrzeugschlüssel in unterschiedlichen Konfigurationen für die Vergleichsmessung zu erkennen. Alle integrierten Schlüssel des Aufbaus unterscheiden sich durch die Controller-Architektur oder die Hardwareauslegung voneinander.

Schematisch ähnelt das Setup dem Aufbau aus Abbildung 2.1. Im LF-Teilbereich von Abbildung 2.10 ist der LF-Treiber JOKER und eine angeschlossene LF-Antenne (mittig im Bild) aufgebaut. Oben links in Abbildung 2.10 ist ein Netzteil für die Spannungsversorgung des LF-Treibers zu sehen. Das TED-Kit 2[17] (unten mittig in Abbildung 2.10) dient als UHF-Empfänger und wird in dieser Bachelorarbeit durch den Lizard ersetzt. Die softwareseitige Implementation des Aufbaus wurde auf einem Laptop vorgenommen. Dieser ist im RSSI-Measurement Setup somit das Hostsystem, auf dem die PKE Applikation implementiert ist.
Alle Schlüssel-ICs führen zur gleichen Zeit eine RSSI-Messung des selben Magnetfeldes durch, wodurch ein Messwertvergleich der einzelnen Fahrzeugschlüssel nach der

[17] **T**ransponder **E**valuation **D**evelopment - Kit 2

Messung möglich ist. Mit dem TED-Kit 2 werden ähnlich wie in Unterabschnitt 2.1.4 erklärt, die Messdaten aller Fahrzeugschlüssel eingesammelt und dem Hostsystem zur Verfügung gestellt. Alle Daten der Schlüssel werden für einen Messdurchlauf in ein Textdokument formatiert abgelegt. Im Nachgang können die Messdaten durch ein Matlab-Skript ausgewertet werden. Mit diesem Skript können die Messergebnisse über Grafiken verbildlicht werden.

Im Endeffekt bietet das RSSI-Measurement-Setup eine Übersicht über die Erreichbarkeit und die Messwerte des Fahrzeugschlüssels. Ebenfalls ist das System portabel und bietet somit die Möglichkeit der direkten Schlüsselauswertung am Kundenfahrzeug. Dem Kunden können durch diesen Messaufbau somit Messergebnisse für die verschiedensten Schlüsselkonfigurationen bereitgestellt werden.

3. Beschreibung der Aufgabenstellung

In diesem Kapitel wird das Ziel der Bachelorarbeit erläutert. Für die Gliederung der Anforderungen werden einige Namensgebungen festgelegt. Auf die Anforderungen des Projektes wird ebenfalls eingegangen. Es kommen die Komponenten aus Kapitel 2 zum Einsatz, welche zum Teil durch verbesserte Komponenten ersetzt werden.

3.1. Ziel des Projektes

Wie schon in Unterabschnitt 2.1.2 beschrieben, bietet NXP Semiconductors in ihrem Produktportfolio verschiedene Schlüssel-IC's an. Durch die verschiedenen Hardwarearchitekturen der Schlüssel bieten diese verschiedene Messgenauigkeiten und eine unterschiedlich ausgeprägte EMI[18]. Eine hohe Messgenauigkeit und Störfestigkeit spiegeln sich auch in einem höheren Stückpreis der Schlüssel-IC's wieder. Die Modullieferanten/Kunden müssen daher eine Entscheidung bei der Produktauswahl treffen. Für den Entscheidungsprozess des Kunden soll eine Demonstration des PKE Systems erstellt werden.

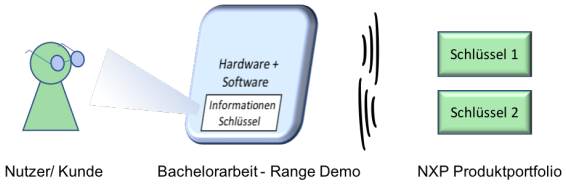


Abbildung 3.1.: Skizze - Projektziel

Mit der Demonstration soll das Produktportfolio von NXP Semiconductors präsentiert werden. Dazu gehören die zuvor beschriebenen Komponenten JOKER, der Lizard und die Fahrzeugschlüssel Token, Token Plus und Token SRX. Dem Nutzer/Kunden soll eine visuelle Anzeige der Messdaten des Schlüssels geboten werden. Es sollen mindestens zwei Schlüsseldaten gleichzeitig auf dieser Anzeige visualisiert werden, um einen Produktvergleich der einzelnen Schlüssel-IC's für den Kunden zu ermöglichen. Damit der Kunde unterschiedlichste Anwendungsfälle der Systemkomponenten testen

[18] **E**lectromagnetic **I**nterference, engl. für Störfestigkeit beispielsweise gegenüber magnetischer Strahlung durch kabelloses Laden eines Elektrofahrzeuges

kann, müssen die Komponenten der Demonstration parametrierbar sein. Die Demonstration soll portabel sein, wodurch eine direkte Anwendung des Systems am Kundenfahrzeug möglich wäre. Abschließend soll dem Kunden mit diesem Demonstrator ein Beispiel für eine mögliche Implementierung im eigenen System geboten werden.

3.2. Termini

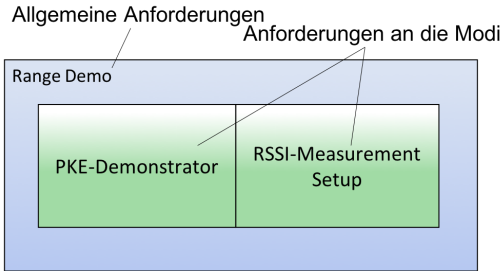


Abbildung 3.2.: Kategorisierung des Projektes

Das Gesamtprojekt wird mit dem Namen **Range Demo** betitelt und beinhaltet zwei unterschiedliche Modi, die auch in Abbildung 3.2 dargestellt sind:

- **Modus 1 - PKE-Demonstrator**
  -> Bezeichnet das präsentationsfähige PKE System
  -> Kunde bekommt einzelne Messinformationen der Fahrzeugschlüssel über Display
  -> Geeignet für Systempräsentationen auf Ausstellungen
- **Modus 2 - RSSI-Measurement Setup**
  -> Bezeichnet die Messdatenaufzeichnung des PKE Systems auf einem PC/ Laptop
  -> Detallierte Schlüsselanalyse möglich
  -> Geeignet für Laborversuche oder Detailpräsentationen vor dem Kunden

In der Anforderungsdokumentation sind alle allgemeinen Anforderungen unter der Kategorie *Range Demo* zu finden. Die Bereiche *PKE-Demonstrator* und *RSSI-Measurement Setup* bezeichnen die verschiedenen Modi, in denen die Range Demo betrieben werden kann. Softwareseitig stellen sie unterschiedliche Anforderungen dar und müssen daher getrennt voneinander betrachtet werden. Hardwareseitig sind die Anforderungen an die Modi gleich, wodurch diese unter den allgemeinen Anforderungen in der Kategorie *Range Demo* zu finden sind.

3.3. Anforderungen an das Projekt

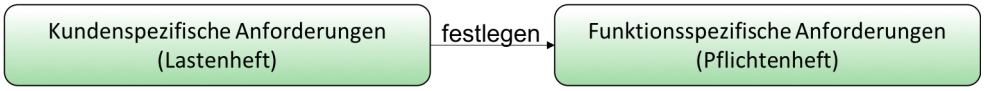


Abbildung 3.3.: Aufbau der Anforderungsdokumentation

Als Aufgabenstellung wurde ein Lastenheft (siehe Anhang A) ausgehändigt. Die erste Aufgabe dieser Bachelorarbeit war die Erstellung eines Pflichtenheftes (siehe Anhang B), mit dem der Rahmen der Bachelorarbeit definiert wurde. Das Pflichtenheft wurde aus den Anforderungen des Lastenheftes abgeleitet. In dem Pflichtenheft findet man die funktionsspezifischen Anforderungen an das Projekt.

Im Folgenden werden die wichtigen Anforderungen aus der Dokumentation erwähnt. Für eine detallierte Übersicht sei weiterhin auf Anhang A und B verwiesen.

3.3.1. Range Demo - Allgemein

NXP Semiconductors hat für das Projekt folgende Hardware zur Verfügung gestellt:

- Mikrocontroller S32K144 (Abschnitt 2.2)
- LF-Treiber JOKER (Unterabschnitt 2.1.3)
- UHF-Empfänger Lizard (Unterabschnitt 2.1.4)
- Fahrzeugschlüssel Token, Token Plus und Token SRX (Unterabschnitt 2.1.2)

Außerdem muss die Range Demo für Transporte zu Ausstellungen oder zum Kunden robust sein. Daher muss die Hardware auf einer Grundplatte befestigt und mit einer Schutzplatte abgedeckt werden.

Softwareseitig muss eine Menüführung im Mikrocontroller implementiert sein. Diese muss die einzelnen Modi voneinander trennen. Ebenfalls muss letzteres die Schnittstellen zum Benutzer separieren. Während das RSSI-Measurement Setup immer über eine GUI[19] auf einem Laptop/ PC bedient wird, muss der PKE-Demonstrator über verschiedene Schnittstellen bedienbar sein.

[19] **G**raphical **U**ser **I**nterface, engl. für graphische Nutzerschnittstelle

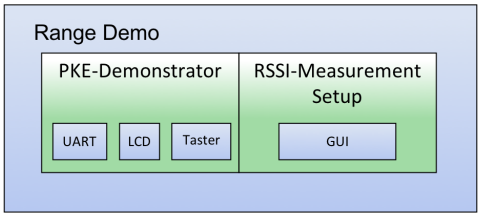


Abbildung 3.4.: Definition der Schnittstellen zum Nutzer

In Abbildung 3.4 ist zu sehen, dass dem Nutzer mehrere Schnittstellen zur Bedienung der Range Demo geboten werden müssen. Der Bediener soll diese selbst auswählen können. Die Schnittstellen der einzelnen Modi werden in dem Unterabschnitt 3.3.2 und 3.3.3 beschrieben.

3.3.2. PKE-Demonstrator - Modus 1

Der PKE-Demonstrator muss dem Nutzer folgende Schnittstellen bieten:

- **Taster** (Eingabe)
  -> Einmaliger Start einer RSSI-Messung möglich
- **UART**[20] (Eingabe)
  -> Einmaliger Start einer RSSI-Messung möglich
  -> Endlosmessung kann gestartet und beendet werden
  -> Eingegebene Anzahl an Messungen wird durchgeführt
- **LCD** (Ausgabe)
  -> Der Messwert der IC Eingangsspannung wird angezeigt
  -> Die Kennziffer (IDE[21]) des Fahrzeugschlüssels wird angezeigt
  -> Darstellung der Aufwachrate des Fahrzeugschlüssels über einen Zähler
  -> Der Messwert des Magnetfeldes wird in einer linearen Darstellung visualisiert (Wunschanforderung)

Im Modus 1 wird das PKE-System präsentiert. Dieser Modus wird vorallem auf Ausstellungen verwendet werden, damit das PKE System mit einem übersichtlichen Beispiel vorgestellt werden kann. Für den schnellen Aufbau des Demonstrators muss dieser ohne ein Laptop/ PC bedienbar sein. Hierfür muss ein Taster verwendet werden,

[20] **U**niversal **A**syncronous **R**eceived **T**ransmitter, engl. für universelles asynchrones Empfangen und Versenden
[21] **IDE**ntifier, engl. für Kennung des Fahrzeugschlüssels

über den die automatische Initialisierung aller Systemkomponenten mit Standardwerten gestartet wird. Es muss ein einmaliges LF-Signal gesendet werden. Das UHF-Signal vom Fahrzeugschlüssel muss empfangen und die Daten im LCD dargestellt werden. Ebenfalls muss eine UART-Schnittstelle des PKE-Demonstrators eingerichtet werden. Diese wird mit einer USB-Verbindung zwischen dem S32K144 und einem Laptop realisiert. Mit diesem Laptop müssen über die Eingabe in einem Terminal-Emulator Kommandos an die Demonstration gesendet werden können, um eine RSSI-Messung zu starten. Die Möglichkeiten zur Messdurchführung sind unter dem Stichpunkt *UART* angegeben.

3.3.3. RSSI-Measurement Setup - Modus 2

Der Modus 2 *RSSI-Measurement Setup* muss dem Bediener eine GUI als Schnittstelle bieten.

- **GUI** (Eingabe)
  -> Messpunktnummer - Textfeld *Measurement point of grid*
  -> Abstand zwischen Schlüssel und Sendespule - Textfeld *Distance of keys[cm]*
  -> Anzahl der Messungen - Textfeld *Number of Measurements*
  -> Button - *Start RSSI-Measurement*
  -> Parametrierung der auszugebenden Messwerte des Fahrzeugschlüssels
  -> Button - *Save on S32*
  -> Stromstärke der Sendeeinheit - Textfeld *Current*
  -> Anzahl der Schlüssel im Feld - Textfeld *Number of Keys*
- **GUI** (Ausgabe)
  -> Verbunden mit *S32K144_RangeDemo*
  -> Ausgabe der einzelnen Messwerte der Fahrzeugschlüssel in Textdokument
  -> Ausgabe der einzelnen Messwerte der Fahrzeugschlüssel in GUI

Das RSSI-Measurement Setup ist ein weiteres Ziel dieses Projektes. Mit diesem müssen alle Messdaten des Fahrzeugschlüssels formatiert in der GUI ausgegeben werden. Ebenfalls müssen mit der GUI die Messwerte formatiert in einem Textdokument abgelegt werden. Für die Generierung dieses Textdokumentes muss der Bediener die Messpunktnummer, den Abstand zwischen Schlüssel und Sendespule und die Anzahl der Messungen eingeben. Über den Button *Start RSSI-Measurement* muss der Bediener die jeweilige Messung starten können. Mit dem Textdokument können nach einer Messung die Messwerte der Fahrzeugschlüssel ausgewertet werden, damit detalliertere Aussagen über die Messgenauigkeit getroffen werden können.

4. Projektplanung

Im ersten Abschnitt dieses Kapitels wird das Vorgehensmodell *Rapid Prototyping* sinngemäß nach [11, S. 30], nach dem diese Bachelorarbeit erstellt wurde, beschrieben. Darauffolgend wird die Projektplanung mit diesem Vorgehensmodell vorgestellt. Zur detallierten Darstellung der Projektplanungsphase werden die Dokumente, wie zum Beispiel die Lösungsskizze, vorgestellt und erläutert.

4.1. Rapid Prototyping

Unter dem Begriff *Rapid Prototyping* versteht man ein Vorgehensmodell für eine schnelle Prototypenentwicklung.

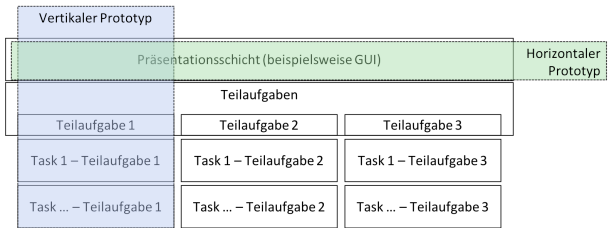


Abbildung 4.1.: Vertikales und horizontales Prototyping, modifiziert nach [6, S. 105]

In Abbildung 4.1 wird eine Beispielstruktur für ein Softwareprojekt mit einer GUI vorgestellt. Das Vorgehensmodell *Rapid Prototyping* bietet dem Projektplaner zwei Möglichkeiten. Das Projekt kann über den horizontalen Prototypen bearbeitet werden. Hierbei kann dem Kunden beispielsweise frühzeitig die Bedieneroberfläche vorgestellt werden. Diese beinhaltet noch keine technische Funktionalität, bietet dem Kunden allerdings die frühzeitige Außensichtweise des zuvor spezifizierten Lastenheftes. Mit dem Kunden können somit im Voraus Missverständnisse geklärt werden. Anders als beim horizontalen Prototypen, wird beim vertikalen Prototypen ein Teilbereich der Aufgaben bis zur vollständigen Funktionalität fertiggestellt. Vorteilhaft ist hierbei die Abschirmung der einzelnen Teilaufgaben zum Gesamtziel. Durch diese Herangehensweise können komplexe Aufgaben in einfachere Schritte aufgeteilt werden. Außerdem bietet dieses Vorgehensmodell ein hohes Maß an Flexibilität unter Berücksichtigung kurzfristiger Änderungen bei der Umsetzung des Projektes. Das vertikale Prototyping eignet sich daher für Projekte, bei denen mehrere Komponenten zu einem Gesamtsystem zusammengeführt werden.

4.2. Projektplanung der Range Demo mit dem Vorgehensmodell Rapid Prototyping

In dieser Bachelorarbeit erfolgte die Implementierung nach dem Prinzip des vertikalen Protoypings.

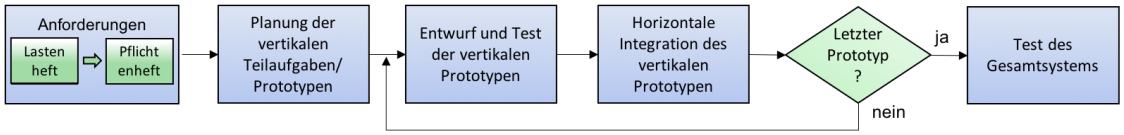


Abbildung 4.2.: Projektablaufplan, modifiziert nach [6, S. 105]

Aus den technischen Anforderungen des Pflichtenheftes wurden die vertikalen Teilaufgaben und somit der Projektplan (siehe Anhang C) erstellt. Nach der Projektplanung folgt der Entwurf der einzelnen vertikalen Prototypen.

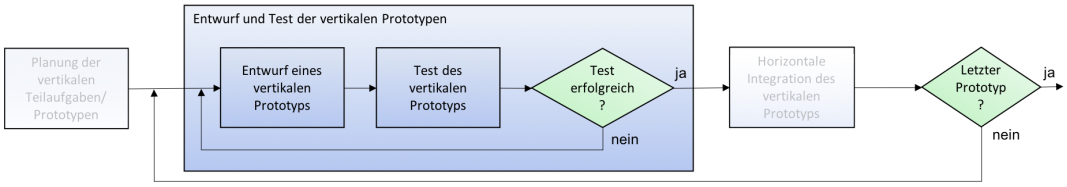


Abbildung 4.3.: Vertikaler Prototypenentwurf, modifiziert nach [6, S. 105]

Sobald der vertikale Prototyp erstellt und ausreichend getestet ist, sodass alle Teilanforderungen an den vertikalen Prototypen erfüllt sind (siehe Anhang B), kann dieser horizontal in das Gesamtprojekt integriert werden. An dieser Stelle sei auf die doppelte Schleife in diesem Vorgehensmodell hingewiesen. Zum Einen befindet sich eine Schleife beim Entwurf des vertikalen Prototypens. Sobald alle Teilanforderungen erfüllt sind, ist der Prototyp fertiggestellt. Die zweite Schleife ist bei der horizontalen Integration zu finden.

Durch diese Heransgehensweise wächst das Gesamtprojekt mit jedem vertikalen Prototypen, allerdings bleiben die Teilaufgaben immer kompakt. Bei der horizontalen Integration können Änderungen an den Schnittstellen am vertikalen Prototypen vorgenommen werden. Sobald der letzte vertikale Prototyp erfolgreich in das Gesamtprojekt integriert wurde, muss das Gesamtsystem getestet werden. Hierzu dient wieder das Pflichtenheft, in dem alle Anforderungen durch das Gesamtsystem erfüllt werden müssen. Die Zeitbemessung dieses Vorgehensmodells wächst sowohl mit der Anzahl der vertikalen Prototypen, als auch mit der Komplexität der vertikalen Prototypen.

