OSTBAYERISCHE
TECHNISCHE HOCHSCHULE
REGENSBURG

BACHELORARBEIT

Martin Albert

Anomalieerkennung im Stromverbrauch von Steuergeräten im Kontext Testautomatisierung

6. Februar 2020

Fakultät:	Informatik und Mathematik
Studiengang:	Bachelor Technische Informatik
Abgabefrist:	08. Februar 2020
Betreuung:	Prof. Dr. Rudolf Hackenberg
Zweitbegutachtung:	Prof. Dr. Frank Hermann

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ADC Analog to Digital Converter

CAN Controller Area Network

CMOS Complementary MOS

DFT diskrete Fouriertransformation

DSL Domain Specific Language

ECU Electronic control unit

FFT Fast Fouriertransformation

GPIB General Purpose Interface Bus

GUI Graphical User Interface

IoT Internet of Things

MOSFET Metall-Oxid-Halbleiter-Feldeffekttransistor

RPI Raspberry Pi

SCPI Standard Commands for Programmable Instruments

TCP Transmission Control Protocol

USB Universal Serial Bus

VME Versa Module Europa

VML Voltage-Model Logic

1 Einleitung

In Zeiten der Digitalisierung werden immer mehr und mehr Prozesse, Informationen und Features aus der anlogen Welt in die digitale Welt geführt. Dies führt dazu, dass Anwendungen, einen immer höher werdenden Anspruch auf Ausfallsicherheit (Safety) und Schutz vor unautorisierten Zugriffen (Security) genügen müssen. Weil das Fehlen dieser Aspekte den Benutzer schädigen können, sind die Anforderungen an die Programme besonders hoch. Dabei ist nicht nur die Softwareentwicklung, sondern auch die Überprüfung dieser sehr aufwendig.
Um den personellen und zeitlichen Aufwand zu reduzieren, werden die Test automatisiert, d.h. vorher definierte Zustände werden programmgesteuert geprüft und ausgewertet. Die Testautomatisierung hat aber ihre Grenzen. So sind z.B. Programme mit Benutzereingaben, welche nicht simulierbar sind (z.B. Programme mit Graphical User Interface (GUI)’s ohne Testautomatisierungsvorkehrungen), nur teilweise oder sehr schwer testbar.
Dadurch entstand die Idee den Stromverbrauch als Rückgabewert zu nutzen. Der Stromverbrauch wird bereits in sogenannten Seitenkanalangriffen verwendet, um Sicherheitsschlüssel aus Geräten zu extrahieren. Dabei bedient man sich der Tatsache, dass dieselbe Operation zu ähnlichen Stromverbrauchswerten führen, Änderungen der Eingabedaten sich als Veränderung in diesem Stromverbrauch abbilden und auf den Schlüssel zurückgeführt werden können.
Nun stellt sich die Frage ob es möglich ist den gesamten Programmverlauf am Stromverbrauch zu erkennen und Anomalien zu detektieren. So könnten Fehler festgestellt werden, die vorher nicht in der Zustandsdefinition berücksichtigt worden sind. Vorstellbar ist auch Schadsoftware oder einen Ausfall von Komponenten eines Systems detektieren zu können.
Aus diesen Bestrebungen heraus ist diese Arbeit entstanden. Dadurch könnten die Safety und Security Aspekte einerseits besser getestet und nach der Auslieferung einfacher und vielleicht sogar durchgehender überprüft werden.

2 Zielsetzung

Das Ziel dieser Arbeit besteht darin herauszufinden ob es möglich ist Zustände von elektrischen Recheneinheiten anhand ihres Stromverbrauchs zu ermitteln und sie mit anderen Zuständen (z.B. einem Sollzustand) zu vergleichen, um so Anomalien zu erkennen. Des Weiteren soll ermittelt werden ob es möglich ist die Vergleichbarkeit im Kontext der Testautomatisierung einzubringen. Zum Abschluss soll die Theorie anhand eines praktischen Beispiels überprüft werden. Dafür müssen die nötigen Werkzeuge zur Datenerhebung und Auswertung selbst erstellt werden.
Um zu beantworten ob der Zustand eines Systems über seinen Stromverbrauch erkennbar ist, müssen folgende Teilfragen beantwortet werden:

- Ist der Stromverbrauch unterscheidbar und abhängig von den durchgeführten Operationen einer Recheneinheit?
- Kann vom Stromverbrauch auf den Zustand geschlossen werden?
- Können die Zustände verglichen und so Anomalien erkannt werden?
- Ist die Theorie auch praktisch umsetzbar?
- Kann das Ganze automatisiert werden?

Werden diese Fragen zielführend beantwortet, kann ein praktischer Versuch durchgeführt werden, um die Thesen zu bestätigen.

3 Theoretischer Rahmen und Definitionen

Innerhalb dieser Arbeit werden einige Begriffe genutzt, welche zu lang sind um sie bei der Benutzung zu erklären. Deswegen werden sie hier bereits vorweggenommen. Diese können aber auch bei Bedarf nachgelesen werden.

3.1 Visa Standard

VISA steht für ”Virtual Instrument Software Architecture” und definiert eine Architektur, welche aus mehreren Ressourcen besteht, die Funktionalitäten zusammenfassen (siehe [Foub]). Der VISA-Standard wird durch die IVI Foundation verwaltet. Durch einheitliche Interfacedefinitionen können, verschiedenste Systeme identisch angesprochen werden. Dabei werden folgende Funktionalitäten für die Implementierung der Architektur vorgeschrieben (siehe [Foub]):

- Auf- und Abbau von Sitzungen (Life Cycle Control)
- Verändern und abfragen der individuellen Ressourcencharakteristiken, genannt Attribute (Characteristic Control)
- Terminieren von laufenden Operationen (Asynchronous Operation Control)
- Beschränken der Ressource Zugriffe (Access Control)
- Durchführen von grundlegenden Kommunikationsdiensten (Operation Invocation and Event Reporting)

Im Standard beschrieben wird die Kommunikation über Universal Serial Bus (USB), Transmission Control Protocol (TCP), General Purpose Interface Bus (GPIB) und Versa Module Europa (VME)-Bus. Wobei auch jeweilige Kommunikationsstandards eingepflegt sind wie z.B. HiSLIP oder VXI-11 für die TCP-Kommunikation, USBTMC für USB oder VXI-11 für den VME-Bus.
Weiterhin werden von der IVI Foundation Bibliotheken der VISA-API für verschieden Programmiersprachen bereitgestellt. Für viele der nicht unterstützten Programmiersprachen sind aber Open-Source-Bibliotheken erhältlich. So z.B. ist PyVisa eine Bibliothek für Python welche das Kommunizieren mit VISA-Geräten ermöglicht.
VISA deklariert welche Funktionen ein Gerät zur Verfügung stellen muss, über welches Medium die Kommunikation stattfindet und wie die Art der Kommunikation verwendet wird. Als Standard wird aber keine Sprache festgelegt,

weswegen die Befehle selbst (welche über die VISA API versendet werden) von Gerät zu Gerät verschieden seien können.
Ein beliebter Befehlssatz-Standard ist dabei aber der Standard Commands for Programmable Instruments (SCPI)-99-Standard. Dieser wird z.B. von ”Rigol” oder ”Rohde & Schwarz” für ihre Messgeräte verwendet, welche den VISA-Standard implementieren.

3.2 SCPI

SCPI (Standard Commands for Programmable Instruments) ist ein Standard zur Definition von Kommandos und Kommandoschreibweisen (Syntax). Die Intension der Standarddefinition ist eine Aufwandsreduzierung bei der Entwicklung von Automatisierungssoftware. Bei Verwendung unterschiedlichem Testequipment erleichtert eine einheitliche Programmierumgebung die Steuerung der verwendeten Geräte, als auch die Verarbeitung der Daten.