4.2.1. Aufteilung der Gesamtaufgabe in die Teilaufgaben

Um die vertikalen Prototypen eindeutiger zu definieren, wurde eine Lösungsskizze des Hardwareaufbaus angefertigt.

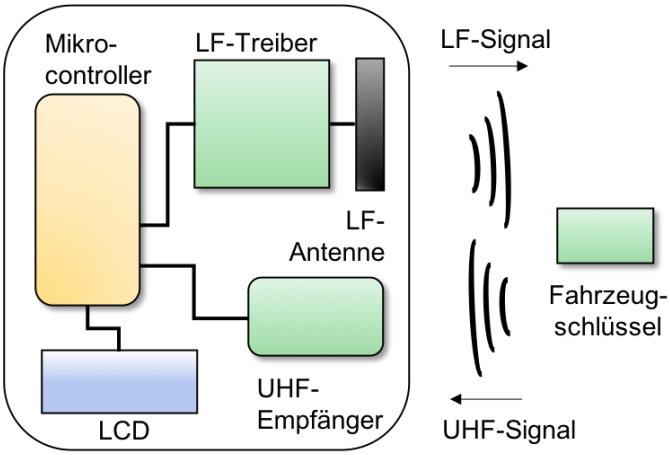


Abbildung 4.4.: Skizze/ Entwurf der Range Demo, modifiziert nach [2]

Über diesen Entwurf konnten die einzelnen Teilaufgaben in den Projektplan überführt werden. Der Projektplan ist in Anhang C zu finden, in dem alle Teilaufgaben zu den einzelnen vertikalen Prototypen beschrieben sind. Als vertikale Prototypen wurden folgende Teilsysteme gewählt:

- LCD an S32K144
- UHF-Empfänger an S32K144
- LF-Treiber an S32K144
- Auswertung der Schlüsseldaten auf S32K144
- Menü und Nutzerführung auf S32K144
- GUI

4.2.2. Festlegungen für die Software

Für die Struktur der Software wurden im Voraus mehrere Richtlinien festgelegt. Zum Einen sollen Komponenten wie der JOKER oder der Lizard softwareseitig durch eine neue Komponente ersetzt werden können. Hierfür sollen Softwaremodule erstellt werden, in der die gesamte Software zur jeweiligen Komponenten (vertikalen Prototypen) zu finden ist. Außerdem wird die Software nach dem folgenden Schichtenmodell entwickelt.

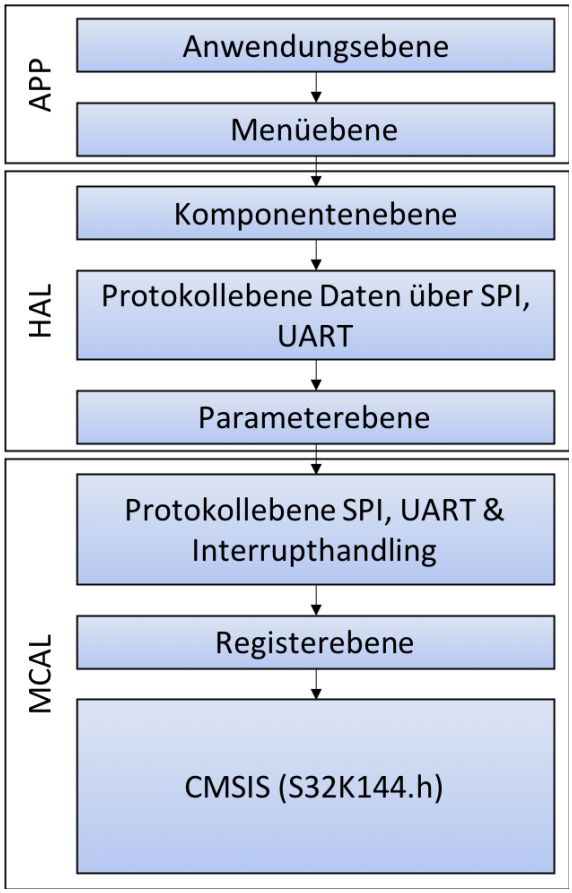


Abbildung 4.5.: Schichtenmodell der Software des S32K144, modifiziert nach [7]

Bei diesem Modell sind die einzelnen Software-Schichten zu sehen. Die Schichten teilen sich in die Kategorien Applikation (APP), HAL[22] und MCAL [23] auf. Jeder einzelne Prototyp soll nach diesem Modell entwickelt werden. Für die Prototypen werden allerdings nur die nötigsten Schichten des Schichtenmodells für die softwaretechnische Realisierung durchlaufen. Durchgriffe zu den einzelnen Ebenen sind von oben nach unten möglich. Ebenfalls sind Durchgriffe auf der gleichen Ebene möglich, damit Protokolle, wie beispielsweise UART, von anderen Protokollen, wie beispielsweise SPI, genutzt werden können.

Funktionen, die in einer Quelldatei nur Intern genutzt werden sollen, müssen über die Headerdatei geschützt werden. Durch diese Strukturierung sollen die Schnittstellenfunktionen zu den einzelnen Softwareprodukten eindeutig sein. Schnittstellenfunktionen werden daher in der Headerdatei deklariert, wobei interne Funktionen in der C-Quelldatei deklariert werden. Die Struktur aus Abbildung 4.5 ist im Anhang D wiederzufinden.

[22] **H**ardware **A**bstraction **L**ayer, engl. für Hardware Abstraktionsschicht
[23] **M**icro**c**ontroller **A**bstraction **L**ayer, engl. für Mikrocontroller Abstraktionsschicht

Namensgebung

Es wurde zudem eine einheitliche Namensgebung festgelegt, mit der die Funktion über den Funktionsnamen direkt einem C-Quelldatei zugeordnet werden kann. Jeder Ordner oder Funktionsname beginnt mit dem Ausdruck *nxpLFCAS_S32K144*. In diesem ist die Firma, die Abteilung und der Mikrokontroller auf dem die Software implementiert werden soll, hinterlegt. Auch der Projektname ist in dieser Bezeichnung zu finden. Weiterführend ist die Komponente ein Bestandteil des Funktionsnamens. Als Beispiel für die Komponente JOKER wäre letzteres *nxpLFCAS_S32K144_jk*. Hinter einem weiteren Unterstrich befindet sich der Funktionsname. Die Bezeichnungen der Funktionen werden in den folgenden Kapiteln auf *nxp..._Unterordner_Funktionsname* verkürzt.

Tabelle 4.1.: Informationen zum Anhang

Inhalt	**Verknüpfung**
Anforderungsdokumentation	Anhang A und B
Projektplan	Anhang C
Programmhierachieplan	Anhang D

5. Beschreibung der Erstellung der Range Demo

Beginnend wird in diesem Kapitel die Verdrahtung und der Hardwareaufbau der Range Demo beschrieben. Weitergehend wird die Softwareentwicklung der einzelnen Prototypen erläutert. Außerdem wird die horizontale Integration der einzelnen Prototypen ein Themenbereich dieses Kapitels sein.

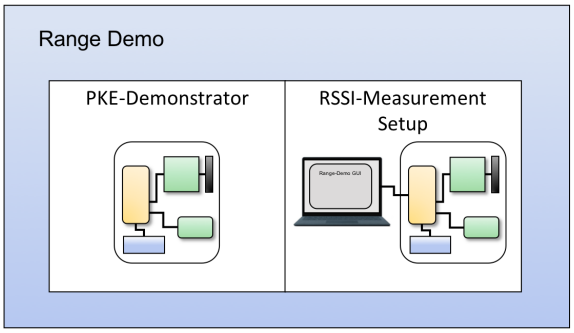


Abbildung 5.1.: Aufteilung der Software, modifiziert nach [2]

In Abbildung 5.1 sind die zuvor, unter Kapitel 3, erläuterten Modi der Range Demo abgebildet. Der Modus *PKE-Demonstrator* kann softwareseitig vollständig auf dem S32K144 realisiert werden. Da auf dem S32K144 nur begrenzt geringe Speichermöglichkeiten vorhanden sind, muss für die Messdatensicherung des RSSI-Measurement Setups eine weitere Softwarekomponente vorgesehen werden. Dieses soll auf dem Rechner arbeiten, auf dem die Messdaten schlussendlich abgespeichert werden. Im weiteren Verlauf dieses Kapitels wird daher eine Unterscheidung zwischen der Entwicklung des PKE-Demonstrators und des RSSI-Measurement Setups vorgenommen.

5.1. Hardware

Mit der iterativen Entwicklung der Prototypen können Überschneidungen der Verdrahtungspläne vorkommen. Damit eine klare Definition der Schnittstellen existiert, sind die Arbeitspakete der Verdrahtung aus der Projektplanung vorgezogen und gesammelt bearbeitet. In der folgenden Abbildung ist der entwickelte Verdrahtungsplan der Range Demo im Gesamten zu sehen. Da Teile der Software für den JOKER und Lizard übernommen werden, sind die Komponenten über getrennte SPI-Kommunikationsbusse angeschlossen. Durch diese Trennung kann jedes Modul getauscht werden, ohne Änderungen an anderen Modulen vorzunehmen.

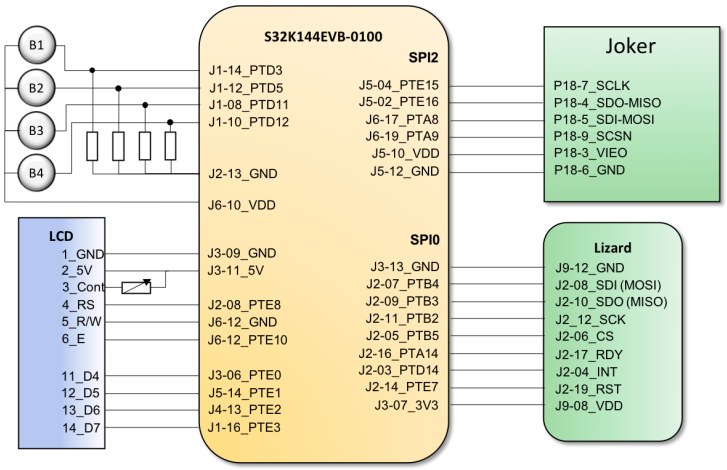


Abbildung 5.2.: Verdrahtungsplan zur Range Demo

Bei der Integration der Softwarekomponenten wurde die Verdrahtung stichprobenartig getsetet und somit die Funktion sichergestellt. Die Informationen zur Port-Belegung des Mikrocontrollers wurden aus der zugehörigen Dokumentation [12, S. 4] sinngemäß entnommen.

LCD

Die Verdrahtung des LCDs lässt sich sinngemäß nach [13] beschreiben.

Das LCD benötigt eine Spannungsversorgung von 5V (Anschlussklemmen 1_GND und 2_5V). Außerdem ist unter dem Kontakt *3_Cont* die Kontrasteinstellung des LCDs zu verstehen. Dieser wird über die Eingangsspannung eingestellt, welche über ein Potentiometer variiert werden kann. Die Anschlüsse *4_RS*, *5_R/W* und *6_E* dienen zur Steuerung der Kommunikation zwischen Host und LCD. Die Kommunikation wird über die vier Datenleitungen *11_D4* bis *14_D7* realisiert. Die Anschlüsse *7_D0* bis *10_D3* wurden aufgrund der 4-Bit-Kommunikation nicht belegt.

Am S32K144 werden die Kontakte *J3-11_5V* für die Spannungsversorgung und *J3-09_GND* als Ground-Anschluss verwendet. Alle weiteren Anschlusstellen sind über

GPIOs[24] realisiert. Für eine einheitliche Kontaktbelegung wurde für das LCD der Port E verwendet.

Lizard

Die Verdrahtungsinformationen des Lizards sind sinngemäß aus der Beschreibung der vorhandenen Bibliothek [14] übernommen.

Für den Anschluss des Lizards an dem S32K144 wurde eine Verdrahtung für den SPI0 vorgesehen. Der Kommunikationsbus SPI benötigt eine MISO [25] und eine MOSI[26] Leitung. Für die Steuerung der bidirektionalen Kommunikation wird die Leitung CS[27] verwendet. Außerdem existiert die Resetleitung *J2-19_RST*, auf der der Lizard einen Resetbefehl für mögliche Kommunikationsprobleme erhalten kann. Mit RDY [28] bietet sich eine Signalleitung über die der Slave (Lizard) dem Master (S32K144) seinen Zustand für die Kommunikation mitteilen kann. Über die Interruptleitung *J2-04_INT* kann der Lizard dem S32K144 seine Slave-Master-Kommunikation ankündigen, wodurch der S32K144 diese priorisiert behandeln kann. Der Lizard benötigt für den Betrieb eine Versorgungsspannung welche über die Anschlüsse *J9-12_GND* und *J9-08_VDD* realisiert ist.

Beim S32K144 können die Kontakte *J2-06_CS*, *J2-17_RDY*, *J2-19_RST* und *J2-04_INT* an jedem beliebigen GPIO angeschlossen werden. Für die Kontakte *J2-08_SDI*, *J2-10_SDO* und *J2-12_SCK* müssen Kontaktstellen für den SPI0 im Pinbelegungs-Plan des S32K144 [15, Registerkarte *Pinout*] ausgewählt werden. Die Kontakte des S32K144 sind in der Abbildung 5.2 zu erkennen.

JOKER

Ähnlich wie beim Lizard ist auch die SPI-Kommunikation des JOKERs mit dem S32K144 realisiert. Die Verdrahtungsinformationen des JOKERs wurden sinngemäß aus den Schematics [16] des Bauteils übernommen.

Für den Betrieb des JOKER wurde auf die optionale Nutzung der Interrupt-, Ready- und Resetleitung verzichtet. Die SPI-Kommunikation wird über das 4-Draht-System realisiert. Hierbei werden MISO, MOSI, SCLK und SCSN verwendet. Neben den Leitungen für die SPI-Kommunikation ist die Versorgungsspannung für das SPI- und I/O-Interface[29] des JOKERs über *P18-3_VIO* realisiert. Am S32K144 wurde der SPI2 für die Kommunikationsschnittstelle mit dem JOKER gewählt.

[24] **G**eneral-**P**urpose **I**nput/**O**utput, engl. für Allzweckeingabe/-ausgabe
[25] **M**aster **I**nput **S**lave **O**utput, engl. für Master Eingang Slave Ausgang
[26] **M**aster**O**utput **S**lave **I**nput, engl. für Master Ausgang Slave Eingang
[27] **C**hip **S**elect, engl. für Chip auswählen
[28] **R**ea**dy**, engl. für bereit
[29] Die Versorgungsspannung des JOKERs wird vom Host gespeist, damit das Spannungslevel für die SPI-Kommunikation gleich ist

5.2. Software - PKE- Demonstrator

Der PKE-Demonstrator wird softwareseitig auf dem S32K144 realisiert. Nach dem Projektplan werden hierfür die vertikalen Prototypen *LCD an S32K144*, *UHF-Empfänger an S32K144*, *LF-Treiber an S32K144*, *Auswertung der Schlüsseldaten auf S32K144* und *Menü auf S32K144* erstellt und zusammengeführt.

5.2.1. Prototyp 1 - LCD an S32K144

Ziel dieses Prototypens ist die Erstellung der visuellen Schnittstelle zum Bediener. Die Fahrzeugschlüsseldaten sollen auf dem LCD ausgegeben werden. Dieser Prototyp integriert das LCD auf dem S32K144.

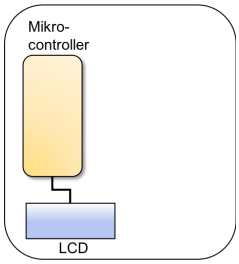


Abbildung 5.3.: Komponentenübersicht zum Prototyp 1, modifiziert nach [2]

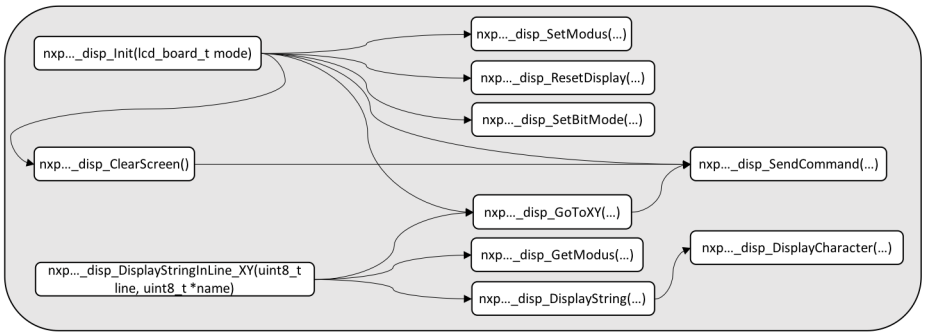


Abbildung 5.4.: Hierachie der Bibliothek *nxpLFCAS_S32K144_RangeDemo_disp*

In Abbildung 5.4 ist die Softwarearchitektur dieses Prototypens zu sehen. Auf der linken Seite befinden sich die Schnittstellenfunktionen. Mit der Funktion *nxp..._disp_Init()* wird das LCD initialisiert. Mit dem Übergabeparameter *mode* erhält diese Funktion Anweisungen für die Initialisierung. Im Fall der Range Demo wird das LCD bei einem Funktionsaufruf für eine 4-Bit-Kommunikation initialisiert. Hierfür müssen mehrere Kommandos über die Funktion *nxp..._disp_SendCommand*

an das Display versendet werden. Diese wiederholten Funktionsaufrufe sind in der Initialisierungsfunktion für das LCD zusammengefasst.

Nach diesem Funktionsaufruf ist das LCD bereit für die Datenausgabe. Durch die erstellte Funktion *nxp..._disp_DisplayStringInLine_XY()* kann ein String mit dem Funktionsaufruf übergeben werden, welcher automatisiert im LCD ausgegeben wird. Zusätzlich muss die Ausgabereihe des Strings im LCD übergeben werden. Für die Darstellung der Fahrzeugschlüsselinformationen wird das LCD an dieser Stelle aufgeteilt. Da dieses insgesamt 80 Zeichen ausgeben kann, muss der übergebene String aus maximal 40 Zeichen bestehen. In Reihe eins werden die Messwerte des Fahrzeugschlüssels eins dargestellt, in Reihe zwei die des Fahrzeugschlüssels zwei. Diese Funktion ist vorallem in Prototyp 4 wichtig, da die Daten dort fertig ausgewertet sind und dargestellt werden sollen.

Neben den eben erwähnten Schnittstellenfunktionen bietet die erstellte Funktion *nxp..._disp_ClearScreen()* die Möglichkeit das LCD auf den Initialisierungsstand zurückzusetzen. Diese Funktion ist für den Prototypen 5 wichtig, damit die Schlüsseldaten, beispielsweise bei dem Wechsel zwischen PKE-Demonstrator und RSSI-Measurement Setup, aus dem LCD entfernt werden können. Die Implementation einer Errorbehandlung konnte aufgrund der einseitigen 4-Bit-Kommunikation nicht implementiert werden.

Zusammenfassend bietet die entwickelte Bibliothek folgende Funktionen:

- *nxp..._disp_Init()*
  -> Automatisierte Initialisierung des LCDs
- *nxp..._disp_ClearScreen()*
  -> LCD Ausgabe leeren
- *nxp..._disp_DisplayStringInLine_XY()*
  -> Getrennte Ausgabe der Strings im LCD für Schlüssel eins und zwei

5.2.2. Prototyp 2 - UHF-Empfänger an S32K144

Ziel dieses Prototypens ist die Integration des UHF-Empfängers Lizard in das Gesamtprojekt. In diesem Teilabschnitt der Softwareentwicklung, wird eine existierende Bibliothek für den S32K144 verwendet, in der der Lizard auf dem S32K144 erfolgreich implementiert ist. Für die Integration in die Range Demo werden Anpassungen im existierenden Quelltext vorgenommen, die in diesem Abschnitt beschreiben werden.

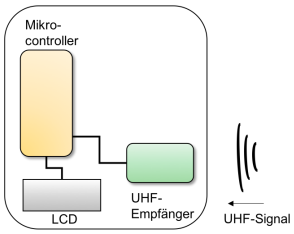


Abbildung 5.5.: Komponentenübersicht zum Prototyp 2, modifiziert nach [2]

Integration der existierenden Lizard-Software in das Range Demo Projekt

Für die Integration der Software in das bestehende S32K144 Projekt, muss die Lizard-Software kopiert und in das Projekt eingefügt werden. Hierfür existiert der Unterordner *nxp..._lz*, in dem alle Quelldateien zur Komponente Lizard eingefügt sind.

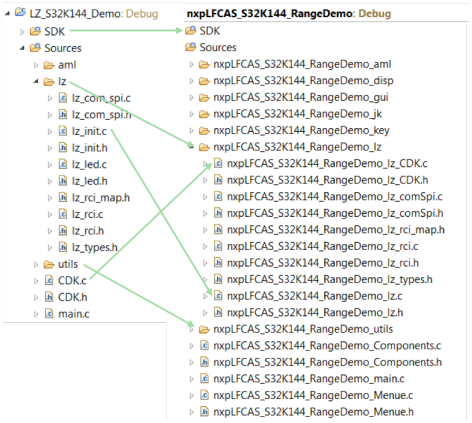


Abbildung 5.6.: Überführung des Lizard-Projektes

In Abbildung 5.6 ist die Überführung der Lizard-Software in das Range Demo Projekt zu sehen. Im linken Bildbereich befindet sich der Projektpfad der vorhandenen Lizard-Software. Dort sind Quelldateien wie *lz_init* kopiert und im Unterordner *nxp..._lz* eingefügt. Außerdem sind die Funktionsnamen und Quelldateinamen bei der Softwareüberführung auf den Projektstil angepasst. Strukturell wurden ebenfalls Änderungen vorgenommen. Als Beispiel hierfür dient *CDK*. Dieses ist im Projekt

LZ_S32K144_Demo auf der Ebene der main-Funktion zu finden. Im Projekt Range Demo ist dieses für eine Modultrennung in den Komponentenordner *nxp..._lz* eingeordnet.

Der Ordner *SDK* liefert Funktionen auf Registerebene, mit S32K144-spezifischen Programmabschnitten. Im diesem sind die Funktionsnamen nicht angepasst, da der Quelltext unverändert zum Originalprojekt ist.

Um ein UHF-Signal über den Lizard auf dem S32K144 zu empfangen, existieren drei Kernaufgaben:

1. Funktionsfähige SPI-Kommunikation zwischen S32K144 und Lizard
2. Parametrierung der Komponente Lizard
3. Parametrierung des Lizards für UHF-Protokoll

Funktionsfähige SPI-Kommunikation zwischen S32K144 und Lizard

Die Initialisierung der SPI-Kommunikation lässt sich über folgende Funktionen durchführen:

- *nxp..._lz_SetupGPIOs()*
- *nxp..._lz_SetupSPI()*

Die Funktion *nxp..._lz_SetupGPIOs()* beinhaltet die Initialisierung der GPIOs, wie beispielsweise *J2-04_INT* oder *J2-19_RST* aus Abbildung 5.2, für die SPI-Kommunikation mit dem Lizard. Über den Funktionsaufruf *nxp..._lz_SetupSPI()* werden Parameter wie beispielsweise die Baudrate des SPIs festgelegt. Darauffolgend wirs der SPI für die Lizard-S32K144-Kommunikation initialisiert.

Der Lizard sendet die empfangenen Daten aus dem UHF-Protokoll des Fahrzeugschlüssels über SPI an den S32K144. Diese Daten müssen gelesen und abgelegt werden. Da in der Range Demo die Daten über SPI jederzeit am S32K144 eintreffen können, müssen diese ohne eine große Verzögerung vom S32K144 empfangen werden. Hierfür wird eine ISR[30] in die Bibliothek *nxp..._lz* integriert.

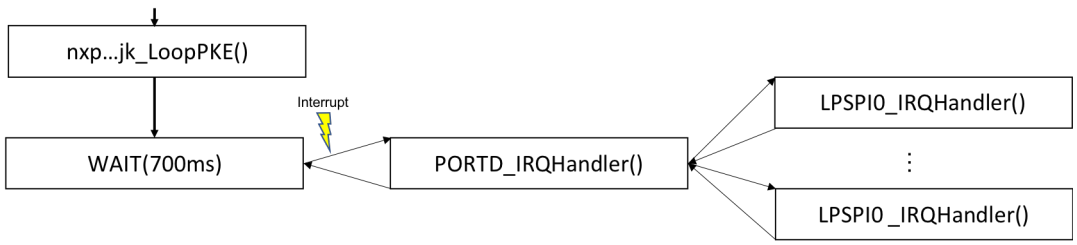


Abbildung 5.7.: Exemplarische Interrupt-Charakteristik

[30] **I**nterrupt **S**ervice **R**outine, engl. für Unterbrechungsroutine

In Abbildung 5.7 ist das Funktionsschema der eingebundenen ISR exemplarisch dargestellt. Während der S32K144 mit dem Abarbeiten der Funktion *WAIT(700ms)* beschäftigt ist, kündigt die Interruptleitung des Lizards einen empfangenen Datensatz eines Fahrzeugschlüssels an. Der S32K144 unterbricht die bearbeitete Funktion und ruft die ISR *PORTD_IRQHandler()* auf. Während dieser Routine werden mehrere Datenpakete über SPI0 am S32K144 eintreffen, welche jeweils mit einem LPSPI0[31]-Interrupt angekündigt und ausgelesen werden. Erst nachdem die Kommunikation über SPI beendet wurde und der letzte SPI-Datensatz ausgelesen ist, kehrt der S32K144 zum Unterbrechungspunkt in *WAIT(700ms)* zurück.

Softwareseitig wird diese Funktionalität in der Quelldatei *nxp..._lz* eingepflegt. *nxp..._lz_SetupGPIOs()* muss den Pin *J2-04_INT* für eine ISR initialisieren. Außerdem muss die Funktion *nxp..._SetupSPI()* den Interrupt des LPSPI0 aktivieren. Neben der Aktivierung ist die Priorität des Interrupts entscheidend. Vorrang hat an dieser Stelle immer der SPI-Interrupt, da dieser jeweils ein Datenwort ankündigt. Mit dem GPIO-Interrupt des Lizards wird ein gesamter Datensatz angekündigt. Über die Funktion *INT_SYS_SetPriority(LPSPI0, 0)* lässt sich die Priorität an die einzelnen ISR vergeben. Der zweite Übergabeparameter beschreibt das Prioritätslevel der einzelnen ISR. Null steht hierbei für die höchste Priorität, die an den SPI-Interrupt vergeben wird. Der GPIO-Interrupt erhält die Priorität eins. Über die Prioritätsvergabe kann die GPIO-ISR immer von der SPI-ISR unterbrochen werden.

Parametrierung der Komponente Lizard

Der letzte Teil zur Initialisierung des Lizards für SPI ist über die Funktion *nxp..._lz_Init()* realisiert. Hierbei wird ein Reboot des Lizards angefordert. Zu diesem Zeitpunkt ist der Lizard bereit für eine SPI-Kommunikation mit der die UHF-Konfigurationen an den Lizard übergeben werden können.

Parametrierung des Lizards für UHF-Protokoll

Die Quelldatei *nxp..._lz_CDK* beinhaltet gesammelte Informationen aus dem SPI-Command-Set des Lizards [17]. Über den Funktionsaufruf *nxp..._lz_CDK_RCILOG Execute()* werden verschiende Kommandos zur UHF-Parametrierung nacheinander über SPI an den Lizard gesendet. In den Kommandos befindet sich beispielsweise die Frequenz mit 433,92MHz, über die der Lizard Daten empfangen soll. Nur bei gleicher UHF-Parametrierung des Lizards und des Fahrzeugschlüssels, können die Daten erfolgreich übermittelt werden. Die Anpassung der UHF-Parametrierung war ein Aufgabenteil dieser Bachelorarbeit. Direkte Informationen zur Parametrierung des Lizards sind aus dem Source-Code der Range Demo und aus dem SPI-Command-Set [17] des Lizards zu entnehmen.

[31] **L**ow **P**ower **S**erial **P**eripheral **I**nterface - effiziente SPI Schnittstelle die auf dem S32K144 verfügbar ist, hier wird weiterhin das bekannte SPI-Protokoll verwendet, siehe [5]

Für die automatisierte Versendung der Kommandos über SPI wird in *nxp..._lz_CDK_RCILOGExecute()* die Funktion *nxp..._lz_comSpi_ReadAfter- WriteFrame()* aufgerufen. Das SPI-Protokoll ist über die Funktionen in der Quelldatei *nxp..._lz_comSpi* realisiert.

Empfangen der Schlüsseldaten über SPI

Listing 5.1: Quelltext zur ISR *PORTD_IRQHandler()*

```
1  void PORTD_IRQHandler(void) {
2    //Prüfe ob die Unterbrechung vom Lizard ausgelöst wurde
3      if ((PORTD->ISFR & (1 << LZ_INT_PIN)) != 0u && irqWithRspOfKey == 1u) {
4          //Schreibe Pointer auf Konfiguration und Struktur vom Lizard in Variablen
5          lz_drv_config_t* tempDrvConfigLizard =
6                  nxpLFCAS_S32K144_RangeDemo_lz_GetDrvConfigLizard();
7          lz_fr_raw_t* tempRecvFrameLizard =
8                  nxpLFCAS_S32K144_RangeDemo_lz_GetRecvFrameLizard();
9          //Lese Daten der aktuellen Unterbrechung über SPI vom Lizard
10         nxpLFCAS_S32K144_RangeDemo_lz_comSpi_ReadFrame(tempDrvConfigLizard,
11                 lzWaitRDY_INT, tempRecvFrameLizard,
12                 MAX_BUFFER_LENGTH);
13   }
14   //Reset des Interruptregisters
15   PORTD->ISFR |= (1 << LZ_INT_PIN);
16 }
```

Über die erstellte ISR *PORTD_IRQHandler()* werden die empfangenen Daten des Lizards, die über SPI am S32K144 eintreffen, angekündigt und verwaltet. Über die Funktion *nxp..._lz_comSPI_ReadFrame()* wird der komplette Datensatz eines Fahrzeugschlüssels eingelesen. Diese Funktion wird vom *LPSPI0_IRQHandler* mehrfach unterbrochen.

Zusammenfassend bietet die überführte Softwarestruktur *nxp..._lz* jetzt folgende Quelldatein:

- *nxp..._lz*

  -> Funktionen zur Initialisierung des SPIs

- *nxp..._lz_comSpi*

  -> Funktionen zur Realisierung des SPI-Protokolls

- *nxp..._lz_CDK*

  -> Funktion zur UHF-Parametrierung des Lizards über SPI

- *nxp..._Components*

  -> ISR zur Sicherung der Messdaten des Fahrzeugschlüssels

5.2.3. Prototyp 3 - LF-Treiber an S32K144

Ziel dieses Prototypens ist die Integration des LF-Treibers JOKER in das Gesamtprojekt. In diesem Teilabschnitt der Softwareentwicklung wird ebenfalls eine existierende Bibliothek eingebunden. Für die Integration in der Range Demo werden Anpassungen im existierenden Quelltext vorgenommen, die in diesem Abschnitt beschrieben werden.

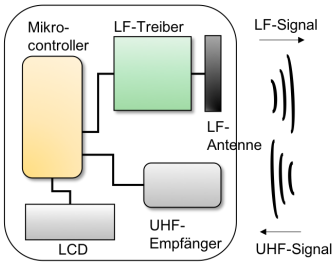


Abbildung 5.8.: Komponentenübersicht zum Prototyp 3, modifiziert nach [2]

Integration der existierenden JOKER-Software in das Range Demo Projekt

Die Bibliothek in den Komponentenordner *nxp..._jk* kopieren, Funktionsnamen an den Programmierstil anpassen und die Anpassung der JOKER- und LF-Signal-Parametrierung waren Aufgaben für die erfolgreiche Integration der vorhandenen Bibliothek in das Range Demo Projekt. Der Unterordner SDK ist ein automatisch generierter Ordner bei der Softwarerealisierung über Processor Expert[32]. Da sowohl die JOKER-Software, als auch die Lizard-Software mit Processor Expert erstellt wurden, kann dieser Ordner einmalig in das Range Demo Projekt überführt werden. In Prototyp 2 wurde letzteres erfolgreich durchgeführt. In Abbildung 5.9 ist exemplarisch die Überführung der einzelnen Quelldateien zu erkennen.

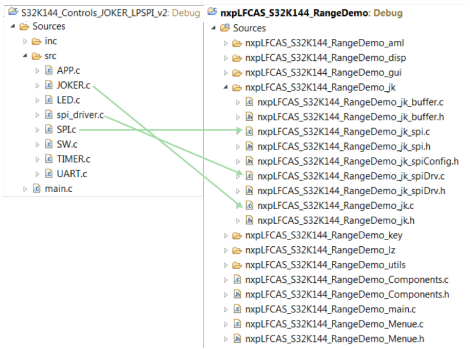


Abbildung 5.9.: Überführung des JOKER-Projektes

[32] Processor Expert ist ein Hilfswerkzeug der S32 Design Studio Entwicklungsumgebung. Mit diesem Werkzeug kann Quelltext für die Initialisierung einzelner Komponenten automatisch generiert werden.

Es gibt drei grundlegende Softwareaufgaben, die vor dem Versenden eines LF-Signals mit dem JOKER abgearbeitet werden müssen. Hierzu zählen:

1. Funktionsfähige SPI-Kommunikation zwischen S32K144 und JOKER
2. Parametrierung der Komponente JOKER
3. Parameterierung des LF-Sendesignals auf dem JOKER

In den folgenden Unterkapiteln werden diese Aufgabenteile erklärt. Außerdem werden die Schnittstellenfunktionen des Softwareprodukts zur Aufgabenlösung systematisch erläutert.

Funktionsfähige SPI-Kommunikation zwischen S32K144 und JOKER

Da in der existierenden JOKER-Bibliothek ebenfalls LPSPI0 verwendet wird, muss diese Schnittstelle in der Software angepasst werden. Die Änderung auf eine SPI-Kommunikation über LPSPI2, musste in den folgenden Quelldateien durchgeführt werden:

- *nxp..._jk_spi*
- *nxp..._jk_spiDrv*
- *nxp..._jk_spiConfig*

Mit der abgeänderten Bibliothek bieten sich für die horizontale Integration des JOKERs die Funktionen *nxp..._jk_spi_Read()* und *nxp..._jk_spi_Write()*. Nach dem Funktionsaufruf *nxp..._jk_spi_Init()* aus einer übergeordneten Funktion, können diese den LPSPI2 bedienen und Kommandos zum JOKER senden beziehungsweise vom JOKER empfangen.

Parametrierung der Komponente JOKER

Für die Parametrierung des JOKERs befinden sich im SPI-Command-Set [18] Informationen über die Protokollführung und Kommandos. Über die Funktion *nxp..._jk_StartUp()* werden dem JOKER Einstellungen und Parameter, wie beispielsweise das Spannungslevel des Pins *P18-3_VIO*, übermittelt. Auch die Versionsnummer des JOKERs wird über diese Funktion angefragt und empfangen. Sobald die Übermittlung dieser Einstellungen beendet wurde, ist die Komponente JOKER initialisiert.

Parameterierung des LF-Signals auf dem JOKER

Auf die Parametrierung der Komponente folgt die Einstellung des LF-Signals (ähnlich wie in Abbildung 2.6). Dieses wird über den Funktionsaufruf *nxp..._jk_Init()* realisiert, wobei das Signal auf verschiedene Formate eingestellt werden kann. Im Fall der Range Demo wurde das LF-Protokoll auf die bereits parametrierten Fahrzeugschlüssel angepasst. Hierfür diente die interne Dokumentation zu den Fahrzeugschlüsseln [4] als Informationsquelle. In der Funktion *nxp..._jk_Init()* befinden sich daher Kommandos zur Länge der Preamble, Form der Synchronisation, Codierung der WUP ID und auch zur Länge der 125kHz Sinuswelle. Außerdem wird dem JOKER die Stromstärke für das LF-Signal übermittelt. Parameter hierzu sind im Testplan unter der Kategorie *Parameter* zu finden.

Versenden des LF-Signals

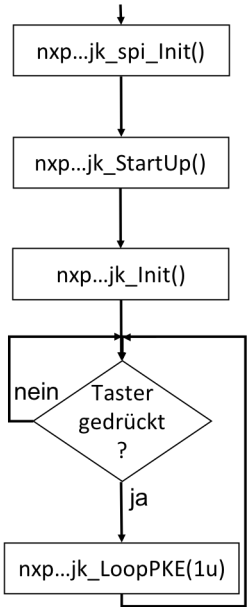


Abbildung 5.10.: Ablaufdiagramm zum Prototyp 3

Für die Integration in das Range Demo Projekt befinden sich in dem Ablaufdiagramm zwei Teilbereiche. Alle Funktionen zur Vorbereitung der Komponente JOKER für ein LF-Signal sind in den ersten drei Initialisierungsschritten zu finden. Mit der Funktion *nxp..._jk_LoopPKE* bietet die Bibliothek des JOKERs eine Schnittstellenfunktion zum Versenden des LF-Signals. In diesem Prototypen ist exemplarisch ein Tastendruck zum Start des LF-Protokolls eingebunden. Unter Prototyp 5 werden andere Schnittstellen zur automatischen Wiederholung der PKE Schleife erläutert. Über den

Funktionsaufruf *nxp..._jk_LoopPKE(1u)* kann das LF-Signal einmalig versendet werden.

Zusammenfassend bietet die Softwarestruktur jetzt folgende Bibliotheken:

- *nxp..._jk_spi* und *nxp..._jk_spiDrv*
  -> Funktionen zur Initialisierung des SPIs
  -> Funktionen zur Realisierung des SPI-Protokolls
- *nxp..._jk*
  -> Funktionen zur Parametrierung des JOKERs über SPI
  -> Funktionen zur Parametrierung des LF-Signals über SPI
  -> Funktion zum Versenden des LF-Signals

Im Prototypen 5 wird weiter auf die horizontale Integration dieses Prototypens eingegangen.

5.2.4. Prototyp 4- Auswertung der Schlüsseldaten auf S32K144

Ziel dieses Prototypens ist die Erstellung einer Quelldatei mit der die Daten des Fahrzeugschlüssels ausgewertet werden können. Außerdem sollen die Daten in einen String für die Ausgabe im LCD angeordnet werden.