3.3 Hammingdistanz/-gewicht

Die Hammingdistanz ist eine Abbildung, welche zwei Zeichenketten als Eingabe nimmt und die Anzahl der Unterschiede zwischen diesen Zeichenketten wiedergibt. Besonders in der Informatik wird die Hammingdistanz genutzt, um Bitfolgen zu vergleichen.
Beispiele von Hammingdistanzen:

$$H\left(Mathematik, Methemotik\right) = 2$$
$$H\left(11101, 10101\right) = 1$$
$$H(123, 123) = 0$$

Das Hamminggewicht ist ein Sonderfall der Hammingdistanz. Es beschreibt die Anzahl der Symbole die verschieden zum Nullelement der Zeichenfolge sind.

$$H_g\left(ABBACADDA\right) = 5 \quad (Nullemlement = A)$$
$$H_g\left(110100\right) = 3 \qquad (Nullemlement = 0)$$

4 Zustandsbestimmung von Systemen

Der Zustand eines Systems ist die Menge aller seiner derzeitigen Merkmale. Die Beschreibung von Zustände können sehr genau (z.B. Aufzählung von Spannungen jeder Leitung und der Widerstandsgrößen der Transistoren) als auch ungenau (z.B. System wird als Ein- oder Ausgehschalten beschrieben) sein. Eine einwandfreie Software durchläuft definiert Zustände, wobei jegliche Abweichung eine Anomalie darstellt. Um diese Anomalien mittels Zustandsvergleich zu erkennen, müssen die Zustände vorher bestimmt werden.

4.1 Konventionell

Das Agieren mit einer Recheneinheit (z.B. Homecomputer, Mikroprozessoren usw.) durch Benutzer erfolgt hauptsächlich über externe Peripheriegeräte wie Tastatur, Maus, Bildschirm und vielem mehr. Auch die Hersteller der Software/Geräte verwenden hauptsächlich externe Peripherie, während der Entwicklung einer Software. Dadurch werden die meisten Zustände wahrgenommen. Wenn ein Userinterface in der Software mitgeliefert wird, kann anhand diesem bedingt genau erkannt werden in welchem Zustand sich das Programm befindet. So ist es möglich an den Rückgaben zu erkennen, ob das Programm noch einwandfrei funktioniert oder ob Fehlfunktionen aufgetreten sind.
Ohne Userinterface kann anhand von den ausgegebenen Ergebnissen entschieden werden ob ein Fehler aufgetreten ist oder nicht. Somit kann auch hiermit nur bedingt der Zustand eines Systems bestimmt werden.
Zur Entwicklung der Software sind meist noch andere Werkzeuge im Einsatz, welche z.B. den Speicher und die im Programm verwendeten Variablen anzeigen können oder die derzeitige Operation darstellen. Diese sogenannten Debuggingtools erfreuen sich großer Beliebtheit unter Programmierern, da sie es einfach und ohne extra Aufwand ermöglichen Fehlerzustände und deren Ursache zu erkennen. Weiterhin ist es möglich für die Entwickler temporäre Teile in eine Software einzubauen, welche den derzeitigen Zustand des Programms genauer beschreiben (z.B. Ausgabe des Wertes einer Variable).
Durch diese Mittel ist es möglich mehr oder weniger genau auf die Zustände der Software zu schließen. Sobald aber kein Zugang zu den Entwicklertools besteht oder die Software nicht mehr verändert werden kann, ist die Zustandsbestimmung nur noch bedingt genau möglich. Dies kann verbessert werden indem unkonventionelle Methoden der Zustandsbestimmung verwendet wird.

4.2 Unkonventionell

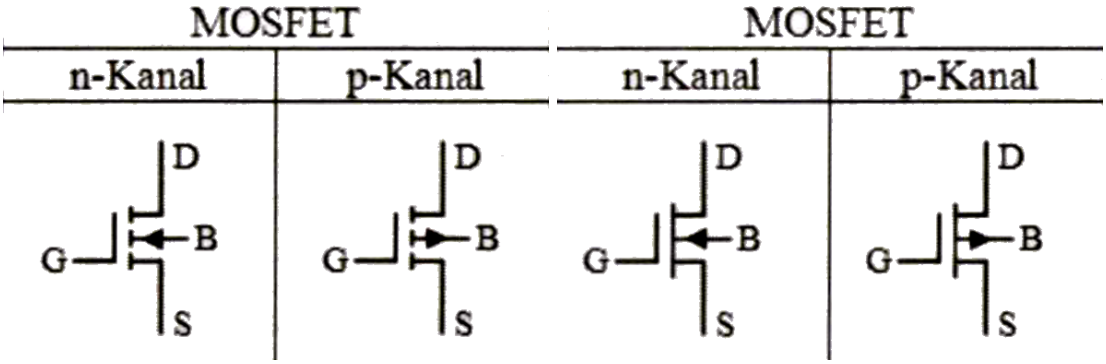
Zu den unkonventionellen Methoden der Zustandsbestimmung gehören jene, welche nicht von den Entwicklern angedacht waren. Oft sind sie als Side-Channel-Attacks (Seitenkanal Angriffe) bekannt.
So z.B. kann die Reaktionszeit eines Programms dafür genutzt werden Informationen über die Anzahl der Operationen zu erhalten. Anwendung findet dies z.B. in zeitbasierten Memorycompare Angriffen, womit Passwörter, Schlüssel und Hashes ermittelt werden können (ein Beispiel: [LFK19]).
Akustik kann auch einen Seiten-Kannal zu weiteren Informationen bilden. So ist es Forschern gelungen mithilfe der Vibration von Kondensatoren, Spulen und ihrem so erzeugten Ton, Timing Attacken auszuführen (zu sehen in [AS04]). Ein weitaus einfacheres Beispiel stellen die Lesevorgänge eines CD / DVD-Laufwerks dar. Das hörbare Drehen eines Laufwerks bedeutet, dass der Lesevorgang noch aktiv ist. Erst nach dem Beenden des Lesens kommt die Scheibe zum Stillstand.
Ein elektromagnetisches Feld wird von allen stromdurchflossenen Leitern abgegeben. Sie können gemessen und ausgewertet werden. So wurden schon Analysen zur Praktikabilität eines elektromagnetischen Seitenkanalangriffs durchgeführt, wobei man zum Schluss kam, dass sie praktisch anwendbar sind. (zu sehen in [WMM+17])
Eine weitere sehr beliebte Art des Side-Channel-Attacks sind Power Consumption Attacks (Stromverbrauchsattacken). Dabei werden Eingaben getätigt, welche die Verschlüsselung oder Entschlüsselung durch einen privaten Schlüssel auslösen während der Stromverbrauch dieses Vorgangs aufgezeichnet wird. Der Stromverbrauch kann an den vom Schlüssel abhängigen Stellen beobachtet werden und durch Simulationen Teil für Teil den Schlüssel bestimmen. Diese Methodik ist bereits eine etablierte Methode der Hardwareangriffe. Eine Übersicht der verschiedenen Arten von Power Consumption Attacks können in dem Paper ”An overview of side channel analysis attacks” [LCC08] noch gelesen werden.
Diese Methoden und Attacken haben meist die Intension Schlüssel, Passwörter und Hashes zu gewinnen. Diese Informationen sind aber nur ein Teil eines Zustandes, weswegen sich die Frage stellt ob es möglich ist mehr über den Zustand eines Gerätes herauszufinden als nur einzelne verarbeitete Daten. Dies wird im Folgenden für den Stromverbrauch als Seitenkanal ermittelt.

5 Analyse der Zustandsermittlung anhand des Stromverbrauchs

Operationen führen ein System von einem Zustand in den Nächsten. Während Operationen ausgeführt werden, befindet sich das System einem Operationszustand. Diese werden im Folgenden ermittelt.
Dafür widmen wir uns zuerst der Quelle des Stromverbrauchs.