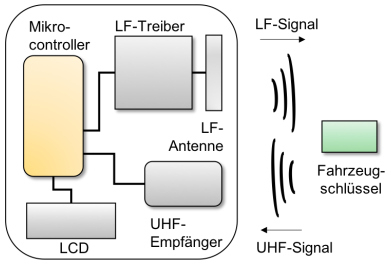


Abbildung 5.11.: Komponentenübersicht zum Prototyp 4, modifiziert nach [2]

Die Quelldatei für diesen Prototypen wurde neu erstellt. Diese verbindet alle Funktionen, die zur Komponente Fahrzeugschlüssel zugeordnet werden können. Die Bibliothek zur Auswertung der Schlüsseldaten wird aus zwei Funktionen aufgerufen. Ein Funktionsaufruf wird aus der ISR *PORTD_IRQHandler()* vorgenommen.

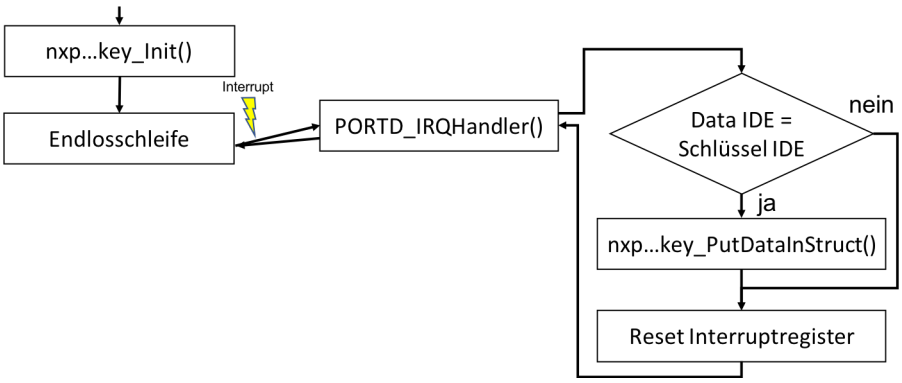


Abbildung 5.12.: Ablaufdiagramm von *nxp..._key*

Mit *nxp..._key_Init()* wird die Struktur für die Messdatensicherung der Fahrzeugschlüsseldaten angelegt. Sobald die ISR *PORTD_IRQHandler()* gestartet wird, wird die Funktion *nxp..._key_PutDataInStruct()* aufgerufen, mit der die neusten Messdaten des Fahrzeugschlüssels abgelegt werden. Über den Funktionsaufruf *nxp..._key_calcData()* in *nxp..._key_PutDataInStruct()* werden die Daten verarbeitet. Ein Teil dieser Funktion bildet beispielsweise die geometrische Summe der Eingangsspannung *V_IN* des Fahrzeugschlüssels. Mit diesem Prototypen wird die ISR erweitert.

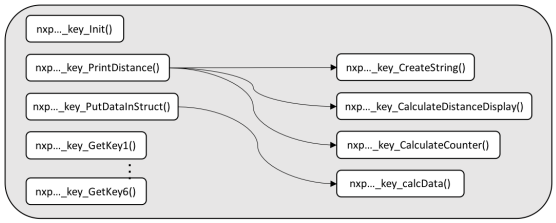


Abbildung 5.13.: Architektur von *nxp..._key*

Strukturell wurde die Quelldatei wie in Abbildung 5.13 aufgebaut. In der linken Hälfte der Abbildung 5.13 befinden sich die Schnittstellenfunktionen zum Prototypen. Damit die Daten des Fahrzeugschlüssels immer in die gleiche Struktur geschrieben werden können, existieren die Schnittstellenfunktionen *nxp..._key_GetKey1()* bis *nxp..._key_GetKey6()*. Über diese wird eine Verknüpfung auf die festgelegte Datenstruktur aus *nxp..._key_Init()* erstellt. Somit kann beispielsweise die ISR *PORTD_IRQHandler()* auf die Datenstruktur zugreifen. Dort befinden sich somit immer die neusten Messwerte des Fahrzeugschlüssels, solange das UHF-Protokoll auf dem Lizard eingetroffen ist[33]. Durch die Verknüpfung kann ebenfalls ein IDE-Abgleich der alten Daten mit den neuen Daten realisiert werden. Sind beispielsweise Daten von der Kennung *0x4C 0x04 0x81 0xE2* am Lizard eingetroffen, können diese mit den vorhandenen Daten im Speicher abgeglichen werden. Stimmen die Fahrzeugschlüssel-IDEs überein, kann mit *nxp..._key_PutDataInStruct()* die Datenstruktur mit den neuen Messwerten überschrieben werden.

Eine weitere Schnittstellenfunktion bietet sich mit *nxp..._key..._PrintDistance()*. Diese wird als Schnittstellenfunktion für den Prototypen 5 und somit für die Vernetzung des Gesamtprojektes verwendet. *nxp..._key..._PrintDistance()* wird aus der Funktion *nxp...Components_Demo()* aufgerufen und startet sowohl Funktionen zur Datenauswertung, als auch die Funktion zur Erstellung des Strings für die Datenausgabe im LCD. Die Rechenwege zur Auswertung der Schlüsseldaten wurden aus der existierenden Software zum RSSI-Measurement Setup übernommen. Eine weitere Funktion, die zu diesem Prototypen entworfen wurde, ist *nxp..._key_CalculateCounter()*. Mit dieser wird die Anzahl der letzten zehn versendeten LF-Signale mit der Anzahl der empfangenen UHF-Signale des Fahrzeugschlüssels verglichen. Antwortet der Fahrzeugschlüssel auf die letzten zehn LF-Signale immer, liefert der Zähler den Wert 10/10. Antwortet der Fahrzeugschlüssel beispielsweise nur jedes zweite Mal auf das LF-Signal, liefert der Zähler den Wert 5/10. Damit die beiden Zahlen verglichen werden können, muss die Anzahl der LF-Signale über den JOKER abgespeichert werden. Über die Funktion *nxp..._jk_GetCounterValue()* konnte die Schnittstelle zwischen Fahrzeugschlüsseldaten und der JOKER-Steuerung verknüpft werden.

[33]Durch Äußere Einflüsse kann es vorkommen, dass der Schlüssel eine Messung durchgeführt hat, die Daten versendet hat, diese allerdings nicht an der Empfangsantenne eintreffen

Ausgabe der Daten über *nxp..._key_PrintDistance*

Für die Programmierung der Datenausgabe im LCD wurde eine Lösungsskizze erstellt.

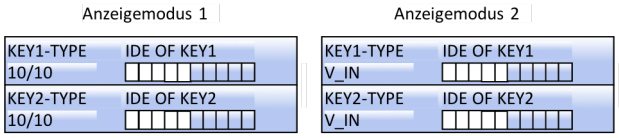


Abbildung 5.14.: Lösungsskizze für die Anordnung der Messwerte der Schlüssel

Das LCD ist in zwei Bereiche aufgeteilt, in denen jeweils die Daten zu einem Fahrzeugschlüssel zu finden sind. In Abbildung 5.14 ist die Trennung der beiden Fahrzeugschlüsseldaten durch die Trennlinie in der Mitte dargestellt. Über die Funktion *nxp..._disp_DisplayStringInLine_XY()* kann die Datenausgabe auf dem LCD realisiert werden. Jeweils ein 40-Zeichen-String wird in die Funktion übergeben, welcher über *nxp..._key_CreateString()* geformt wird. Hierbei greift *nxp..._key_CreateString()* immer auf die aktuellsten Messdaten des Fahrzeugschlüssels zu. Außerdem sind in Abbildung 5.14 unterschiedliche Anzeigemodi zu erkennen. Diese sollen über einen Taster umgeschaltet werden. In *nxp..._key_CreateString()* werden zwei unterschiedliche Strings erstellt, damit im Prototyp 5 der Anzeigewechsel über den Wechsel des Strings implementiert werden kann. Im Anzeigemodus eins befinden sich Informationen zur Aufwachrate des Schlüssels. Anzeigemodus zwei beinhaltet den aktuellen V_IN-Wert. In beiden Anzeigen ist der Schlüsseltyp und die IDE zu finden. Außerdem ist eine Balkenanzeige für die Darstellung der Entfernung eingebunden. Die Balkenanzeige ist ähnlich wie in Abbildung 5.14 jeweils rechts unten im Teilbereich des LCDs zu finden und wird ebenfalls in beiden Anzeigemodi dargestellt.

Zusammenfassend bietet die Quelldatei *nxp..._key* folgende Funktionen:

- Funktion zur Initialisierung der Datenstruktur
- Funktion zur Ablage der Daten des Fahrzeugschlüssels
- Funktion zur Berechnung einiger Kenngrößen aus den Messdaten des Fahrzeugschlüssels
- Berechnung des Zählers 10/10
- Erstellung des Strings für die Datenausgabe

5.2.5. Prototyp 5- Menü und Nutzerführung auf S32K144

Die Trennung der beiden Modi der Range Demo ist über das Zustandsdiagramm realisiert. In der folgenden Abbildung sind die zugehörigen Zustände zu den Modi markiert. Für eine detalliertere Ansicht des Zustandsdiagramms sei auf den Anhang Abschnitt E.2 verwiesen.

Abbildung 5.15.: Menü - Zustandsmaschine

Nach dem Hochfahren oder Reset des S32K144 wird die Range Demo in den Stop-Zustand überführt. Das folgende Ablaufdiagramm beschreibt diesen Vorgang.

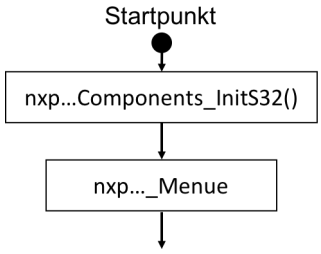


Abbildung 5.16.: Ablaufdiagramm der Startsequenz der S32K144 Software

Mit dem Funktionsaufruf *nxp..._InitS32()* werden die Komponenten Clock,

EEPROM[34], GPIO für die LED und Taster des S32K144EVB-0100 initialisiert. Im Zustand *Stop* wird die Funktion *nxp..._Menue* aufgerufen, mit der eine Menüausgabe in UART realisiert ist (siehe Abbildung E.4). Der Bediener hat in diesem Zustand die Möglichkeit in die Zustände aus Abschnitt E.2 der Range Demo zu wechseln. Das Menü wird über UART ausgegeben und bietet dem Bediener folgende Zustände zur Auswahl:

- PKE-Demonstrator
  - Endlos PKE Loop (Starte über UART)
  - PKE Loop für X Wiederholungen (Starte über UART)
  - PKE Loop über Tasterdruck (Starte über Button oder UART)
- RSSI-Measurement Setup
  - RSSI-Measurement (Starte über UART)

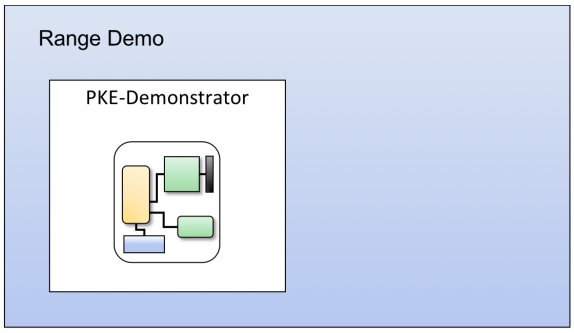


Abbildung 5.17.: Erstellung des PKE-Demonstrators im Gesamtprojekt, modifiziert nach [2]

Der Bediener kann somit den PKE-Demonstrator über einen Tasterdruck oder über die ASCII[35] Texteingabe mittels UART bedienen. Für das RSSI-Measurement Setup ist die Schnittstelle UART eingerichtet, allerdings wird die weitere Bedienung über ein erstelltes UART-Protokoll durchgeführt. Für eine einfachere Schnittstelle zwischen RSSI-Measurement Setup und Nutzer, als die Eingabe über UART, wurde eine GUI erstellt, die im Abschnitt 5.3 detallierter vorgestellt wird. Sobald ein Zustand des PKE-Demonstrators über UART oder Tasterdruck gewählt wurde, wird die Zustandsmaschine in den jeweiligen Zustand überführt.

Alle Fehleingaben werden im Menü abgefangen. Sobald die Fehleingabe über *Enter* bestätigt ist, wird das Menue erneut ausgegeben und bietet die Möglichkeit eine neue

[34] **E**lectrically **E**rasable **P**rogrammable **R**ead **O**nly **M**emory, engl. für elektrisch löschbarer programmierbarer Nur-Lese-Speicher

[35] **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange, beschreibt 7-Bit-Zeichenkodierung

Eingabe durchzuführen. Ebenfalls kann der Bediener über die Taste *Return* die zuletzt eingegebenen Zeichen löschen.

PKE Loop über Tasterdruck

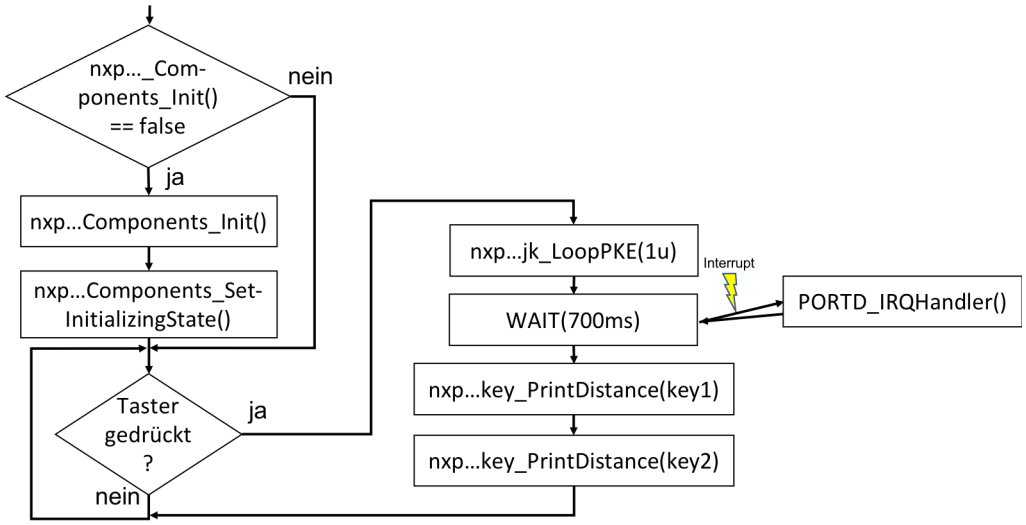


Abbildung 5.18.: Ablaufdiagramm des Zustandes PKE Loop auf Tasterdruck

Wird der Zustand PKE Loop über Tasterdruck gewählt, wird ein LF-Signal über den Tastendruck ausgesendet. Die empfangenen Messdaten über UHF sollen im LCD ausgegeben werden. Zunächst werden die Komponenten für diesen Zustand initialisiert. Über den Funktionsaufruf *nxp...Components_Init()* sind die bekannten Initialisierungsfunktionen aus Unterabschnitt 5.2.1 bis 5.2.4 zusammengefasst. Nach der Initialisierung der Kommunikationsstränge und Komponenten, kann der Taster betätigt werden. Ein einmaliges LF-Signal wird über den JOKER versendet. Während der Wartezeit von 700ms treffen die Messdaten des Schlüssels am S32K144 ein und werden über die ISR in der passenden Struktur abgespeichert und ausgewertet. Mit den Funktionsaufrufen *nxp...key_PrintDistance(key1)* und *nxp...key_PrintDistance(key2)*, werden die aktuellsten Daten des Fahrzeugschlüssels im LCD ausgegeben. Dieser Ablauf kann durch ein erneutes Betätigen des Tasters wiederholt werden.

Endlos PKE Loop

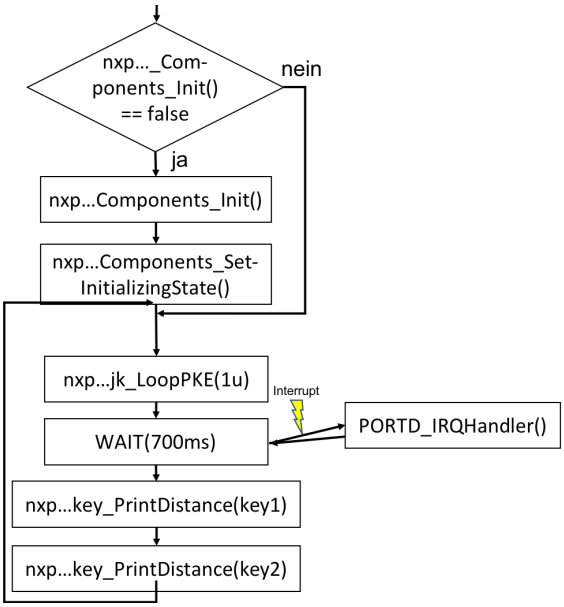


Abbildung 5.19.: Ablaufdiagramm des Zustandes Endlos PKE Loop

Eine ähnliche Funktionsweise bietet auch der Zustand *Endlos PKE Loop*. Hierbei wird die Wiederholung des LF-Signals über eine Endlos-Schleife gesteuert. Somit bietet sich ein System in dem das LF-Signal in einem Zeitintervall wiederholt wird.

PKE Loop für X Wiederholungen

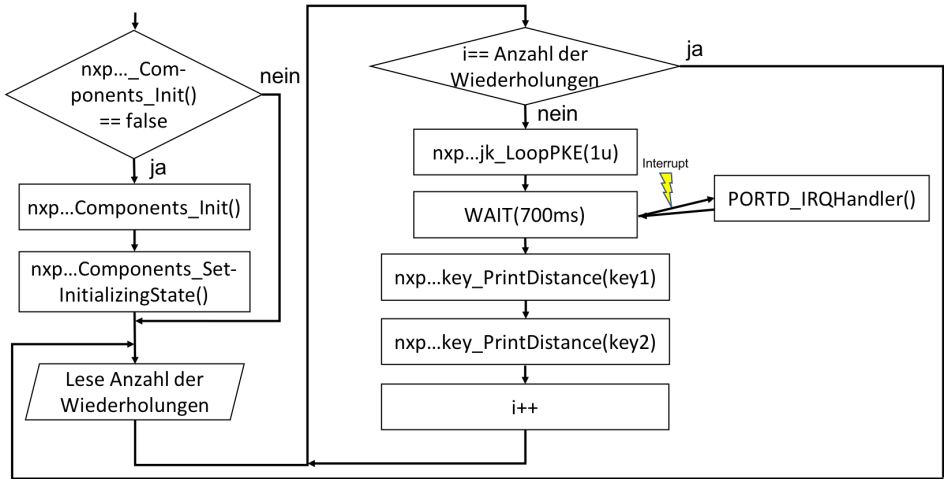


Abbildung 5.20.: Ablaufdiagramm des Zustandes PKE Loop für X Wiederholungen

Auch das Ablaufdiagramm des dritten Zustandes des PKE-Demonstrators ist ähnlich zu den Vorgängern. Da eine Anzahl an Wiederholungen vom Bediener vorgegeben werden muss, ist dieser Zustand nur mit einer Texteingabe über UART bedienbar. Nachdem der Nutzer die Anzahl der Wiederholungen eingegeben hat, wurd das LF-Signal genau wie im Zustand Endlos PKE Loop nach einer Zeitverzögerung wiederholt.

Mit dem Prototypen 5 bieten sich folgende Quelldateien:

- *nxp...main*
  - Verknüpfung der Quelldateien *nxp...Components* und *nxp...Menue*
- *nxp...Menue*
  - Ausgabe der Zustände in UART
  - Einlesen des gewählten Zustandes über UART oder Tasterdruck
- *nxp...Components*
  - Sammelfunktion zur Initialisierung des S32K144EVB-0100
  - Sammelfunktion zur Initialisierung der Komponenten LCD, Schlüsselstruktur, Lizard und JOKER
  - Funktionen zur Reaslisierung der Zustände durch Steuerung der Komponenten
  - Beinhaltet *PORTD_IRQHandler()*

5.3. Software - RSSI-Measurement Setup

Für das RSSI-Measurement Setup wird ein zweiter Modus der Range Demo eingeführt. Dieser wird im Prototyp 6 softwareseitig entwickelt und in das Menü eingebunden.

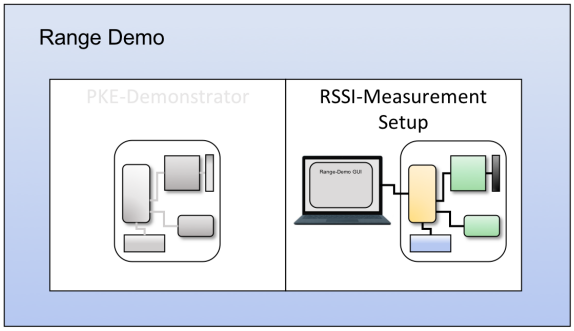


Abbildung 5.21.: Erstellung des RSSI-Measurement Setups im Gesamtprojekt, modifiziert nach [2]

5.3.1. Prototyp 6 - GUI

Das RSSI-Measurement Setup beschreibt den zweiten Modus der Range Demo. Ziel dieses Prototypens ist die Entwicklung einer GUI zur Kommunikation zwischen S32K144 und Laptop. Mit der entwickelten Software sollen die Messdaten des Fahrzeugschlüssels in einem Textdokument abgespeichert werden. Neben der Entwicklung der Software auf dem Laptop, muss die S32K144-Software für die UART-Kommunikation erweitert werden. Außerdem werden mehrere Zustände mit in das bestehende Menü integriert. Das zweite Softwareprodukt in diesem Prototypen wird mit der Sprache C-Sharp in der Entwicklungsumgebung *Visual Studio* entwickelt.

Einführung des UART-Protokolls

Damit die Übertragung der Messdaten des Fahrzeugschlüssels von den Kommandos unterschieden werden kann, wurde ein UART-Protokoll für die Kommunikation zwischen S32K144-Software und GUI eingeführt.

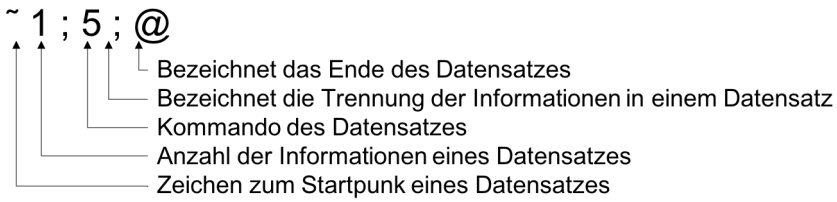


Abbildung 5.22.: Erläuterung des UART-Protokolls

In Abbildung 5.22 wird beispielhaft ein Kommando vom S32K144 zur GUI erläutert. Dieses Kommando wird über UART versendet und gibt der GUI die Information, dass ein Messdurchlauf beendet wurde. Für detalliertere Protokollinformationen sei auf den Abschnitt E.4 hingewiesen, in dem die gesamte GUI-S32K144-Kommunikation mit einer Darstellung beschrieben ist.

Erweiterung der S32K144 Software

Für diesen Prototypen wurde die Range Demo Software um die Quelldatei *nxp..._gui* erweitert. In dieser ist das Kommunikationsprotokoll zwischen S32K144 und GUI sowohl zum Senden, als auch zum Auflösen der empfangenen Daten integriert.

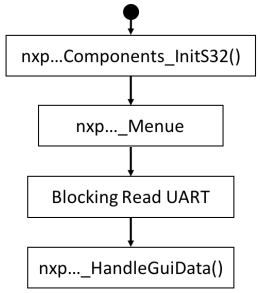


Abbildung 5.23.: Ablaufdiagramm RSSI-Measurement Setup

Ähnlich wie beim PKE-Demonstrator wird der S32K144 nach dem Hochfahren oder auch Reset initialisiert. Hierbei wird der Mikrocontroller auf die UART-Kommunikation mit der GUI vorbereitet. Sobald über UART der Zustand RSSI-Measurement gewählt wird, wartet der S32K144 auf das nächste Kommando. Dieses muss in der Form des erwähnten UART-Protokolls aus Abbildung 5.22 an den S32K144 gesendet werden. Mit der Funktion *nxp...gui_HandleGuiData()* bietet sich eine Funktion zur Verarbeitung des Protokolls. Diese interpretiert das Kommando und liest die Daten aus dem Protokoll. In dem Kommando stehen Anweisungen für die Range Demo Software. Folgende Anweisungen kann der S32K144 verarbeiten:

1. RSSI-Measurement
   - Der S32K144 wird eine definierte Anzahl an Messungen durchführen und die Daten über das UART-Protokoll an die GUI senden
2. RSSI Measurement Set Settings
   - Der S32K144 sendet Einstellungen an die GUI aus dem EEPROM
3. RSSI Measurement Get Settings
   - Der S32K144 bekommt einen Datensatz mit Einstellungen für die GUI gesendet und speichert diesen in seinem EEPROM

Die Zustände Set Settings und Get Settings wurden für die Einstellungssicherung der GUI entwickelt. Mit diesen Modi kann im nichtflüchtigen Speicher des S32K144 die Konfiguration der GUI gesichert werden. Unabhängig von der Software kann der S32K144 die Konfiguration zur GUI senden. Neue Konfigurationen der GUI können im EEPROM des S32K144 gespeichert werden. Mit der Funktion *nxp..._gui_SendMessageToGui()* ist das Protokoll auf der S32K144 Seite realisiert. Mit *nxp..._gui_SendDataToGui()* versendet der S32K144 die Messdaten zur GUI.

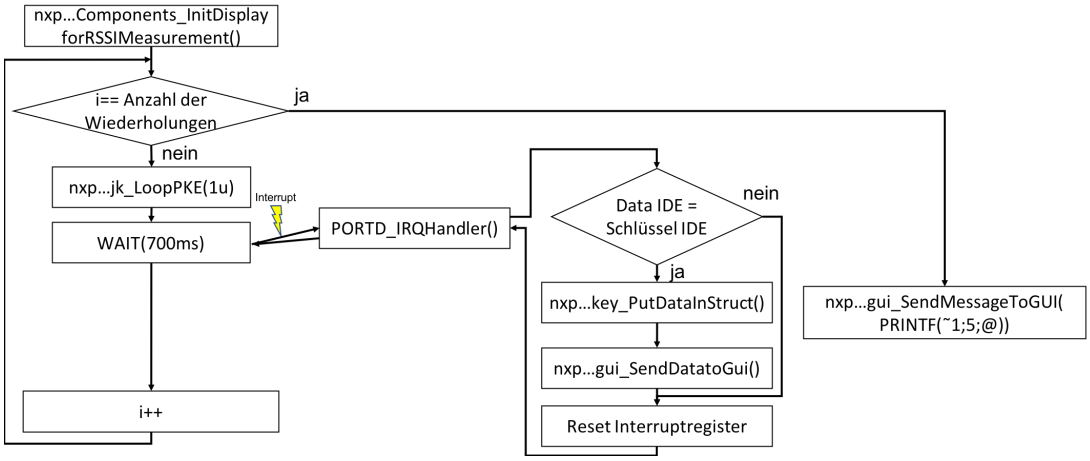


Abbildung 5.24.: Ablaufdiagramm RSSI-Measurement Setup

Ähnlich wie im PKE-Demonstrator ist die Software des RSSI-Measurement Setups aufgebaut. Die ISR wurde um den Aufruf der Funktion *nxp...gui_SendDataToGui()* erweitert. An dieser Stelle werden die Daten dirket aus der Routine an die GUI über das UART-Protokoll weitergeleitet. In diesem Softwareabschnitt wird das LF-Signal mit der in der GUI vorgewählten Anzahl wiederholt. Nach dem letzten Durchlauf der Schleife, sendet der S32K144 über UART das Kommando *~1;5;@*, um der GUI mitzuteilen, dass die Messung beendet wurde.

Erstellung der GUI

Für die Erstellung der GUI-Software wurde ein bestehendes C#-Projekt verwendet. Dieses beinhaltete ein GUI-Fenster ohne Inhalt und die Verbindung der GUI mit der seriellen Schnittstelle. Für den weiteren Verlauf der Erstellung der GUI konnte dieses Softwareprodukt erweitert werden. Für diese Aufgabe wurden vorerst alle Schnittstellen tabellarisch dokumentiert.

Tabelle 5.1.: Schnittstellen der GUI

Nutzer-GUI-Schnittstelle	**Funktion**
Button *Connect*	Auswählen der seriellen Schnittstelle unter Windows
Button *Disconnect*	Freigeben der seriellen Schnittstelle unter Windows
Button *RefreshComPorts*	Aktualisiere angezeigte serielle Schnittstellen
Button *Start RSSI-Measurement*	Starte RSSI-Messung
Button *Save on S32*	Speichere Einstellungen im EEPROM des S32K144
Checkboxen - siehe Abbildung E.8	Auswahlmöglichkeiten zur Messwertdarstellung
Texteingabefeld *Number of measurements*	Eingabe der durchzuführenden RSSI-Messungen
Texteingabefeld *Measurement point of grid*	Eingabe des Messpunktes
Texteingabefeld *Distance of keys [cm]*	Eingabe der gemessenen Distanz zwischen Schlüssel und Sendeantenne
Texteingabefeld *Number of keys*	Eingabe der aktiven Schlüssel im Messaufbau
Texteingabefeld *Current*	Eingabe des JOKER-Stroms für die folgende RSSI-Messung
Texteingabefeld *Path of file*	Eingabe des Pfades für die Erstellung des Textdokumentes
Textdokument (Ausgabe)	Abspeichern der Messdaten der Schlüssel in einem Textdokument
Serial Monitor (Ausgabe)	Ausgabe der Messdaten der Fahrzeugschlüssel in der GUI
GUI-S32K144-Schnittstelle	**Funktion**
MAIN_ReceiveSerialData()	Verarbeiten der empfangenen Kommandos über das UART-Protokoll
MAIN_SendPerProtocol()	Verpacken und Versenden der Kommandos zur GUI über das UART-Protokoll

Auf die dokumentierten Schnittstellen folgte die Bearbeitung der Unteraufgaben zum vertikalen Prototypen. Hierbei wurden die Schnittstellen des Benutzers funktional mit der Schnittstelle zwischen GUI und S32K144 verbunden. Die entwickelte Software ist über die folgende Abbildung dokumentiert.

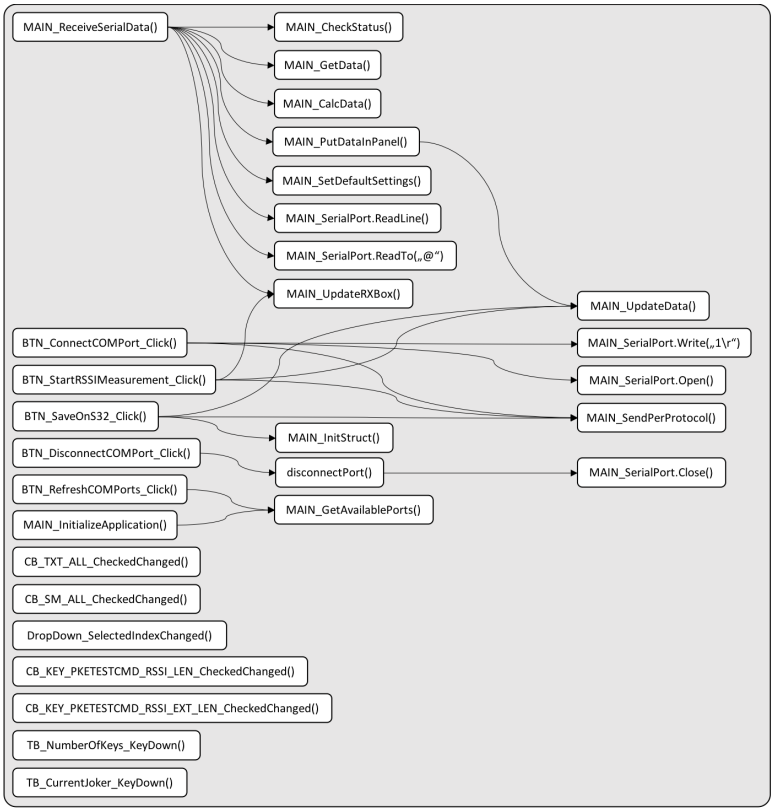


Abbildung 5.25.: Entwickelte Software zur GUI

In Abbildung 5.25 sind auf der linken Seite alle Schnittstellenmethoden der GUI zu sehen. Dort befinden sich beispielsweise Methoden, die durch das Betätigen eines Buttons wie *Start RSSI-Measurement* aufgerufen werden. Diese Aktion beinhaltet unter anderem den Methodenaufruf *MAIN_SendPerProtocol*. Mit diesem wird ein Kommando an die GUI im erwähnten UART-Protokoll verpackt und versendet, wodurch der S32K144 im Modus *RSSI-Measurement Setup* bedient wird. An dieser Stelle ist die Verknüpfung zwischen den Schnittstellen durch die Software zu erkennen. Eine Handlung des Nutzers hat eine Aktion an der GUI-S32K144-Schnittstelle zur Folge.
Neben dem Bedienen des S32K144 über die GUI, werden unter anderem die Messdaten des Fahrzeugschlüssels vom S32K144 zur GUI gesendet. Diese werden sowohl in der GUI dargestellt, als auch in einem Textdokument gesichert. Als Gegenstück zur Funktion *nxp...gui_HandleGuiData()* aus dem S32K144 Projekt, bietet sich somit in der GUI-Software die Methode *MAIN_ReceiveSerialData()*. Mit dieser werden alle Kommandos des S32K144 aus dem UART-Protokoll über den Methodenaufruf, des Objektes *MAIN_SerialPort*, *MAIN_SerialPort.ReadLine()* gelesen. Über *MAIN_CheckStatus()* werden die Kommandos interpretiert, worauf verschiedene Methodenaufrufe zur Initialisierung der GUI, Messdatenausgabe in der GUI und Messdatensicherung im Textdokument folgen.

Zusammenfassend bietet die erstellte GUI-Software folgende Methoden:

- *MAIN_ReceiveSerialData()*
  - Auslesen und Verarbeiten der Kommandos aus dem UART-Protokoll
- *MAIN_SetDefaultSettings()*
  - Einlesen der EEPROM-Daten aus dem S32K144 über das UART-Protokoll
- *BTN_SaveOnS32_Click()*
  - Abspeichern der Plattformeinstellungen in dem S32K144 über das UART-Protokoll
- *BTN_StartRSSIMeasurement_Click()*
  - Plattform zum Starten einer RSSI-Messung mit Vorgabe der Messwertanzahl
- *MAIN_GetData()* und *MAIN_CalcData()*
  - Speichern und Verarbeiten der Messwerte/ Rohmesswerte wie auf dem S32K144
- *MAIN_PutDataInPanel()*
  - Ausgabe der Messdaten in der GUI
  - Abspeichern der Messdaten im Textdokument
- Plattform zur Auswahl der Live-Messdatenanzeige in der GUI
- Plattform in der GUI zur Auswahl der Messwertsicherung in dem Textdokument
- Plattform zur detallierteren Messwertdarstellung der Fahrzeugschlüssel

Für die entwickelte Software sei auf den Quelltext auf der beigelegten CD zur Bachelorarbeit verwiesen. Alle Anforderungen zu diesem Prototypen konnten über die erstellte Software in C# in Kombination mit dem S32K144 Projekt zur Range Demo erfolgreich umgesetzt werden.

6. Testen

In diesem Kapitel wird das Testschema der erstellten Software vorgestellt. Außerdem wird auf den angefertigten Testplan eingegangen.

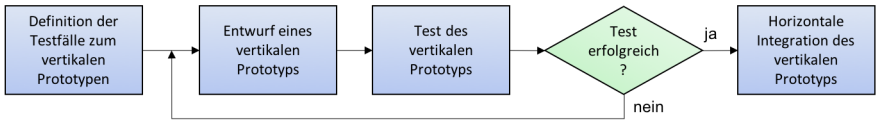


Abbildung 6.1.: Definition der Testfälle, modifiziert nach [6, S. 105]

Aus den technischen Anforderungen (TA-Nummer) des Pflichtenheftes wurden Testfälle (T-Nummer) für die Software definiert. Jeder technischen Anforderung ist mindestens ein Testfall zugeordnet. Diese wurden vor der Entwicklungsphase der einzelnen vertikalen Prototypen, wie in Abbildung 6.1 zu sehen, festgelegt und in einer Excelliste tabellarisch niedergeschrieben. Mit dem Vorgehensmodell *Rapid Prototyping* ist das Testdokument iterativ gewachsen.

In dieser Bachelorarbeit wurden bereits bestehende Bibliotheken zu den Komponenten JOKER und Lizard verwendet. Die übernommenen Quelldatein wurden dabei als vollständig getestet vorausgesetzt. Bei der Abänderung einer Quelldatei, wurden Stichprobentests für die Verifikation der Funktionalität vorgenommen. Als Beispiel gilt die integrierte Software des UHF-Empfängers. Der Unterordner *SDK* wurde hierbei ohne Systemstests übernommen. Die Veränderungen durch das Einbinden zweier Unterbrechungsroutinen wurden durch Debugtests mit dem S32K144 auf eine vollständige Funktionalität verifiziert (siehe Testplan). Für die Kontrolle der neuentwickelten Software wurden ebenfalls Stichprobentests vorgenommen. Hierbei diente vorallem das Oszilloskop und die S32K144-Debug-Umgebung als Hilfsmittel.

Der Testplan wurde zu größten Teilen auf dem Toplevel erstellt. Dabei beziehen sich die Tests häufig auf die Funktionalität einer gesamten Komponente, beispielsweise der des JOKERs. Mit dem Oszilloskop wurde der JOKER-Strom als Referenz für das LF-Signal verwendet. Die modellierten Daten im LF-Signal haben bei vielen Tests zum Prototypen 3 Aufschluss über die Konfiguration des JOKERs gegeben.

Die wichtigsten Inhalte des Testdokumentes werden an dieser Stelle kurz vorgestellt:

- Eindeutige Testnummer
- Nummer der technischen Anforderung
  -> Verknüpfung zur technischen Anforderung aus dem Pflichtenheft
- Funktionsbeschreibung
  -> Vorgabe zum Test
- Parameter
  -> Einzustellende Parameter bei einer Komponente
- Plattform
- Funktion zum Durchführen des Tests
  -> Falls der Test nicht erfolgreich war, werden Änderungen in dieser Funktion/ diesen Funktionen vorgenommen
- Wo ist das Ergebnis des Tests sichtbar?
  -> Messungen mit dem Oszilloskop, Logic-Analyser, Testfunktionen oder auch über den Debug-Modus des S32K144
- Wie wird der Test durchgeführt?
  -> Textuelle Beschreibung der Testdurchführung
- Test erfolgreich
  -> Ablesen des Projektfortschritts

Die technischen Muss-Anforderungen des Pflichtenheftes wurden erfolgreich umgesetzt und im Gesamtsystem implementiert. Durch das Testdokument konnte der Projektfortschritt während der Bearbeitungsphase bemessen werden. Die technische Kann-Anforderung TA-17 konnte nicht vollständig funktionsfähig umgesetzt werden. Diese Information findet sich im Testplan unter der Kategorie *Test erfolgreich?* und dem Testpunkt T-14 mit der Einstufung *Nein* wieder. Die technischen Anforderungen TA-1, TA-4, TA-8, TA-9 und TA-21 wurden erfüllt, aufgrund der hardwareseitigen Anforderung allerdings nicht im Testplan zur Software erwähnt.
Aus dem Testdokument können Informationen zur Verifikation der Funktionalität der Range Demo entnommen werden. Außerdem können Systemtest bei einer Erweiterung der Software wiederholt werden.