5.1 Quelle des Stromverbrauchs in elektrischen Systemen

Für das Verständnis wie sich der Stromverbrauch von Recheneinheiten zusammensetzt, muss ein grobes Verständnis für den Aufbau und der Funktionsweise von Recheneinheiten vorhanden sein. Deshalb wird diese Thematik grob erklärt.
Moderne Prozessoren, Mikrocontroller und viele weitere Recheneinheiten heutzutage basieren auf Transistoren. Diese werden so verschaltet, dass sie die bekannten Funktionen liefern. Transistoren können in vielen verschiedenen Arten gebaut und verschaltet werden.
Ein Transistor (TRANSfer reSISTOR) ist ein variabler Widerstand, welcher durch eine angelegte Spannung oder einem Stromfluss die Größe des Widerstandes steuert. Insbesondere in der Schaltungstechnik wird dieser Widerstand so groß gewählt, dass kaum bis kein Strom fließt. Dadurch kann ein Transistor wie ein Schalter betrachtet werden. Ein Transistor, welcher bei anliegender Spannung die Verbindung zwischen Eingang und Ausgang unterbricht (entspricht einer Erhöhung des Widerstandes), wird als selbstleitender Transistor bezeichnet. D.h. bei nicht vorhandener Spannung leitet der Transistor (daher der Name). Komplementär existiert ein selbstsperrender Transistor, welcher bei anliegender Spannung den Stromfluss ermöglicht und bei fehlender Spannung die Verbindung zwischen Eingang und Ausgang sperrt. Weiterhin werden in der Metall-Oxid-Halbleiter-Feldeffekttransistor (MOSFET)-Familie Transistoren als p-Kanal und n-Kanal gebaut. Somit existieren vier verschiedene Typen an MOSFET's, wessen Schaltsymbole in Abbildung 1 zu sehen sind.
Unter den Transistoren sind MOSFET's die wohl weitverbreitetste Transistorgruppe, während Complementary MOS (CMOS) als der meist genutzt Logikstil zählt (siehe [SM07], S.23-24)).
Transistoren können so verschaltet werden, dass sie AND-, OR-, NOT-, NAND- oder NOR-Gatter ergeben, wie in Abbildung 2 zu sehen ist. Aus NOT-Gatter



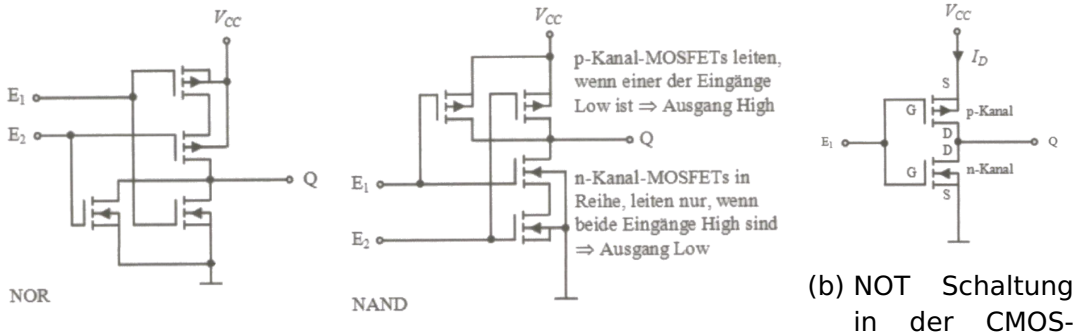
(a) MOSFET Schaltzeichen Selbstleitend

(b) MOSFET Schaltzeichen Selbstsperrend

Abbildung 1: Schaltzeichen für selbstleitende (a) und selbstsperrende (b) MOSFET (Transistoren), Quelle: [Sti09]

und einer beliebigen anderen Gatterart kann jedes andere Gatter gebildet werden. So ist es z.B. möglich mit einigen NOT's und AND's ein OR zu erstellen.
Aus diesen Grundgatter können dann alle möglichen Logikzellen gebaut werden, wie z.B. Addierer, Dividierer, Speicherzellen usw. Wobei man zwischen den zustandslosen (basic) und den zustandbehafteten (sequentiellen) Zellen unterscheidet. Die basic Logikzellen Schalten unabhängig von dem letzten Zustand bei gleicher Eingabe immer in den gleichen Ausgang (z.B. Addierer). Sequentielle Zellen beachten den Zustand, welchen sie auch vor der Operation hatten (z.B. Flip-Flops oder Speicherzellen). Da jede Operation aus diesen Grundgatter gebaut werden, ist es möglich den Stromverbrauch beim Berechnen von Werten auf Prozessoren, auf diese Logikgatter zurück zu führen. [1]
Ein NOT-Gatter(Abbildung 2(b)), bestehend aus zwei Transistoren, invertiert den Eingang E1 und gibt es an Ausgang Q aus. Ein NAND-Gatter und NOR-Gatter kann in Abbildung 2(a) gesehen werden. Durch Verschaltung eines NAND- und einem NOT-Gatter ergibt sich ein AND-Gatter. Ein OR-Gatter wird analog mit einem NOR-Gatter erstellt, welches durch ein NOT-Gatter erweitert wird. Tabelle 1 stellt Eingaben und Ausgaben der jeweiligen Gatter dar.

[1] Innerhalb dieser Arbeit wird der der Voltage-Model Logic (VML)-Stil betrachtet. In diesem Stil werden die Eingangsspannung und der GROUND als Logikwerte genutzt. Außerdem wird sich hauptsächlich darauf gestützt, dass ein anliegendes HIGH einer logischen 1 entspricht während ein anliegender LOW als logische 0 gesehen wird. In Schaltungen, in denen des genaue Gegenteil vorherrscht, können dieselben Aussagen über den Stromverbrauch gemacht werden. Nur ist zu beachten, dass eine 0 für ein HIGH steht und eine 1 für ein LOW und somit der Stromverbrauch durch die Daten genau andersherum verhält.



(a) NAND und NOR Schaltung in der CMOS-Technik

(b) NOT Schaltung in der CMOS-Technik

Abbildung 2: NAND, NOR (a) und NOT (b) Schaltungen, Quelle: [Sti09]

E1	0	1
	1	0

(a) NOT Gatter

E2 \ E1	0	1
0	1	0
1	0	0

(b) NOR Gatter

E2 \ E1	0	1
0	1	1
1	1	0

(c) NAND Gatter

E2 \ E1	0	1
0	0	1
1	1	1

(d) OR Gatter

E2 \ E1	0	1
0	0	0
1	0	1

(e) AND Gatter

Tabelle 1: Schalttabelle für die Grundgatter

Anhand eines NOT-Gatters wird nun im Folgenden der Stromverbrauch von Schaltkreisen erklärt. Der CMOS-Logikstil erlaubt es diese Schlussfolgerungen auf alle anderen Gatter zu übertragen.
Jedes einzelne Gatter verbraucht Strom und wandelt dies u.a. in Hitze um, welche hauptsächlich nach außen abgegeben wird. Der Stromverbrauch eines Gatters kann aufgeteilt werden in den statischen und dynamischen Teil. [SM07]
Der statische Stromverbrauch entsteht durch den sogenannten Kriechstrom, welcher durch den Widerstand der Transistoren sickert. Da Transistoren nur Widerstände herstellen und nicht die Verbindung unterbrechen und unter

CMOS die Verbindung zwischen GROUND und der Stromquelle nur durch diese Transistoren getrennt werden, entsteht der Kriechstrom. Dieser Verbrauch geschieht kontinuierlich.
Der dynamische Teil des Verbrauchs entsteht einerseits durch kurzzeitige Kurzschlüsse beim Schalten und durch das Aufladen des Ausgabekondensators. Während die Gatter so gebaut sind, dass sie eigentlich nie den GROUND und die Spannungsquelle direkt verbinden sollen, gibt es eine kurze Zeit, beim Schalten, in der es doch passiert. Im Falle der NOT-Schaltung passiert das, wenn man beim Eingang E1 von der logischen 0 auf 1 wechselt. Dadurch wird der p-Kanal (mit dem selbstleitenden Transistor) gesperrt und der n-Kanal (mit dem selbstsperrenden Transistor) entsperrt. Dieser Sperr- und Entsperrvorgang passiert aber nicht sofort. Die Transistoren brauchen eine bestimmte Zeit, um ihre Widerstände auf oder abzubauen. In dieser Zeit leiten sowohl p-Kanal als auch n-Kanal Transistoren, was zu einem kurzweiligen Kurzschluss führt und somit zu einem Stromfluss und einem Stromverbrauch. Auch bei der Transition von 1 auf 0 tritt dieser Kurzschluss auf. Der Ausgabekondensator liegt am Ausgang des Gatters an und wird aufgeladen, wenn der Ausgang auf eine logische 1 gesetzt wird. Bei einer logischen 0 wird der Kondensator entladen indem er geerdet wird. Somit tritt ein Stromfluss beim Aufladen des Kondensators auf, wenn dieser zuvor ungeladen war.
Folglich führen Transitionen am Ausgang einer NOT-Schaltung zu folgenden Stromverbräuchen (in Tabelle 2).