Für ausführlichere Informationen über die Testszenarien dieses Softwareproduktes sei auf die Excelliste auf der beigefügten CD zur Bachelorarbeit verwiesen. Informationen zur Strukturierung dieser CD befinden sich im Anhang F.

7. Abschlussfazit

7.1. Bewertung der Range Demo

7.1.1. Pro

Das Vorgehensmodell Rapid Prototyping wurde passend zu dieser Bachelorarbeit gewählt. Durch die verschiedenen Komponenten, die existierenden Softwareprodukte und ein hohes Maß an geforderter Flexibilität bei der Entwicklung der Demonstration, passte die iterative Softwareentwicklung zur Aufgabenstellung. Unter Anwendung des Vorgehensmodell und einer Strukturierung in Ordnern, konnten die Komponenten softwareseitig getrennt werden. Durch diese Struktur können jetzt einzelne Komponenten leicht aus dem Gesamtprojekt entfernt werden.

Wie schon unter Kapitel 6 erwähnt, konnten alle technischen Muss-Anforderungen aus dem Pflichtenheft erfüllt werden. Über die technischen Anforderungen zur Bedienung konnte ein einfach bedienbares Produkt entwickelt werden. Nachdem die Stromversorgung der Range Demo angeschlossen und der Taster *Endless Loop* betätigt wurde, beginnt die Range Demo im Modus *PKE-Demonstrator* ein wiederholendes LF-Signal zu senden. Voll automatisiert werden mit dem S32K144 der Range Demo die Komponenten JOKER und Lizard eingestellt. Außerdem wird das UHF- und LF-Signal parametriert. Der Fokus des Nutzers liegt daher auf den Messwerten der Fahrzeugschlüssel, die im LCD dargestellt werden. Es bietet sich eine Plattform zum Vergleich zweier Fahrzeugschlüssel-ICs, bei dem die Performance der Fahrzeugschlüssel im Mittelpunkt steht. Mit der erstellten GUI bietet sich ein Werkzeug zur Detailpräsentation der Messwerte des Fahrzeugschlüssels. Durch die Funktion des Ausblendens, können wichtige Messwerte hervorgehoben werden, wodurch diese übersichtlich präsentiert werden können. Neben der Präsentation vor dem Kunden bietet die Range Demo eine Zusammenfassung der Kundendemonstrationen und dem RSSI-Measurement Setup.

Durch die Darstellung der Messwerte des Fahrzeugschlüssels im LCD konnte während einer internen Präsentation eine Messungenauigkeit eines Schlüssel-ICs festgestellt werden, welche vor der Verwendung der Demonstration nicht sichtbar war. Unter Verwendung der GUI konnte die Ursache der Messungenauigkeit auf einen Fehler im Fahrzeugschlüssel-IC zurückgeführt werden. Es stellt sich heraus, dass sich die Range Demo auch zur System-Validierung nutzen lässt.

7.1.2. Contra

Die Modularität der erstellten Software ist schon im Projektpfad zu erkennen. Durch die Wiederverwendung existierender Projekte, wurden allerdings unterschiedliche Softwarelösungen in ein Gesamtprojekt portiert. Ein Vergleich der Implementierung der SPI-Steuerung des Lizards und der, des JOKERs zeigt einen großen Unterschied. Funktional ist ein alternativer Aufbau von Funktionsstrukturen kein Problem. Für die Nachvollziehbarkeit des Quelltextes bedeutet dieser Punkt allerdings, dass die Software unübersichtlich wird sobald man die Komponentenebene verlässt und detalliertere Informationen zur Komponentenansteuerung benötigt.

7.2. Ausblick

Für den zukünftigen Verlauf dieses Projektes wurden mehrere Zusatzaufgaben/ Erweiterungen festgehalten. Außerdem wurde eine Messungenauigkeit eines Fahrzeugschlüssel-ICs festgestellt, welche vor der Demonstration mit der Range Demo nicht sichtbar war.

- Erweiterung der Range Demo mit einer 85kHz-Störbox
  - Präsentation der Störfestigkeit des neusten Schlüssel-ICs gegenüber Ladesystemen von Elektrofahrzeugen
- Austauschen des LF-Treibers JOKER mit dem Cendric-Board
  - durch die Modularität der Range Demo kann das neue LF-Treiber-Board Cendric in die Range Demo integriert werden
  - dem Kunden können somit die neusten Produkte von NXP Semiconductors präsentiert werden
- Verbesserung der Unterbrechungsroutinen
  - Einbinden der *Protected Area* beim LPSPI2 für den JOKER zur Sicherung der Kommunikation auf LPSPI2 ohne Unterbrechung des LPSPI0
- Definition von Schwellwerten, die aufgrund einer Entfernung zwischen Fahrzeugschlüssel und LF-Sendeantenne eine Aktion auslösen können
  - die Range Demo würde eine realitätsnähere Präsentation des Systems bilden
  - beispielsweise kann das *Welcome Light* eingebunden werden

Die Entwicklung eines Prototypen der Störbox zeigt, dass die Präsentation der Störfestigkeit mit der Range Demo funktioniert. Außerdem ist die Range Demo für die kommenden Kundenpräsentationen bei verschiedenen OEMs fest eingeplant. Auch für Ausstellungen ist die Range Demo verfügbar, wodurch NXP Semiconductors diese als Plattform zum Anwerben von Neukunden nutzen kann.

A. Anforderungsdokumentation - Lastenheft

Anforderungs - nummer	Wichtigkeit	Anforderungsbeschreibung
	Ziel	**Range Demo - Allgeime Anforderungen**
A-1	muss	Als Referenzsystem für ein Hostsystem muss der S32K144 Mikrokontroller auf dem S32K144EVB-0100 verwendet werden.
A-2	muss	Als UHF-Empfänger muss der Lizard (LID2276) zur Datenübertragung der Messwerte des Fahrzeugschlüssels verwendet werden.
A-3	muss	Als LF-Treiber muss der Joker (LID2334) zum Aufwecken des Schlüssles und zur Magnetfeldbildung für die RSSI-Messung verwendet werden.
A-4	muss	Die Range Demo muss einen Modus zur einfachen Präsentation des PKE-Systems beinhalten.
A-5	muss	Die Range Demo muss den Modus RSSI-Measurement Setup beinhalten, mit dem die Vermessung mehrerer Messpunkte im Fahrzeugaußenbereich und Innenraum möglich ist.
A-6	muss	Das aufzubauende System muss für Modus 1 und Modus 2 kompatibel zu dem bestehenden RSSI-Measurement Setup auf der Schlüsselseite sein.
A-7	muss	Das System muss "stand alone" ohne Zuhilfename eines PC's betrieben werden können.
A-8	muss	Das Demonstrationsboard muss für den Transport zu Messen oder zum Kunden stabil aufgebaut sein.

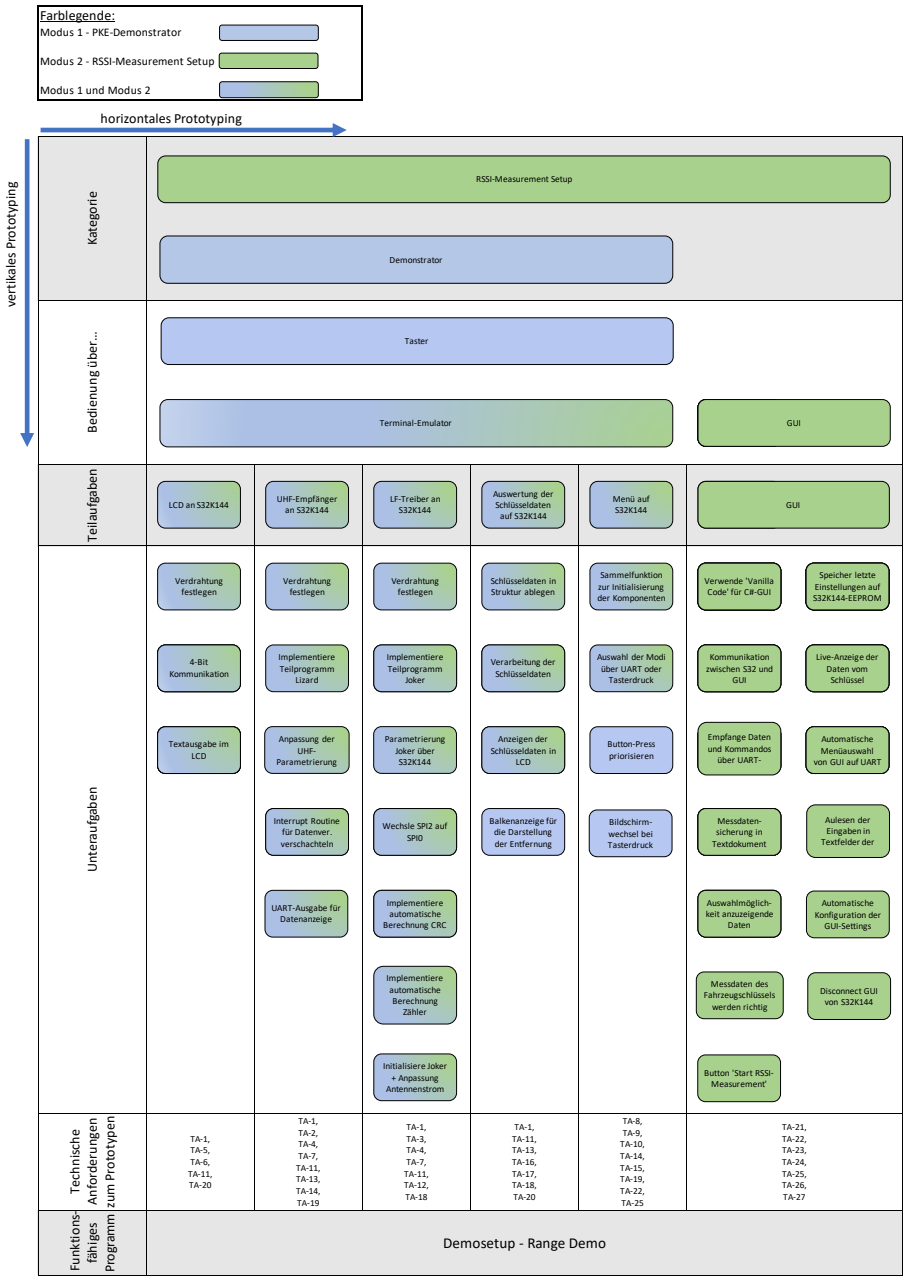
Anforderungs - nummer	Wichtigkeit	Anforderungsbeschreibung
	Ziel	**PKE-Demonstrator - Modus 1**
A-9	muss	Es muss eine einmalige Magnetfeldmessung des Fahrzeugschlüssels gestartet werden können.
A-10	muss	Über eine serielle Schnittstelle (UART) muss der Bediener die Möglichkeit haben eine Entfernungsmessung zu starten.
A-11	muss	Der Bediener muss folgende Eingabemöglichkeit haben: 'Anzahl der durchzuführenden Entfernungsmessungen'.
A-12	muss	Der letzte Messwert der Eingangsspannung des Ics an der LF-Empfängerspule und die IDE des Fahrzeugschlüssels müssen im LCD angezeigt werden.
A-13	kann	Die Magnetfeldstärke des Fahrzeugschlüssels kann im LCD über eine lienare Anzeige dargestellt werden.
A-14	muss	Die Aufwachrate des Fahrzeugschlüssels muss im LCD über einen Zähler dargestellt werden.
A-15	muss	Die Messdaten von mindestens zwei Schlüsseln müssen gleichzeitig visualisiert werden können.
	Ziel	**RSSI-Measurement Setup - Modus 2**
A-16	muss	Der Demoaufbau muss für das RSSI-Measurement Setup mit einem Windowsrechner verbunden werden können.
A-17	muss	Das RSSI-Measurement Setupt muss die Möglichkeit bieten einen oder mehrere Entfernungsmessdurchläufe zu starten.
A-18	muss	Es müssen von mindestens sechs Schlüsseln in einem Messdurchlauf Daten aufgenommen und verarbeitet werden können.
A-19	kann	Die Bedienung des RSSI-Measurement Setups kann über einen Terminal-Emulator möglich sein.
A-20	muss	Die aufgenommenen Messdaten des Fahrzeugschlüssels müssen im Messdurchlauf textuell dargestellt werden.
A-21	muss	Die aufgenommenen Messdaten des Fahrzeugschlüssels müssen separat pro Messpunkt in einem Textdokument gesichert werden.
A-22	muss	Es muss möglich sein, die Werte der textuellen Ausgabe in der GUI und im Textdokument auszuwählen.

B. Anforderungsdokumentation - Pflichtenheft

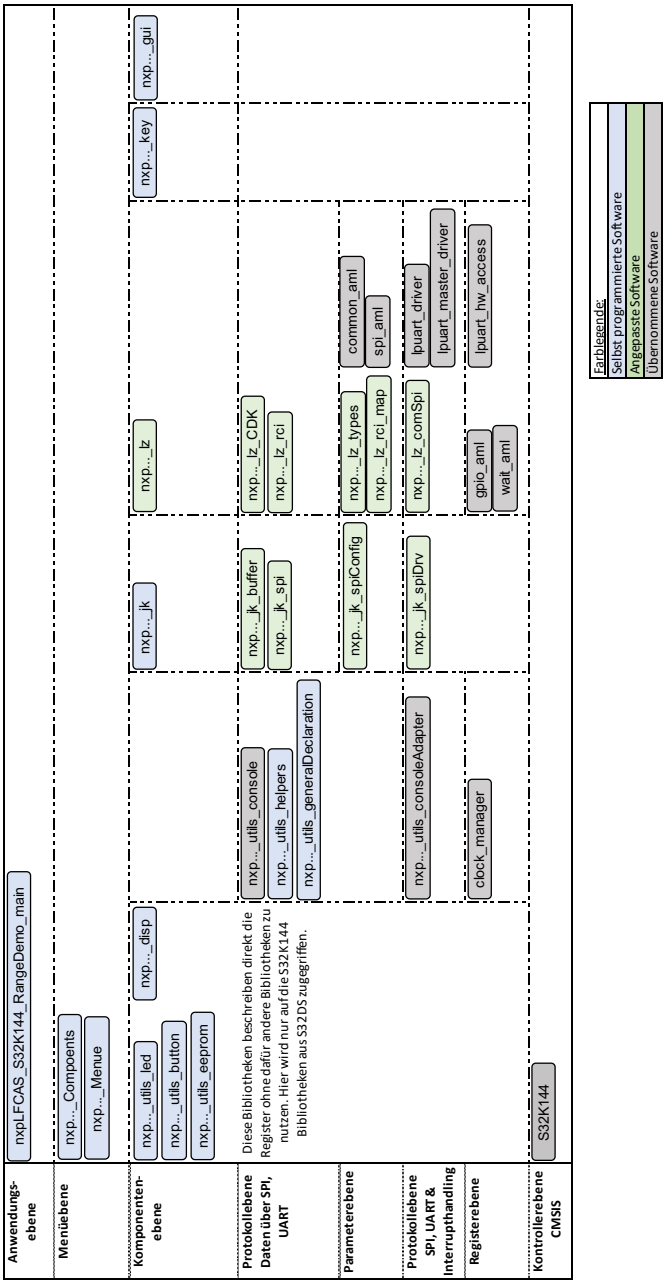
Technische Anforderungs-nummer	Anforderungs-nummer aus Lastenheft	Wichtigkeit	Funktionsspezifische Anforderungsbeschreibung
		Ziel	**Range Demo - Allgemeine Anforderungen**
TA-1	A-1	muss	Als Referenzsystem für ein Hostsystem muss der S32K144 Mikrokontroller auf dem S32K144EVB-0100 verwendet werden.
TA-2	A-2	muss	Als UHF-Empfänger muss der Lizard (LID2276) zur Datenübertragung der Messwerte des Fahrzeugschlüssels verwendet werden.
TA-3	A-3	muss	Als LF-Treiber muss der Joker (LID2334) zum Aufwecken des Schlüssles und zur Magnetfeldbildung für die RSSI-Messung verwendet werden.
TA-4	A-4, A-5, A-6	muss	Das aufzubauende System muss für Modus 1 und Modus 2 kompatibel zu dem bestehenden RSSI-Measurement Setup auf der Schlüsselseite sein.
TA-5	A-7	muss	Das System muss "stand alone" ohne Zuhilfename eines PC's betrieben werden können.
TA-6	A-1	muss	Die Hardware und Software muss für eine 4-Bit Kommunikation zwischen S32K144 und LCD ausgelegt werden.
TA-7	A-1, A-2, A-3	muss	Auf dem S32K144 muss die Kommunikation mit dem UHF-Empfänger und LF-Treiber über SPI realisiert werden.
TA-8	A-8	muss	Das Demonstrationsboard muss auf einer festen Platte installiert sein, welche in einem Koffer für den Transport verstaut werden kann.
TA-9	A-8	muss	Das Demonstrationsboard muss ausschließlich mit Niederspannungen von 3,3V und 12V Gleichspannung betrieben werden.
TA-10	A-4, A-5	muss	Der S32K144 muss softwareseitig ein Menü für die unterschiedlichen Modi für eine klare funktionale Trennung beinhalten

Technische Anforderungs-nummer	Anforderungs-nummer aus Lastenheft	Wichtigkeit	Funktionsspezifische Anforderungsbeschreibung
		Ziel	**PKE-Demonstrator - Modus 1**
TA-11	A-9, A-10	muss	Der S32K144 muss den UHF-Empfänger, den LF-Treiber und das LCD vollständig automatisch mit festen Parameterwerten nach dem System-Reset initialisieren.
TA-12	A-9, A-10	muss	Mit dem S32K144 muss ein LF-Signal über den LF-Treiber zum Aufwecken des Fahrzeugschlüssles versendet werden.
TA-13	A-9, A-10	muss	Mit dem S32K144 muss ein Datensatz aus einem UHF-Signal vom Fahrzeugschlüssel über den UHF-Empfänger empfangen und verarbeitet werden.
TA-14	A-7, A-9	muss	Über den Taster 'Polling' muss automatisch eine einzelne Magnetfeldmessung durchgeführt werden.
TA-15	A-10, A-11, A-16	muss	Der S32K144 muss über UART Kommandos vom Bediener erhalten und über diese Bedienbar sein.
TA-16	A-12, A-13, A-14	muss	Der S32K144 muss die Daten, aus dem zuletzt empfangenen Datensatz, der Fahrzeugschlüssel ausgewertet auf einem LCD anzeigen können.
TA-17	A-13	kann	Der S32K144 kann in einer linearen Anzeige auf dem LCD die Entfernung des Fahrzeugschlüssels zur Sendeeinheit visuell darstellen.
TA-18	A-14	muss	Der S32K144 muss aus den letzten zehn LF-Signalen und den Daten des Fahrzeugschlüssels die Aufwachrate dieses Schlüssels ermitteln.
TA-19	A-15, A-18	muss	Der S32K144 muss die Daten des UHF-Empfängers über eine Interrupt-Routine aufnehmen, damit eine geringe Latenz bei der Datenaufnahme über SPI erreicht wird.
TA-20	A-15	muss	Die Anzeige auf dem LCD muss zweigeteilt werden, damit die Daten von zwei Schlüsseln dargestellt werden können.
		Ziel	**RSSI-Measurement Setup - Modus 2**
TA-21	A-21	muss	Es muss eine GUI für die Textdokumentgenerierung erstell werden, da der S32K144 nur begrenzt kleine Textdokumente erstellen kann.
TA-22	A-16, A-17, A-18, A-21, A-22	muss	Die GUI muss Kommandos über die Kommunikationsschnittstelle UART an den S32K144 senden und Kommandos vom S32K144 empfangen können.
TA-23	A-17	muss	Die GUI muss den Starttaster 'Start RSSI-Measurement' für das implementierte RSSI-Measurement Setup beinhalten.
TA-24	A-18	muss	Die GUI muss die Datensätze vom Fahrzeugschlüssel über die Kommunikationsschnittstelle UART vom S32K144 erhalten und verarbeiten können.
TA-25	A-19	kann	Das RSSI-Measurement Setups kann über einen Terminal-Emulator über UART bedient werden.
TA-26	A-20, A-21	muss	Die verarbeiteten Schlüsseldaten muss die GUI sowohl textuell in ein Ausgabebildschirm, als auch textuell in das generierte Textdokument schreiben.
TA-27	A-20, A-21, A-22	muss	Es muss in der GUI die Möglichkeit geben nur bestimmte Spalten der Datenmatrix (mehrere Datensätze der unterschiedlichen Fahrzeugschlüssel) auszugeben um diese miteinander vergleichen zu können.

C. Projektplan - Rapid Prototyping

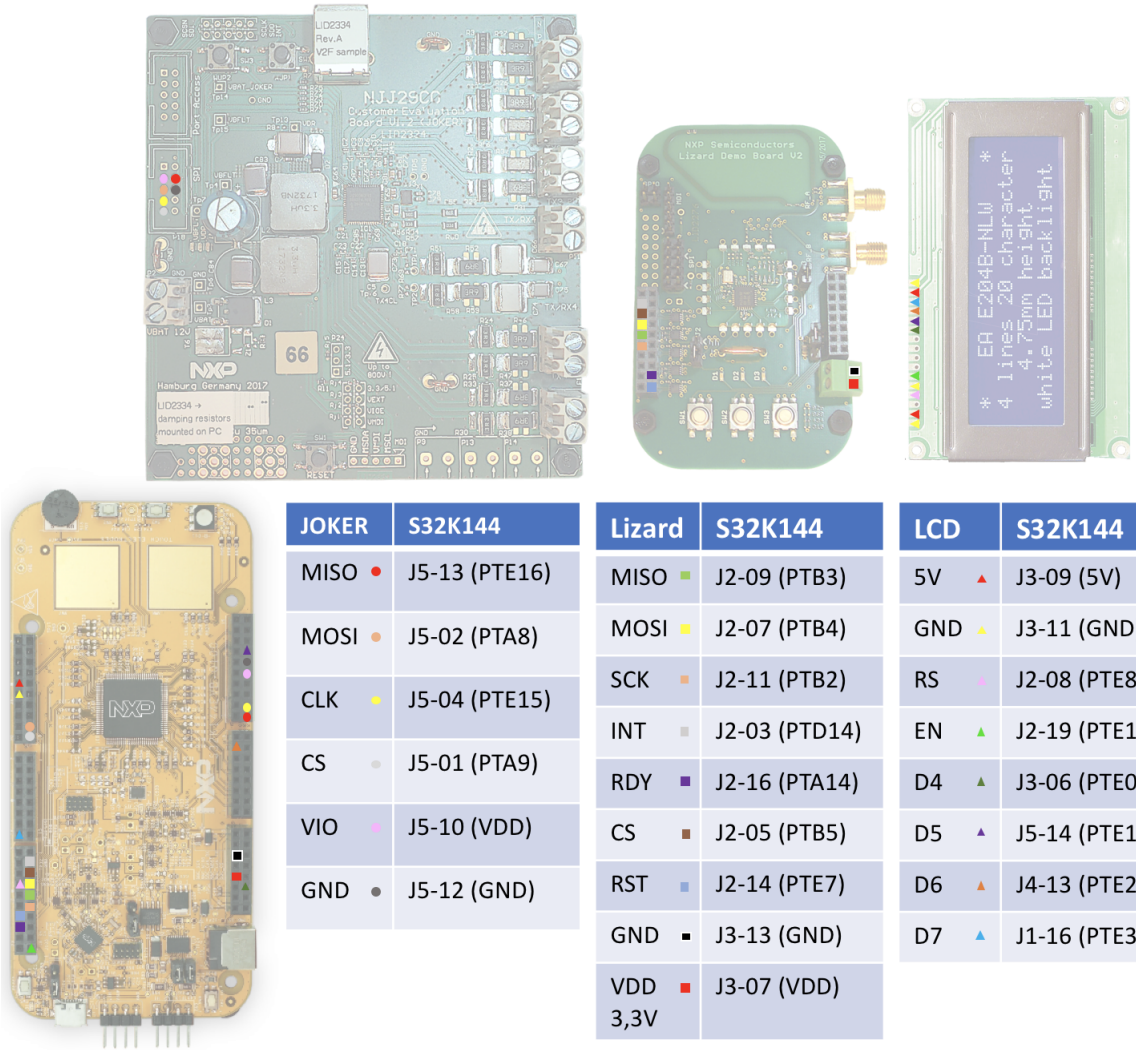


D. Programmhierachieplan



E. Die Range Demo

E.1. Hardware



JOKER	S32K144
MISO	J5-13 (PTE16)
MOSI	J5-02 (PTA8)
CLK	J5-04 (PTE15)
CS	J5-01 (PTA9)
VIO	J5-10 (VDD)
GND	J5-12 (GND)

Lizard	S32K144
MISO	J2-09 (PTB3)
MOSI	J2-07 (PTB4)
SCK	J2-11 (PTB2)
INT	J2-03 (PTD14)
RDY	J2-16 (PTA14)
CS	J2-05 (PTB5)
RST	J2-14 (PTE7)
GND	J3-13 (GND)
VDD 3,3V	J3-07 (VDD)

LCD	S32K144
5V	J3-09 (5V)
GND	J3-11 (GND)
RS	J2-08 (PTE8)
EN	J2-19 (PTE10)
D4	J3-06 (PTE0)
D5	J5-14 (PTE1)
D6	J4-13 (PTE2)
D7	J1-16 (PTE3)

Abbildung E.1.: Übersicht zur Verdrahtung der Range Demo, modifiziert nach [5] und [8]

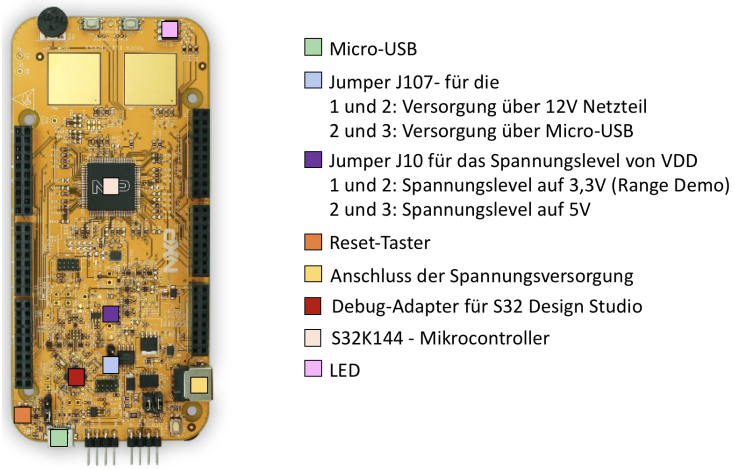


Abbildung E.2.: Konfiguration und Informationen zum Mikrocontroller S32K144, modifiziert nach [5]

Mit dem Jumper J107 kann die Spannungsquelle für das S32K144EVB-0100 ausgewählt werden. Außerdem muss der Jumper 107 auf Steckplatz 2 und 3 positioniert sein, damit die Software auf dem S32K144 debuggt werden kann. Für den Betrieb ohne einen Rechner, muss der Jumper auf Steckplatz 1 und 2 platziert werden. Die Range Demo kann ohne einen Rechner über die unter Abbildung E.3 beschriebenen Taster bedient werden.

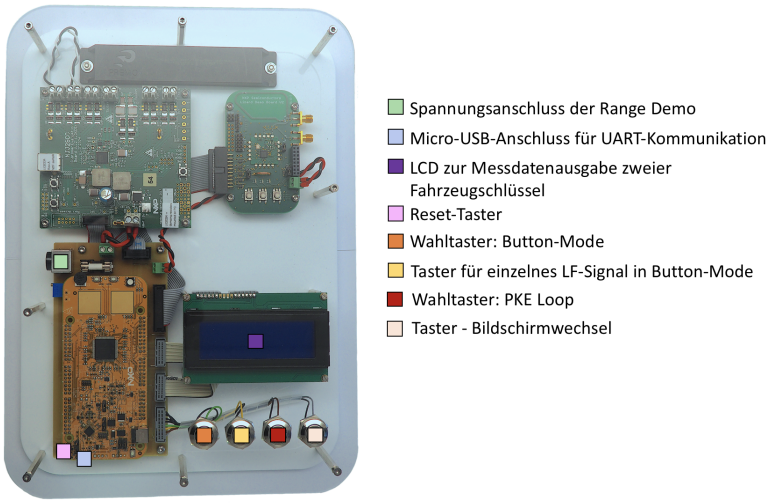
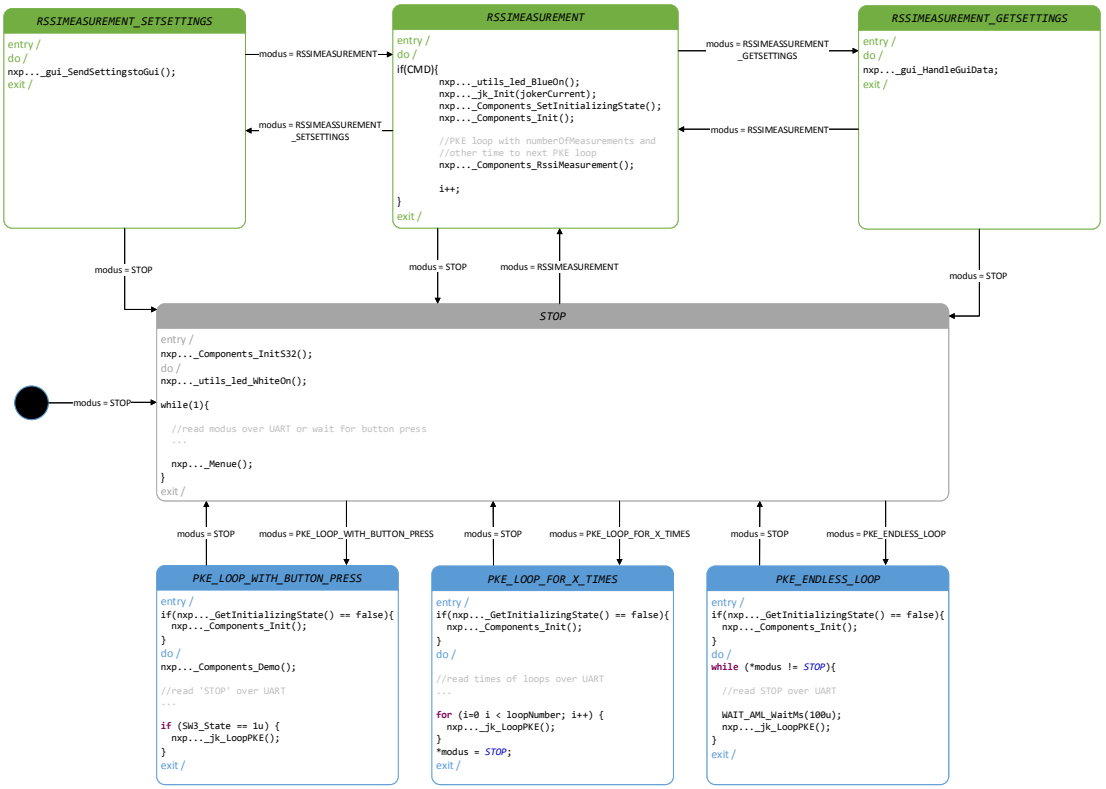


Abbildung E.3.: Schnittstellen zwischen Bediener und Range Demo

In dieser Abbildung ist die Range Demo zu sehen. Ähnlich wie in der Lösungsskizze, sind die Komponenten auf diesem Board angeordnet. Außerdem sind alle Schnittstellen zum Nutzer in dieser Darstellung beschrieben.

In dieser Abbildung ist das fertige Demo Board zu sehen, bei dem eine Messung durchgeführt wurde. Die Messwerte werden auf dem Display im PKE-Demonstrator Modus angezeigt. Die Demo ist auf der linken Seite über einen 12V-Anschluss gespeist. Das Kabel am S32K144 dient zur Kommunikation über UART.

E.2. Zustandsdiagramm der Range Demo



Im unteren Bereich des Zustandsdiagramms (blau markiert) befinden sich die Zustände des PKE-Demonstratos. Diese können über UART und teilweise über Tasterdruck ausgewählt werden. Im oberen Bereich befinden sich die grün markierten Zustände. In diese Zustand kann über das Kommando *1\r* gewechselt werden. Aus dem Zustand *RSSIMEASUREMENT* kann in die Zustände *RSSIMEASUREMENT_GETSETTINGS* oder *RSSIMEASUREMENT_SETSETTINGS* nur über einen Befehl in der Form des entworfenen UART-Protokolls gewechselt werden. Die Implementation des Zustandsdiagramms ist in den Quelldatein *nxp...main* und *nxp...Menue* zu finden.

E.3. Menüausgabe und exemplarische Bedienung in UART

```
*******************************************************************************
APPLICATION NAME:                                   RANGE DEMO

*******************************************************************************
MENU:                                               RANGE DEMO

STOP MODUS:                                         0
RSSI-MEASUREMENT-SETUP MODUS:                       1
PKE ENDLESS LOOP:                                   2
PKE LOOP FOR X TIMES:                               3
PKE LOOP WITH BUTTON PRESS:                         4
SELECT THE MODUS NOW (REQUEST WITH ENTER):
```

Abbildung E.4.: Ausgabe des Menüs der GUI in UART

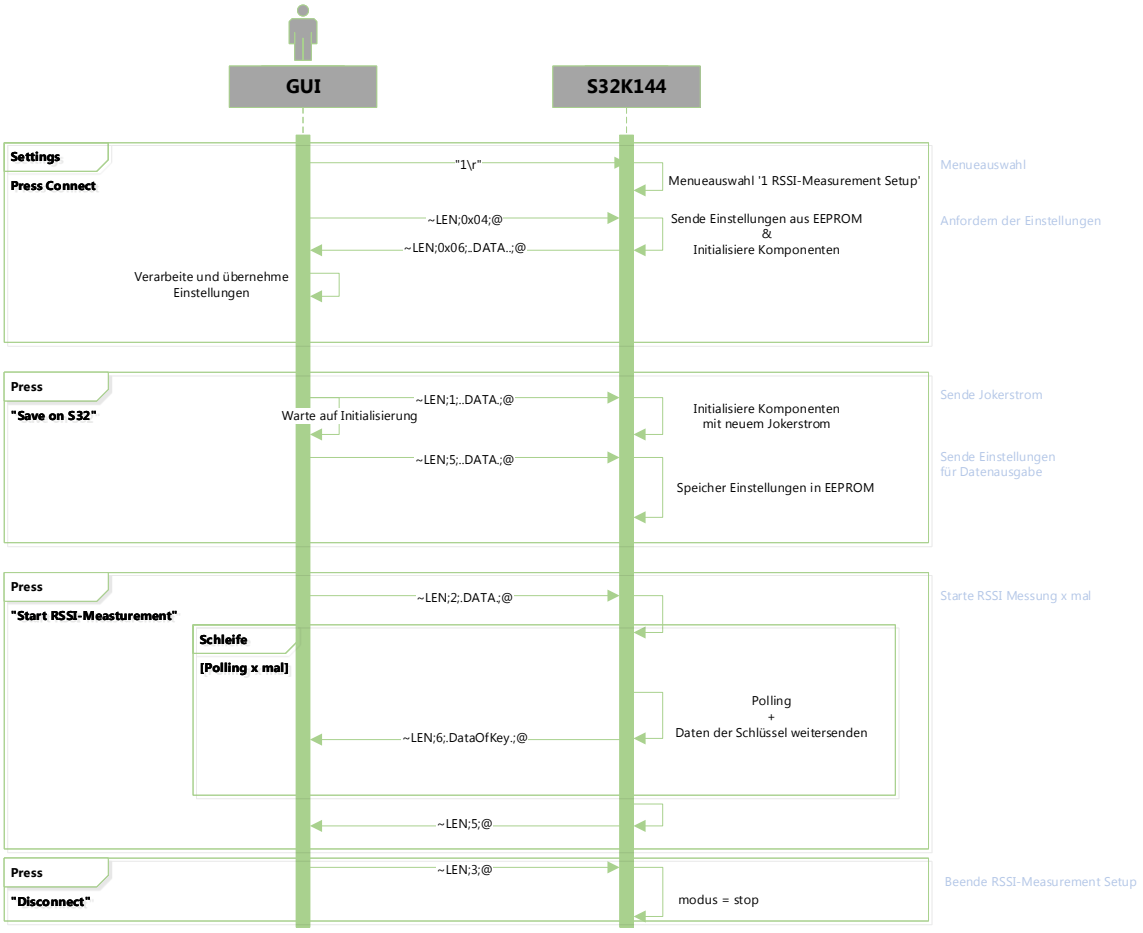
Über die Eingabe des gewählten Menüpunktes kann der Zustand der Range Demo ausgewählt werden. Durch die Bestätigung mit *Enter* wird die Range Demo in den Zustand überführt.

```
*******************************************************************************
APPLICATION NAME:                                   RANGE DEMO

*******************************************************************************
MENU:                                               RANGE DEMO

STOP MODUS:                                         0
RSSI-MEASUREMENT-SETUP MODUS:                       1
PKE ENDLESS LOOP:                                   2
PKE LOOP FOR X TIMES:                               3
PKE LOOP WITH BUTTON PRESS:                         4
SELECT THE MODUS NOW (REQUEST WITH ENTER):          2
*******************************************************************************
INITIALIZING:                                       JOKER HARDWARE SUCCESSFUL
INITIALIZING:                                       JOKER SOFTWARE SUCCESSFUL
INITIALIZING:                                       LIZARD SUCCESSFUL
INITIALIZING:                                       ALL COMPONENTS SUCCESSFUL

*******************************************************************************
MODUS:                                              PKE ENDLESS LOOP
PRESS 0 AND ENTER TO STOP INF-LOOP:                 
```

Abbildung E.5.: UART Ausgabe im Modus *ENDLESS PKE LOOP*

In Abbildung E.5 wurde der Modus 2 *PKE ENDLESS LOOP* gewählt. In diesem werden alle Komponenten initialisiert. Über die Ausgabe kann die erfolgreiche Initialisierung der Komponenten überprüft werden. Darauffolgend wird das LF-Signal zyklisch versendet. Die Messdaten der Fahrzeugschlüssel werden jetzt im LCD angezeigt. Der Zustand kann über die Eingabe *0* und der Bestätigung durch *Enter* in den Stop-Zustand überführt werden.