Transition bei Ausgabe Q	Stromverbrauch	Art des Stromverbrauchs
$0 \rightarrow 0$	P_{00}	statisch
$0 \rightarrow 1$	P_{01}	statisch + dynamisch
$1 \rightarrow 0$	P_{10}	statisch + dynamisch
$1 \rightarrow 1$	P_{11}	statisch

Tabelle 2: Stromverbräuche bei Transitionen der Ausgabe eines NOT-Gatters, Quelle: [SM07] (Kopf wurde verändert)

Dabei ist zu beachten, dass P_{01} und P_{10} immer viel größer sind als P_{00} und P_{11} [SM07].
Nun können wir daraus schließen, dass Operationen direkt im Stromverbrauch resultieren. Besonders ausschlaggebend sind Operationen, bei denen die Ausgabe der Logikgatter von 0 auf 1 oder 1 auf 0 schaltet. Die beiden anderen Transitionen verbrauchen auch Strom, der sich aber nur minimal auf den gesamten Stromverbrauch auswirkt.

5.2 Störgrößen

Versucht man Messungen von gleichen Messgrößen zu reproduzieren, kommt man schnell zu dem Schluss, dass diese sich immer wieder Unterscheiden. Das liegt an Störgrößen die sowohl extern als auch intern des Versuchs verursacht werden. Deswegen sollten diese Messfehler identifiziert und so weit wie möglich umgangen oder abgeschwächt werden.
Innerhalb des Buches ”Power Analysis Attacks” [SM07] werden die Störgrößen in Fünf Punkte aufgeteilt:

- Rauschen des Netzteils
- Strahlungsemissionen
- Quantisierungsfehler
- Emissionen auf verbundene Leitungen
- Taktgeberungenauigkeit

Während der Autor sich in diesen Punkten hauptsächlich auf die Stromverbrauchsanalyse als Seitenkanalattacke konzentriert, sollen hier die Punkte in Hinsicht der Stromverbrauchsanalyse zur Zustandserkennung neu revidiert werden.

Rauschen des Netzteils
Spannungs- und Stromquellen können verschieden gewählt werden. Anforderungen wie Maximallast, Kosten und Stabilität müssen aufeinander abgewägt werden. Netzteile, Batterien, Computerschnittstellen, Mikrocontroller selbst und viele weitere Elektronikgeräte können als Quelle verwendet werden. Dabei stellen sie die Spannung mehr oder weniger kontinuierlich und präzise dar. Bei einer Computerschnittstelle, wie USB, kann es passieren, dass die Spannung abhängig von der Eigenlast ausgeben wird, was bei der Messung zu Spannungsschwankungen führen kann. Andere Stromquellen, die neben dem Bereitstellen des Stroms andere stromverbrauchende Funktionen haben, sind meist anfällig für Spannungsschwankungen und sollten vermieden werden.
Deswegen sind Quellen, welche eine möglichst konstante und rauschfreie Spannung ausgeben, besonders wichtig. Darunter zählen Netzgeräte und Batterien (im Falle einer Kurzzeitmessung).

Strahlungsemissionen
Elektromagnetische Strahlung umgibt uns heutzutage überall. Sie wird von jedem elektronendurchflossenen Material ausgestrahlt und von jedem leitenden Material als Spannung und Stromstärke aufgenommen. Dazu zählen auch Leitungen der zu messenden Gegenständen. Deswegen können Geräte in der Nähe mit ihrer Strahlung die Messungen verfälschen und sollen aus der unmittelbaren Nähe des Geräts entfernt werden.

Quantisierungsfehler
Der Quantisierungsfehler bezieht sich auf die Rundungsfehler von digitalen Messgeräten. Da digitale Messgeräte nur endlich viel Speicher für die Messung haben, müssen Werte früher oder später auf die vorhandene Bitanzahl gerundet werden. Je hoher die Auflösung des Messgeräts ist, desto kleiner ist der Quantisierungsfehler.
Dieser Fehler tritt nur auf, wenn ein digitales Messgerät verwendet wird.

Emissionen auf verbundenen Leitungen
Alle Geräte, die im Messstromkreis eingebunden sind, verändern die Größen der Messung. Sogar die Messgeräte selbst modifizieren die Messgröße.
Man sollte versuchen so wenig Geräte wie möglich in den Messstromkreis einzubringen. In dem Buch ”Power Analysis Attacks” [SM07] wird sogar davon gesprochen irrelevante Komponenten einer zu testenden Einheit zu extrahieren und nicht zu bemessen. In unserem Fall der Zustandsermittlung wäre das aber konterproduktiv, da im Konzept einer Einheit nie ganz klar ist welche Teile vom System genutzt werden und welche nicht.
Nur falls genau bekannt ist, welche Komponenten irrelevant sind, sollten diese vom Messstromkreis entfernt werden. Sonst kann das zu einer Fehlerquelle bei der Zustandsermittlung führen.

Taktgeberungenauigkeit
Ungenauigkeiten eines Taktgebers im Messgerät als auch die im zu testenden Gerät haben die gleichen Folgen. Beide Male verzerrt sich die zeitliche Sicht auf die Daten und so können Muster sowie Frequenzen von Messung zu Messung unterschiedlich erscheinen.
Zu verhindern ist dieses Phänomen meist nur auf der Seite des Messgeräts, indem eines benutzt wird, welches kaum Frequenzschwankungen im Takt aufweist. Das zu testende Gerät kann auch verbessert werden, wenn die Taktquelle selbst gewählt werden kann.

5.3 Messung des Stromverbrauchs über die Zeit

Für das Messen des Stromverbrauchs sind verschiedenste Messgeräte verfügbar. Dabei können sie direkt oder indirekt den Stromverbrauch anzeigen. Die Auswahl reicht von analogen Nadelmessgeräten bis hin zu digital Hightech-Oszilloskopen. Soll das Messinstrument indirekt den Stromverbrauch wiedergeben, wird die Spannung und Stromstärke vermessen, welche mit der Formel 1 zum Stromverbrauch umgerechnet wird. Da diese Messgeräte auch verschiedene Vor- und Nachteile bieten, lohnt es sich sie genauer zu analysieren, um ein passendes Gerät für die Anomalieerkennung zu finden.

$$P = U * I \tag{1}$$

Messung der Spannung
Die Spannung kann durch elektromechanische oder elektronische Messgeräte ermittelt werden. Im Falle eines elektromechanischen Geräts, wird eine dreh gelagerte Spule in ein Magnetfeld gelegt. Durch die Spule fließt der zu messende Strom. Aufgrund der nun vorhanden Elektromagnetnischenkräfte richtet sich die Spule neu aus. Durch einen Zeiger auf der Spule kann nun linear der Ausschlag abgelesen werden, welcher die Stromstärke angibt (siehe Abbildung 3). Durch Umrechnung der Stromstärke mit der Ohm'schen-Formel (Formel 2) kann die Spannung ermittelt werden.

$$R = \frac{U}{I} \qquad \Leftrightarrow \qquad U = R * I \qquad \Leftrightarrow \qquad I = \frac{U}{R} \tag{2}$$

Da die Ausgabe des Ergebnisses analog erfolgt, ist es schwer diese Messmethode in Kontext einer Automatisierung in ein Testsystem einzubauen. Auch die relativ große Trägheit des Messsystems und der relativ große Auslesefehlerbereich sind ungünstig.
Die elektronischen Messgeräte (auch digitale Messgeräte genannt) verwenden meist eine Verstärkerschaltung, welche die anliegende Spannung verstärkt oder abschwächt und an einen Analog to Digital Converter (ADC) weitergibt. So kann ein großes Spektrum an Spannungsleveln mit einem begrenztem ADC abgedeckt werden.
Wird ein digitales Messgerät durch das Erheben und temporäre Abspeichern von Messdaten in gleichmäßig großen Zeitintervallen erweitert, so spricht man von einem Oszilloskop. Ein Oszilloskop ist somit in der Lage Spannungen oder Stromstärken über die Zeit darzustellen. Moderne Oszilloskope können diese Daten auch digital abspeichern und/oder übermitteln.
Als Ergebnis der Spannungsmessungen erhält man immer Größen, welche in der Messtechnik mit den Einheiten "Volt" versehen werden.

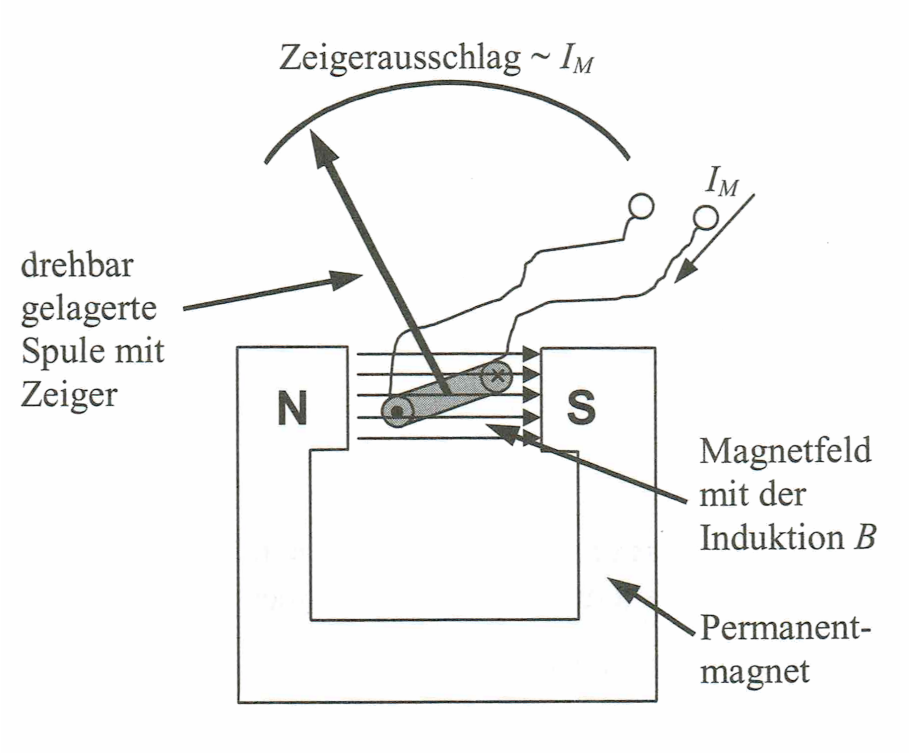


Abbildung 3: Grundprinzip eines Drehspulmesswerks (Stark vereinfacht), Quelle: [Par14] S.103

Messung der Stromstärke

Das Messen von Strömen kann sowohl analog als auch digital geschehen. Ein analoges Messgerät hat dieselbe Funktionsweise wie die eines elektromechanischen Spannungsmessgeräts, wobei der Schritt der Spannungsumrechnung wegfällt und das Messgerät in Reihe geschallten wird.
In der digitalen Welt sind zwei Messverfahren von Bedeutung. Zum einen erfolgt die Messung der Spannung durch Verstärkerschaltungen, ADCs, einem Shuntwiderstand und der Ohm'schen Formel (Formel 2). Dabei ist die Größe des Widerstands passend zu wählen. Ein niedrig ohmiger Widerstand ändert an der Ausgabe der Spannung wenig, auch wenn der Stromverbrauch höher ist. Dies hat zur Folge, dass die zu messenden Geräte keine Spannungsschwankungen kompensieren müssen. Aber durch diese niedrigen Spannungsunterschiede (gemessen vor und nach dem Widerstand) kann die Stromstärke zum Teil nur ungenau ermittelt werden. In manchen Fällen ist das Rauschen auf der Leitung so groß, dass man nicht zwischen Rauschen und dem eigentlich Stromfluss unterscheiden kann.

Zum anderen kann die Stromstärke indirekt über die entstehenden elektromagnetischen Felder beim Durchfließen der Leitungen gemessen werden. Dafür werden meist Messzangen (z.B. Abbildung 4), welche die elektromagnetischen Kräfte mithilfe eines Hallsensors oder der Kompensationsmethode in Stromstärke umwandeln und anzeigen. Nur wenn die einzelne stromführende Leitung, alleine umfasst werden kann, ist die Messung mit der Messzange möglich.

Abbildung 4: Ein Multimeter mit Stromzange für Gleichstrom und Wechselstrom, Quelle: de.wikipedia.org/wiki/Zangenstrommesser, Letzter Besuch: 23.11.2019

Diese beiden digitalen Methoden finden auch Anwendung in Oszilloskopen. Sonden zum Messen der elektromagnetischen Kräfte werden für die meisten Oszilloskope angeboten. Sie sind jedoch mit der Anschaffung oft teurer als das Grundgerät und somit nicht für alle Messprojekte tragbar. Der Vorteil dieser Messmethode ist die minimale Auswirkung auf das Messsystem und eine passable Messgenauigkeit.
Im Gegensatz dazu stehen Messungen mithilfe von Spannungssonden und einem Shuntwiderstand. Widerstände sind in der Regel relativ günstig, jedoch muss die Schaltung für die Messung angepasst werden. Der Shuntwiderstand wird in Reihe zur Spannungsquelle eingebaut und verändert so die Spannung bei einem Stromverbrauch. Die Präzision mit der gemessen werden kann, hängt von der Größe des verwendeten Shuntwiderstands und somit von der Robustheit der zu testenden Einheit ab. Kann dieses Objekt in einem großen Spannungsbereich arbeiten, so kann ein großer Shuntwiderstand gewählt und somit die Stromstärke genauer ermittelt werden.
Als Ergebnis der Messungen erhält man immer Größen, welche in der Messtechnik mit den Einheiten ”Ampere” versehen werden.

Messung über die Zeit
Für die Auswertung der Messergebnisse sind Messungen über die Zeit Notwendig, wobei die Abtastrate besonders relevant ist. Die Abtastraten von Multimeter sind meist viel niedriger als die von Oszilloskopen. Ändert sich der Wert (die Spannung oder die Stromstärke) zwischen zwei Abtastungen, ermittelt das Messgerät meist den Mittelwert des Wertes. Somit ist es wichtig die Abtastrate groß genug zu wählen, um nicht zu viele Daten zu verlieren.
Möchte man eine feste Abtastrate so muss eine Vorrichtung dafür im Messgerät verbaut sein oder gebaut werden. Oszilloskope haben so eine Vorrichtung und sind darauf ausgelegt hohe Abtastraten vorzuweisen, weswegen sie sich besonders gut für unser Vorhaben eignen.

Messung des Stromverbrauchs
Mithilfe von Geräten zur Messung der Spannung und der Stromstärke kann der Stromverbrauch als Leistung ermittelt werden. Sowohl zur Spannungsmessung als auch der Messung der Stromstärke hat sich ein Oszilloskop als die beste Option herausgestellt. Bei der Stromstärkemessung empfiehlt sich die Methode des Shuntwiderstands, wenn der Kostenfaktor eine Rolle spielt.
Nach dem Messen kann über jeden Zeitpunkt/Messpunkt aus der Spannungsdifferenz U_d und der Größe des Shuntwiderstands R_s , die Stromstärke I ausgerechnet werden (siehe Formel 2).
Das Verrechnen der erzielten Stromstärke und Spannung, zu identischen Zeitpunkten, ergibt dann einen zeitlichen Verlauf der absorbierten Leistung (siehe zum berechnen Formel 1).

5.4 Zustandsbestimmung und Anomalieerkennung

5.4.1 Zustandsbestimmung

Recheneinheiten benutzen Rechenoperationen, welche abhängig von der Eingabe und der Rechenoperation selbst, verschieden große Stromverbräuche erzeugen [KJJR11](für mehr Details siehe Kapitel 5.1).
Es lässt sich von einer Operation mit ihren Parametern und dem derzeitigen Ausgangspunkt der Ausgabe eindeutig auf eine Leistung schließen. Umgekehrt kann aber ein Stromverbrauch nicht direkt zu einer Operation mit den gleichen Parametern zurückgeführt werden, sondern nur auf eine Sammlung

von Operation mit Parametern und Ausgangspunkten. Würde man das ganze Mathematisch beschreiben, gilt die Abbildung zwischen den Mengen der $Ausgangspunkte A \times Operoationen O \times Parametern P$ und der Mengen der möglichen Stromverbräuchen S als Surjektiv (wobei $|A \times O \times P| > |S|$ gilt).
Daraus kann gefolgert werden, dass aus den Stromverbrauchswerten nicht alle Informationen wiederhergestellt werden können. Trotzdem ist es möglich die Stromverbrauchswerte zu vergleichen und zu deuten. So kann bei übereinstimmenden Verbrauchswerten nicht entschieden werden, ob keine Veränderung vorliegt. Wenn aber Abweichungen vorliegen kann darauf geschlossen werden, dass mindestens eine Operation, Parameter oder Ausgangspunkt verändert wurde.
Zustände sind das Ergebnis aller durchgeführten Operation eines Systems. Eine Operation kann (muss aber nicht) den derzeitigen Zustand eines Systems verändern. Während eine Operation stattfindet ist das System auch in einem Zustand, diesen nennen wir Operationszustand. Wir konzentrieren uns hauptsächlich auf diese Operationszustände, da die Bestimmung der Zustände vor und nach den Operationen nicht eindeutig durch den Stromverbrauch herzuführen sind.
Messen wir den Stromverbrauch während den Operationen und können ein Muster feststellen, welches auch bei einer anderen Operation zu finden ist, werden für uns beide Operationszustände in einem zusammengefasst. Es ist nicht möglich ohne weitere Informationen festzustellen, welche der beiden Operationen ausgeführt wurde. Das nachfolgende Beispiel soll dies verdeutlichen:

Dieses Beispiel ist frei erfunden und soll nur verdeutlichen wie ungenau wir Zustände bestimmen können.
Die beiden Operationen Addieren und Multiplizieren werden jeweils auf zwei baugleichen Systemen ausgeführt. Als Parameter erhalten die Operationen zum einen $Parameter_1 = 4$ und $Parameter_2 = 4$ für das Addieren und als $Parameter_1 = 2$ sowie $Parameter_2 = 4$ für das Multiplizieren. Beide Male ist das Ergebnis 8.
In beiden Fällen erhalten wir das gleiche Muster und den gleichen Durchschnittsverbrauch. Sind nur die Stromverbrauchswerte bekannt, müssten also die zwei Operation in den gleichen Operationszustand eingeordnet werden.

Wie aus dem Beispiel ersichtlich ist, kann die Zustandsbestimmung nur in einem limitierten Rahmen erfolgen.

Messungen zur Zustandsbestimmung untersuchen

In einer Messung werden ein oder mehrere Tests ausgeführt. Innerhalb dieser Tests wird versucht das Gerät durch Interaktionen in Zustände zu versetzen. Diese Zustände bestehen aus vielen einzelnen Operationen, die meist in Schleifen ablaufen. Interagiert man mit dem System so, dass sich ein Wechsel in eine andere Schleifen mit einer anderen Reihenfolge von Operation ergibt (Zustandswechsel), so ist es sehr wahrscheinlich, dass sich das Muster des Stromverbrauchs ändert.

Deswegen sollte nicht die gesamte Messung auf einen Zustand untersucht werden. Stattdessen wird der Stromverbrauch auf der Zeitachse unterteilt und jeder Teil einzeln untersucht. Die Größe der Teile ist entscheidend um gute Ergebnisse zu erhalten. Wählt man sie zu groß so überlappen sich Zustände, was zu fehlerhaften Analysen führt. Sind die Zeitintervalle zu klein, können Teile des Musters nicht mit eingeschlossen sein oder die Muster wiederholen sich nicht oft genug, um eine genügende Konfidenz zur Mustererkennung zu erhalten.

Werden die Zeitintervalle gleichgroß gewählt, sind die Schritte im Folgenden einfacher. Mit ungleich großen Intervallen müssen weitere Faktoren beachtet werden, die bei der Fouriertransformation näher beschrieben sind.

Die so entstandenen Teilmessungen werden nun auf die Faktoren Durchschnittsverbrauch und enthaltene Frequenzen reduziert.

Der Durchschnittsverbrauch ist relativ einfach durch das arithmetische Mittel (Formel 3) zu bestimmen.

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{3}$$

Frequenzen können mithilfe der Fourieranalyse bestimmt werden. Innerhalb der Fourieranalyse werden Fouriertransformationen beschrieben, die sowohl zeitkontinuierliche als auch zeitdiskrete Signal in das Frequenzspektrum transformieren. Bei einem zeitkontinuierlichen Signal liegt die Zeitachse im reellen Raum $\mathbb{R}$. Da Messungen vorliegen welche diskret abgetastet werden, wird sich mit der Fouriertransformationen für den zeitdiskreten Fall beschäftigt, welche die Zeitachse mit der Menge der ganzen Zahlen $\mathbb{Z}$ verwendet.

Darunter zählt die diskrete Fouriertransformation (DFT) oder Fast Fouriertransformation (FFT), wobei zweiteres nur eine verbesserte und schnellere Variante der DFT ist. Durch Benutzung dieser diskreten Fouriertransformation tritt ein Nebeneffekt auf. Das resultierende Spektrum ist in der Mitte gespiegelt, weswegen Frequenzen, welche über die Hälfte der Abtastrate hinaus gehen, ignoriert werden können.

In Abbildung 5 ist ein Teil einer Fouriertransformation zu sehen. Die Indexe der Spitzen geben an welche Frequenzen vorzufinden sind. Die Höhe der

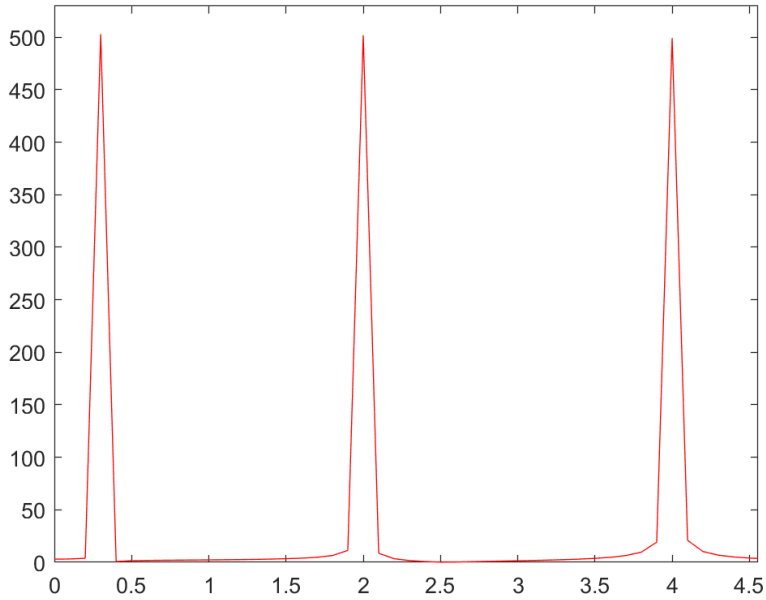


Abbildung 5: Das Absolute der Fouriertransformation für die Gleichung $f(x) = cos(2 * pi * x * 0.3) + cos(2 * pi * x * 2) + cos(2 * pi * x * 4)$ wobei $x \in [1 : 4, 5]$ mit einer Abtastrate von 100 Hz

Spitzen steht in einer direkten Proportionalität zur Amplitudenhöhe der Eingabefrequenzen und der Länge des Zeitraums, in welchen die Frequenzen auftreten. Des Weiteren nimmt die Abtastrate auch einen großen Einfluss auf die Höhe der Spitzen.
Die drei Faktoren Amplitudenhöhe, Messzeitraum und Abtastrate verändern also die Höhe der Spitzen und somit der Unterscheidbarkeit zwischen Rauschen und den gewollten Frequenzen.
Die Amplitudenhöhe des Signals kann nicht von uns verändert werden. Der Messzeitraum und die Abtastrate aber schon. Die Abtastrate sollte so hoch wie möglich gewählt werden, da so große lokale Maxima entstehen. Dabei ist aber zu beachten, dass mit steigenden Anforderungen das Messequipment meist teurerer wird. Genauso werden Recheneinheiten benötigt die stark genug sind, die gemessenen Daten in vertretbarer Zeit zu verrechnen. Die Größe der gewählten Zeitintervalle wirkt sich sowohl auf die Höhe als auch die Breite der entstehenden Berge aus. Verwendet man also für die Teile der Messungen unterschiedlich lange Zeitintervalle werden die Spitzenhöhen verschieden groß sein, selbst wenn das gleiche Signal analysiert wird. Sind die Berge sehr dünn, so kann sehr genau gesagt werden, welche Frequenzen vor-

handen sind. Durch eine große Breite können Rauschfrequenzen angehoben werden, was dazu führt, dass die minimale Höhe der akzeptierten Spitzen angehoben werden muss und so Frequenzen wegfallen, die normalerweise erkannt werden könnten. Um dünne Ausschläge zu erhalten, müssen also die Zeitintervalle erhöht werden, was aber auch dazu führt, dass die Wahrscheinlichkeit steigt in einem Teil mehrere Zustände zu messen. Somit kann durch das verlängern der Zeitintervalle die Konfidenz der gefundenen Frequenzen erhöht werden, während die Wahrscheinlichkeit, dass mehrere Zustände in einem zusammengefassten Zustand interpretiert werden, steigt. Umgekehrt führen kurze Zeitintervalle zum unwahrscheinlicheren Bestimmen der richtigen Frequenzen, aber zu einer höheren Wahrscheinlichkeit dass nur einzelne Zustände gemessen werden. Dieser Trade-off kann auf die gleiche Oberklasse von Trade-offs zurückgeführt werden, unter welcher auch die heisenbergsche Unschärferelation unterliegt.
Mithilfe des Durchschnittsverbrauchs und den erkannten Frequenzen kann also der Stromverbrauch charakterisiert und verglichen werden. Setzt man den Charakter des Stromverbrauchs einem Zustand gleich, erhält man Zustände die verglichen werden können.
Allein durch diese Information können zwar Zustände unterschieden, aber keine genauere Bedeutung gegeben werden. Ob es sich z.B. um einen Zustand handelt in dem kryptographische Inhalte entschlüsselt werden, kann nur durch weitere Informationen herausgefunden werden.
Als Ergebnis erhält man also eine Liste von Teilintervallen, welche jeweils die gefundenen Frequenzen auflisten und den durchschnittlichen Stromverbrauch angeben. Genau auf diese Aspekte werden die gemessenen Signale reduziert. Somit Charakterisieren wir auch die Zustände mit diesen Variablen.