E.4. Interaktion zwischen GUI und S32K144



In dieser Abbildung ist die Protokolldokumentation zwischen S32K144 und GUI zu erkennen. Das erste Kommando von der GUI zum S32K144 ist die Auswahl des Zustandes RSSI-Measurement Setup mit $1\backslash r$ zu sehen.

E.5. Die GUI

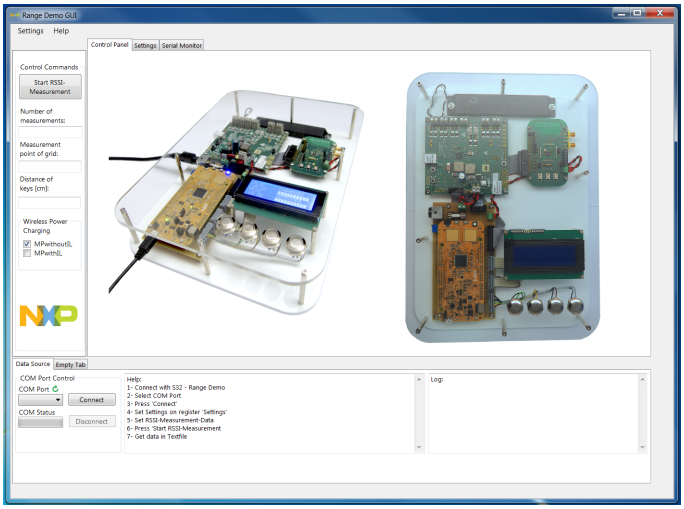


Abbildung E.6.: GUI direkt nach dem Programmstart

In Abbildung E.6 ist die GUI nach dem Start über die ausführbare Datei (.exe-Datei) zu sehen. Das Range Demo S32K144 Board ist noch nicht mit der GUI verbunden.

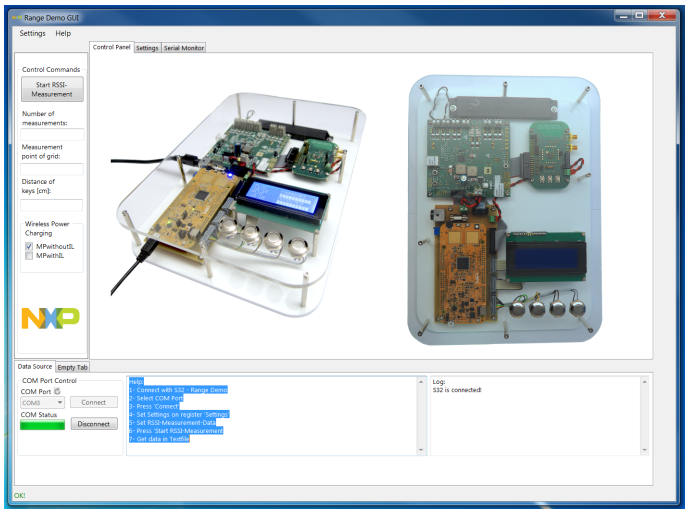


Abbildung E.7.: GUI - Verbunden mit dem Range Demo S32K144 Board

Nach der Auswahl der seriellen Schnittstelle unter Windows kann sich die GUI über das Betätigen des Buttons *Connect* mit dem Range Demo S32K144-Board verbinden und wählt mit dem Kommando $1\backslash r$ automatisch den Modus RSSI-Measurement über UART. Zu erkennen ist die erfolgreiche Verbindung an dem grünen Ladebalken unten links im Bild.

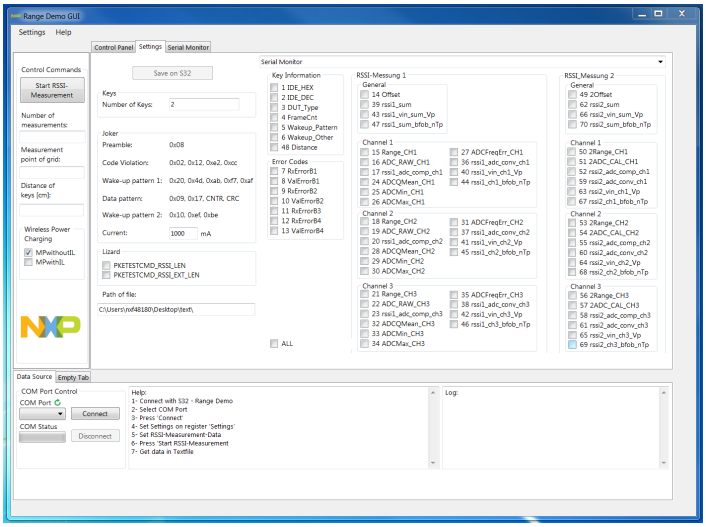


Abbildung E.8.: GUI-Einstellungen vor dem Verbinden mit dem Range Demo S32K144 Board

Wird die GUI gestartet, zeigt sich unter dem Reiter *Settings* die Struktur wie in Abbildung E.8. Zu sehen sind die Wahlkästchen für die Ausgabe der Messwerte der Fahrzeugschlüssel im Reiter *Serial Monitor*, welche nicht angewählt sind.

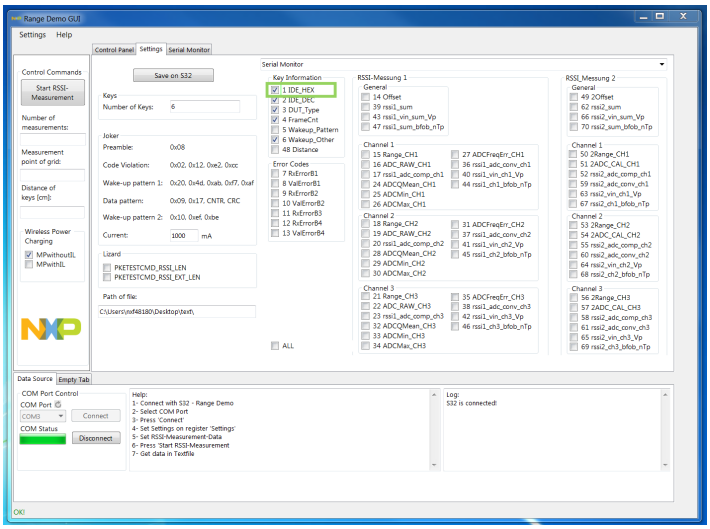


Abbildung E.9.: GUI-Einstellungen für den seriellen Monitor nach dem Verbinden mit dem Range Demo S32K144 Board

Wird die Verbindung zwischen Range Demo S32K144 Board und der GUI hergestellt (siehe grüner Ladebalken unten links in Abbildung E.9), werden die zuletzt gesicherten Einstellungen der GUI automatisch vom S32K144 zur GUI gesendet und eingepflegt. Als Beispiel gilt hierfür das Kästchen *1_IDE_HEX* (grün markiert).

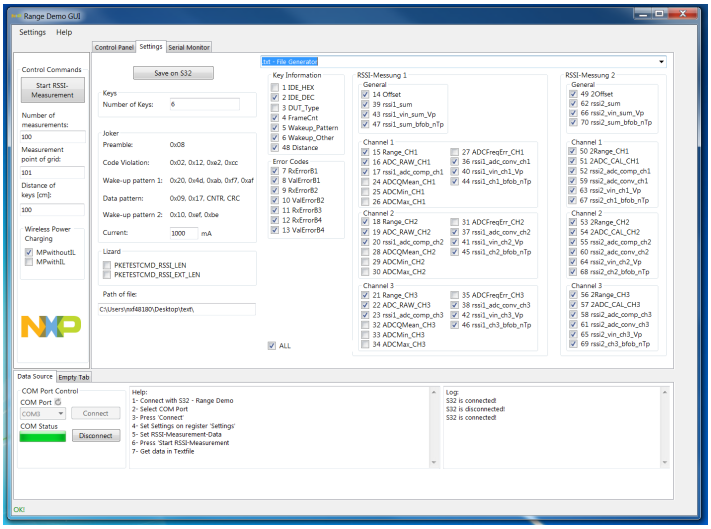


Abbildung E.10.: GUI-Einstellungen für das Textdokument nach dem Verbinden mit dem Range Demo S32K144 Board

In Abbildung E.10 ist in blau markiert das Drop-Down-Menü für die Kästchen zu erkennen. Wechselt man dort auf *.txt-File Generator*, können die Einstellungen für das Textdokument vorgenommen werden. Unter *Serial Monitor* können die angezeigten Live-Messdaten in der GUI zu- beziehungsweise abgeschaltet werden.

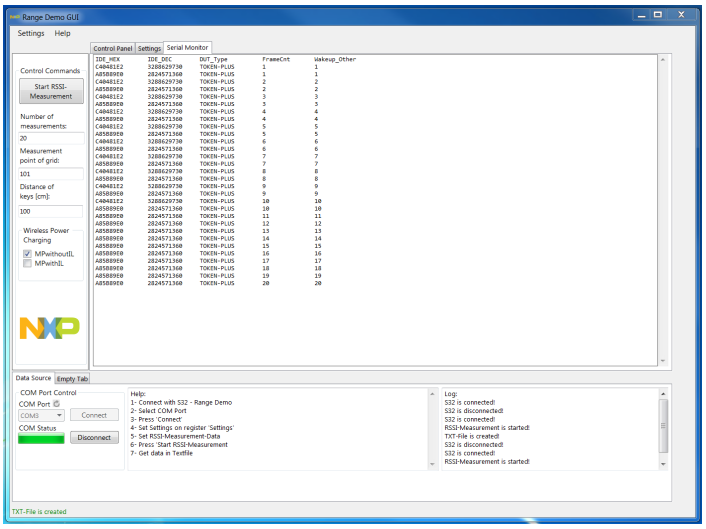


Abbildung E.11.: Starten einer RSSI-Messung

Die Anzahl der Messungen, der Messpunkt, an dem der Schlüssel liegt und die Entfernung zwischen Sendespule und Fahrzeugschlüssel können in den Textfeldern unter den Button *Start RSSI-Measurement* im linken Bereich der GUI eingetragen werden.

Eine RSSI-Messung kann mit dem Button *Start RSSI-Measurement* gestartet werden. Die GUI wechselt nach dem Betätigen automatisch in den Reiter *Serial Monitor*, in dem die empfangenen UHF-Daten der Fahrzeugschlüssel dargestellt werden. Außerdem generiert die GUI ein Textdokument in dem die Messdaten ebenfalls wiederzufinden sind.

	IDE_DEC	FrameCnt	Wakeup_Pattern	Wakeup_Other	RxErrorB1	ValErrorB1	RxErrorB2	ValErrorB2	RxErrorB3	RxErrorB4	ValErrorB4	Offset	Range_CH1	ADC_RAW_CH1	rssi1_adc_comp_ch1	R
1	3285629730	1	0	1	0	0	0	0	0	0	0	52	3	597	545	2
2	2824571360	1	0	1	0	0	0	0	0	0	0	54	3	449	393	2
3	3285629730	2	0	2	0	0	0	0	0	0	0	52	3	598	545	2
4	2824571360	2	0	2	0	0	0	0	0	0	0	57	3	450	393	2
5	3285629730	3	0	3	0	0	0	0	0	0	0	51	3	597	544	2
6	2824571360	3	0	3	0	0	0	0	0	0	0	55	3	449	394	2
7	3285629730	4	0	4	0	0	0	0	0	0	0	51	3	598	544	2
8	2824571360	4	0	4	0	0	0	0	0	0	0	55	3	450	394	2
9	3285629730	5	0	5	0	0	0	0	0	0	0	51	3	597	544	2
10	2824571360	5	0	5	0	0	0	0	0	0	0	56	3	450	393	2
11	3285629730	6	0	6	0	0	0	0	0	0	0	51	3	598	544	2
12	2824571360	6	0	6	0	0	0	0	0	0	0	54	3	450	393	2
13	3285629730	7	0	7	0	0	0	0	0	0	0	52	3	597	545	2
14	2824571360	7	0	7	0	0	0	0	0	0	0	54	3	450	393	2
15	3285629730	8	0	8	0	0	0	0	0	0	0	51	3	597	544	2
16	2824571360	8	0	8	0	0	0	0	0	0	0	56	3	449	393	2
17	3285629730	9	0	9	0	0	0	0	0	0	0	51	3	597	544	2
18	2824571360	9	0	9	0	0	0	0	0	0	0	56	3	450	393	2
19	3285629730	10	0	10	0	0	0	0	0	0	0	52	3	597	545	2
20	2824571360	10	0	10	0	0	0	0	0	0	0	56	3	450	394	2
21	2824571360	11	0	11	0	0	0	0	0	0	0	56	3	450	393	2
22	2824571360	12	0	12	0	0	0	0	0	0	0	56	3	450	393	2
23	2824571360	13	0	13	0	0	0	0	0	0	0	56	3	450	393	2
24	2824571360	14	0	14	0	0	0	0	0	0	0	56	3	450	393	2
25	2824571360	15	0	15	0	0	0	0	0	0	0	55	3	450	394	2
26	2824571360	16	0	16	0	0	0	0	0	0	0	56	3	449	393	2
27	2824571360	17	0	17	0	0	0	0	0	0	0	56	3	450	393	2
28	2824571360	18	0	18	0	0	0	0	0	0	0	56	3	450	394	2
29	2824571360	19	0	19	0	0	0	0	0	0	0	56	3	450	393	2
30	2824571360	20	0	20	0	0	0	0	0	0	0	56	3	450	393	2

Abbildung E.12.: Exemplarischer Inhalt eines Textdokumentes des RSSI-Measurement Setups

F. Informationen zur beigefügten CD

Auf der beigefügten CD ist die folgende Ordnerstruktur mit den beschriebenen Inhlaten zu finden:

- *00_Schriftstück*
- *01_Anforderungsdokumentation*
  - Lasten- und Pflichtenheft (Exceltabelle)
- *02_Projektplan*
  - Vertikales Prototyping (Exceltabelle)
- *03_Hardwaredokumentation*
  - Bilder und PDF-Dokumente zur Hardwareplanung (PNG und PDF)
  - Pinout Hardware (PDF)
- *04_Softwaredokumentation*
  - *Projekt C#* - GUI-Projekt (Ordnerstruktur aus Visual Studio)
  - *Projekt S32K144* (Ordnerstruktur aus S32 Design Studio)
  - *Referenzsoftware*
    - *Lizard-S32K144* (Ordnerstruktur aus S32 Design Studio)
    - *JOKER-S32K144* (Ordnerstruktur aus S32 Design Studio)
    - *Fahrzeugschlüssel C++* (Ordnerstruktur aus Visual Studio)
    - *C# - Leere GUI* (Ordnerstruktur aus Visual Studio)
  - *Interaktion zwischen GUI und S32K144* (PDF)
  - *Programmherachieplan zum S32K144 Projekt* (Excelliste)
  - *Zustandsdiagramm zum S32K144 Menü* (PDF)
- *05_Testplan* (Excelliste)
- *06_Literatur* (PDF, Power Point Präsentation, Excellisten)

Die beigelegten CDs sind bei Prof. Dr. Heike Neumann (HAW-Hamburg) und Dirk Besenbruch (extern, NXP Semiconductors Germany GmbH) hinterlegt.

Literaturverzeichnis

[1] Philipp Liegl. *Was ist ein Tier-Supplier oder Tierlieferant? [online].* https://ecosio.com/de/blog/2017/03/10/Was-ist-ein-Tier-Supplier-oder-Tier-Lieferant/, März 2017.

[2] NXP Semiconductors. *NJJ29C0B - JOKER [Anwendungshinweis]*, Rev. 2.0 , Juni 2018.

[3] Renke Bienert. *RFID-Technik [Skript].* Vorlesungstermin 1 - Einführung, Sommersemester 2018.

[4] NXPSemiconductors. *TOKEN family [Interne Powerpoint-Präsentation].* Zusammenfassung der Schlüsselprotokolle LF und UHF, September 2018.

[5] NXP Semiconductors. *S32K1xx Series [Handbuch]*, Rev. 6.0, Dezember 2017.

[6] Henning Wolf. *Agile Softwareentwicklung [Buch].* dpunkt.verlag GmbH, 69115 Heidelberg, Ringstraße 19B, 2., aktualisierte und erw. Edition, 2011.

[7] AUTOSAR Confidential. *Layered Software Architecture [online].* https://www.autosar.org/fileadmin/user_upload/standards/classic/4-3/AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf, Dezember 2017.

[8] Reichelt Elektronik. *Produktangebot - LCD 204B BL [online].* https://www.reichelt.de/lcd-modul-4x20-h-4-8mm-bl-ws-m-bel-lcd-204b-bl-p53952.html.

[9] Joerg Becker. *Keyless Car Access: More Options for Drivers [online].* https://www.eetimes.com/document.asp?doc_id=1272676, Dezember 2004.

[10] Dr. Ing. Wolfgang Tobergte. *RFID-Technik [Skript].* Vorlesungstermin 2 - Induktiv gekoppelte Systeme, Sommersemester 2018.

[11] Prof. Maaß. *Vorlesung Prozessautomatisierung [Skript].* Vorlesungstermin 1 - Einführung in das Software-Engineering, Sommersemester 2018.

[12] NXP Semiconductors. *S32K144-EVB Quick Start Guide [Interne Präsentation]*, Rev. 4.1.

[13] ELECTRONIC ASSEMBLY. *BLUELINE - DOTMATRIX DISPLAYS 1x16 ... 4x40 [Handbuch].* Juli 2018.

[14] NXP Semiconductors. *Usage of the S32K Microcontroller to Control a RCI Device [Handbuch]*, Rev. 1.0, September 2018.

[15] NXP Semiconductors. *S32K144 OI Signal Description Input Multiplexing [Excelliste].*

[16] NXP Semiconductors. *NJJ29C0 Customer Evaluation Board [Schaltplan]*, rev. 1.2 edition, Juni 2017.

[17] NXP Semiconductors. *RCI Operation [Handbuch]*, Rev. 1.18, Dezember 2017.

[18] NXP Semiconductors. *NJJ29C0B - SPI Command Set [Handbuch]*, Rev. 2, Oktober 2017.

[19] NXP Semiconductors. *TOKEN Family RSSI Measurement and Calibration[Anwendungshinweis]*, Rev. 1.3, Februar 2018.

[20] NXP Semiconductors. *NJJ29C0B (JOKER) [Anwendungshinweis]*,Rev. 2.0, Juni 2018.

[21] Andreas Kühnel. *C# 6 mit Visual Studio 2015: das umfassende Handbuch [Buch].* Bonn: Rheinwerk, 7., aktualisierte und erweiterete Edition, 2016.

[22] Thomas Theis. *Einstieg in C# mit Visual Studio 2017 [Buch].* Bonn: Rheinwerk Verlag GmbH, 5., aktualisierte Edition, 2017.

[23] NXP Semiconductors. *S32K144EVB: S32K144 Evaluations Board [online].* https://www.nxp.com/products/processors-and-microcontrollers/arm-based-processors-and-mcus/s32-automotive-platform/s32k144-evaluation-board:S32K144EVB?lang_cd=en.

[24] NXP Semiconductors. *PKE System Design [Anwendungshinweis]*, Rev. 1.0 , Juni 2016.

Versicherung über die Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §16(5) APSO-TI-BM ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen habe ich unter Angabe der Quellen kenntlich gemacht.

Hamburg, 21. Februar 2019	
Ort, Datum	Unterschrift