5.4.2 Anomalieerkennung

Um Anomalien erkennen zu können, müssen Zustände anhand des Stromverbrauchs erkannt werden und mit den erwarteten Zuständen verglichen werden. Deswegen muss erst eine Stromverbrauchskennlinie für den Normalzustand erstellt werden, mit welchem verglichen wird. Zum Erstellen eines vergleichbaren Verbrauchs gibt es mehrere Ansätze.

Normalstromverbrauch ermitteln

Einerseits kann der Normalstromverbrauch mithilfe von Simulationen ermittelt werden. Simulationen des Stromverbrauchs von digitalen Schaltungen werden im Design Prozess eines elektronischen Geräts benutzt, um herauszufinden, ob die Anforderungen erfüllt werden. (siehe [SM07]) Dies kann mit verschiedenen Methoden geschehen. Dabei müssen die Präzision und der Rechenaufwand der Simulation aufeinander abgeschätzt werden. Denn Methoden großer Genauigkeit, bringen meist hohe Rechenaufwände mit sich. Für eine vollständige Simulation des Stromverbrauchs ist es aber nötig alle Softwareteile, Transistoren und/oder Logikzellen zu kennen, was meist nur den Entwicklern der Digitalschaltung vorbehalten ist.
Weitere Simulationsarten wie das Simulieren mithilfe des Hammingdistanzmodells oder des Hamminggewichtsmodell werden gerne für Seitenkanalangriffe verwendet, da sie wenig bis keine Informationen über den inneren Aufbau des Geräts benötigen.
Das Hammingdistanzmodell kommt eigentlich aus der Gruppe der Simulationen, welche eine vollständige Transistorliste und die Software benötigen. Aber durch Abwandlung des Modells auf eine simplere Ansicht, kann man sich auf die wichtigsten Komponenten eines Systems konzentrieren (wie z.B. einem Datenbus und interessante Teile der Software) und nur diese simulieren. Dadurch entsteht keine präzise aber eine aussagekräftige Simulation. Wenn also die Software bekannt ist, kann sie mit diesem Modell simuliert werden.
Dabei werden die Transitionen $0 \rightarrow 1$ und $1 \rightarrow 0$ in allen Gattern gezählt und als Stromverbrauch interpretiert. Somit nimmt dieses Modell die Hammingdistanz als Stromverbrauch an.
Damit können Strukturen und Frequenzen sehr genau erkannt werden, aber der reale durchschnittliche Verbrauch des Gerätes bleibt ohne Transistorliste ungewiss.
Das Hamminggewichtsmodell ist dem Hammingdistanzmodell sehr ähnlich. Dieses Modell benötigt keine Informationen über die verwendete Hardware und sieht das Hamminggewicht als Stromverbrauch. Wie in Kapitel 5.1 beschrieben, entspricht dies nicht der Realität. Trotzdem liefert dieses Modell teilweise richtige Simulationen. Dieses Modell kann nur Muster und Frequenzen darstellen. Die echten Verbrauchswerte werden damit nicht simuliert.
Für die beiden Hammingmodelle existiert eine weitaus detailliertere Beschreibung innerhalb des Buchs ”Power Analysis Attacks” [SM07] (S.39-42).
Versucht man Frequenzen und den realen Stromverbrauch in einen Normalverbrauch zu bringen bieten sich die folgenden zwei Methoden an.
Zum einen kann jede Operation (Addition, Multiplikation, Speichern, Laden, usw.) mit jedem möglichem Parameter ausgeführt werden, der Stromverbrauch

währenddessen gemessen und gespeichert werden. Da aus dem Kapitel 5.1 bekannt ist, dass Transitionen $0 \rightarrow 1$ und $1 \rightarrow 0$ den meisten Stromverbrauch miteinbringen, müssen diese Messungen für jede Operation Mehrmals ausgeführt werden. Für jede Möglichkeit des vorherigen Zustands der Ausgabe der Operationen müssen diese Schritte wiederholt werden.
Damit kann eine Abfolge von Operationen (ein Programm) direkt in die Prognose des Normalverbrauchs umgewandelt werden. Dabei sind die gespeicherten Messungen abhängig vom Prozessortyp und müssen für jeden neuen Typen neu erstellt werden. Doch die Komplexität einer solchen Messungskampagne steigt exponentiell mit der Komplexität des Prozessors (Anzahl der Operationen, Bitlänge der Eingaben, usw.) und wird somit ab einem bestimmten Punkt zeitlich unmöglich. Außerdem setzt dies Ganze voraus, dass man selbst bestimmen kann welche Operationen auf dem System ausgeführt werden. Auch ein tiefes Wissen über die Architektur der Recheneinheit wird benötigt.
Eine weitaus simplere Möglichkeit den Normalverbrauch zu erhalten, ist es ihn bei einem normalen Ablauf des Programms auszumessen. Dabei geht man wie in Kapitel 5.4.1 vor. Dafür werden keine Informationen über die Hardware oder Software des Systems benötigt. Nur ein grobes Verständnis der Handhabung und/oder des Betriebsverlaufs des Geräts, werden benötigt, um das System auszumessen.
Der Normalverlauf sollte mehrfach zu erreichen sein. Dadurch ist es möglich ihn mehrmals zu messen und so die Ergebnisse zu mitteln, um einen Großteil der Störgrößen auszufiltern.
Mittelt man den Stromverbrauch, ist darauf zu achten, dass Abweichungen in der Abtastrate oder beim Oszillator, des zu testenden Geräts, die Zeitbasis verzerren können und so das entstandene Signal degradieren.
Bestimmt man hingegen die Frequenzen und den Normalverbrauch und mittelt diese für jede Messung, können Störgrößen in den Frequenzen auftauchen, die nicht dazu gehörten. Trotzdem hat das Mitteln nach der Analyse den Vorteil resistenter gegen die Fehler, der variablen Abtastraten und Oszillatoren, zu sein.
Welche Methodik man auch verwendet hat (auch Kombinationen dieser sind möglich), sollte jetzt der Normalstromverbrauch in analysierter Form vorliegen. Also wird der Normalverbrauch auf die enthaltenden Frequenzen und den Durchschnittsverbrauch reduziert um sie im Folgenden zu vergleichen.

Vergleichen
Nachdem sowohl der Normalstromverbrauch als auch der zu analysierende Stromverbrauch ermittelt und auf Frequenzen und Durchschnittsstromverbrauch charakterisiert wurden, können sie verglichen werden.
Dafür werden die durchschnittlichen Verbrauchswerte P voneinander abgezogen. Dann sollte das Ergebnis als absolutes, einen vorher festgelegten Schwellenwert ε nicht überschreiten (Formel 4). Dieser Grenzwert muss anhand von den Störgrößen bestimmt werden. Wird der Grenzwert überschritten liegt eine Abweichung A_p vor, wobei dann $A_p = 1$ gilt, im Gegensatz zum Normalfall $A_p = 0$.

$$|P_{normal} - P_{analyse}| < \varepsilon \tag{4}$$

Auch für das Vergleichen von Frequenzen wird ein Schwellwert benötigt, welcher zwei Frequenzen zur selben Frequenz zusammen fast, wenn sie nah genug zusammenliegen. Können zwei Frequenzen vom Normal- und dem gemessenen Zustand zusammengefasst werden, liegt eine Übereinstimmung vor. Alle Frequenzen, die nicht mindestens mit einer weiteren Frequenz zusammengefasst werden können (sowohl beim Normalstromverbrauch als auch bei dem zu analysierenden Stromverbrauch), zählen als Abweichungen. Teilt man die Anzahl der Abweichungen A_f durch die Summe der Abweichungen A_f und der Übereinstimmungen U erhält man die Abweichrate R der Frequenzen (Formel 5).

$$R = \frac{A_f}{A_f + U} \tag{5}$$

Hat man die Abweichungen und die Abweichraten bestimmt, können sie zur Anomalierate verrechnet werden. Dafür müssen die zwei Aspekte gewichtet werden. $\alpha_{frequenz}$ gewichtet die Abweichrate bei den Frequenzen und $\alpha_{\varnothing verbrauch}$ bei den durchschnittlichen Verbrauchswerten. Es gilt $\alpha_{frequenz} + \alpha_{\varnothing verbrauch} = 1$. Die Anomalierate AR wird mit der Formel 6 berechnet.

$$AR = \alpha_{frequenz} * R + \alpha_{\varnothing verbrauch} * A_p \tag{6}$$

Wiederholt man das für jedes Zeitintervall der beiden zu vergleichenden Messungen, erhält man für jedes dieser Intervalle eine Anomalierate. So ist es bereits möglich Anomalien zu erkennen.

6 Testautomatisierung

6.1 Automatisierung im Allgemeinen

Automatisierung ist die Technologie, welche einen Vorgang oder Prozess mit minimaler menschlicher Hilfe ausführen lässt. Besonders gerngesehen ist das im Softwarebereich. Dabei werden Schritte die manuell von einer Person durchgeführt werden (z.B.: Umbenennen einiger Dateien) in Software soweit umgesetzt, dass (fast) alles von der Software abgearbeitet wird (z.B.: ein Skript zum Umbenennen vieler Dateien).
Das Reduzieren der menschlichen Komponente in Softwarevorgängen birgt Vor- und Nachteile. Die Vorteile in der Zeitersparnis und einer kleineren von Menschen verursachten Fehlerrate setzen sich in den meisten Fällen über alle Nachteile hinweg.
Die Nachteile sind aber trotzdem zu beachten und zu vermeiden. So können neuartige Fehler bis jetzt immer von Menschen erkannt und behandelt werden. Tritt nach der Automatisierung ein Fehlerfall auf, der nicht bedacht wurde, resultiert das im Absturz des Programms oder einer fehlerhaften Durchführung des Vorgangs. Weswegen auch Fehlerfälle von den Programmen erkannt und bewältigt werden müssen.
Daten, die von dem Programm bearbeitet oder erstellt wurden, können durch Fehler modifiziert worden sein, die nicht nachvollziehbar sind. Deswegen ist das Protokollieren der durchgeführten Aktionen eine hilfreiche Methode, um herauszufinden, ob die Daten vertrauenswürdig sind und hilft Fehler überhaupt zu finden, die während des Vorgangs aufgetreten sind. Dabei sollten so viele Daten wie möglich abgespeichert werden, solange die Daten klar gekennzeichnet sind. Im Nachhinein kann gefiltert werden, sollten es zu viele Informationen sein. Andersherum funktioniert das mit fehlenden Protokolldaten nicht.
Die erstellten Daten müssen auch abgespeichert werden (wenn auch nur temporär) bis sie genutzt werden. Dies kann in verschiedenen Formaten passieren. Dabei muss die Flexibilität und die resultierende Größe des Formats abgeschätzt werden, um das richtige Format zu wählen. CSV, JSON und XML sind relativ bekannte Domain Specific Language (DSL)’s, welche dies ermöglichen. Bibliotheken werden in den meisten Programmiersprachen für das Parsen dieser Sprachen angeboten. Eine eigen entwickelte DSL kann aber auch Speicher sparen, resultiert aber meist in mehr Aufwand.

6.2 Anomliererkennungsschritte automatisieren

Während der Vorbereitungsteil (Bestimmen was gemessen werden soll und wie die Parameter oder die Messlänge gesetzt werden sollen) der Anomalieerkennung schlecht bis gar nicht automatisiert werden kann, ist der Datenerhebungsprozess sowie das Verarbeiten der Daten gut automatisierbar. Dies gilt während der Normalverbrauch bestimmt wird als auch später bei der Bestimmung der zu überprüfenden Zustände.

6.2.1 Zustandsanalyse

Damit die Datenerhebung völlig automatisiert durchläuft, muss sowohl das Messen des Stromverbrauchs als auch das Interagieren mit dem zu testenden System durch die Software geschehen. [2] Bei der Datenerhebung muss zuerst die Messung gestartet und dann der Test durchlaufen werden. Dadurch garantiert man keine Daten zu verlieren, wenn die Messung zu spät startet. Weiterhin ist das Protokollieren auch bei der Anomalieerkennung ein entscheidender Faktor, der die gemessenen Daten in brauchbare und unbrauchbare Ergebnisse einteilen kann. Zu den nützlichen Metadaten, die in Logs gespeichert werden, zählen: Zeitintervall des Messvorgangs, Start- und Endzeitpunkte der Interaktion mit dem Testgerät. Auch die Zykluszahl findet später noch Nutzen. Weitere Daten, die mehr über das Gerät preisgeben, können später weiterhin entscheiden, ob der Zustand genauer bestimmt werden kann. Deswegen sind auch hier mehr Daten besser. Nach der Bestimmung der abzuspeichernden Metadaten laufen die Messungen immer im groben gleich ab:

1. Starte die Messung
2. Interagiere mit dem System
3. Beende das Interagieren
4. Beende die Messung
5. Speichere die Messungen und Metadaten ab
6. Springe zu 1. bis die Abbruchbedingung erfüllt ist

[2] Ein Test ist dabei ein vollständiger Interaktionsablauf.

Eine Abbruchbedingung kann dabei selbst bestimmt werden. Bedingungen wie das Erreichen eines bestimmten Zykluses oder das Verbrauchen einer bestimmten Speichergröße, sind gebräuchlich. Damit ein möglichst aussagekräftiger Normalverbrauch bestimmt werden kann sollte die Anzahl der Zyklen von der Größe der Störgrößen abhängen. Die Messungen, die auf Anomalien überprüft werden, sollen auch mit den gleichen Parametern öfters durchgeführt werden (wenn möglich) um eine höhere Fehlertoleranz zu erhalten.
Das Automatisieren der Verarbeitung der erhaltenen Daten birgt einige Schritte mehr, die von einem Menschen ausgeführt werden müssen. Da unsere Analyse mit der Fouriertransformationen einen Schwellwert, ein Intervall der zu erwartenden Frequenzen und die Minimaldistanz zwischen Frequenzen benötigt, müssen diese davor bestimmt werden. Weiterhin muss in den meisten Fällen die Größe der Zeitintervalle, in welche die Messungen zur Analyse unterteilt werden, von Hand ermittelt werden.
Die Signalanalyse kann nach der Bestimmung dieser Parameter dann auch automatisiert werden:

1. Lade die Messungen ein
2. Berechne den Stromverbrauch
3. Unterteile den Stromverbrauch in die Zeitintervalle
4. Analysiere jedes Intervall auf Frequenzen und den Durchschnittsverbrauch
5. Speichere die Ergebnisse ab

6.2.2 Anomalien

Möchte man also die Anomalieerkennung so weit wie möglich automatisieren, geht man wie folgt vor:

1. Parameter für die Messungen und Analyse bestimmen
2. Normalzustände durch das Messen oder Simulationen ermitteln
3. Tests ausführen und ausmessen
4. Normalzustände und gemessene Zustände vergleichen
5. Anomalien dokumentieren

Innerhalb des ersten Schritts werden die Parameter wie Dauer, Wertebereiche oder Offset des Wertebereichs für die Messung bestimmt. Des Weiteren sind die Analyseparameter wie Schwellwerte, das Intervall der zu erwartenden Frequenzen und die Minimaldistanz zwischen Frequenzen für die Fouriertransformation zu bestimmen. Auch außerhalb der Fouriertransformation sind einige Paramater vor der eigentlichen Automatisierung zu bestimmen. Darunter zählen die Zeitintervalle, in welche das Signal zur Analyse unterteilt wird, so wie auch der Schwellwert, unter welchem die nahe liegende Durchschnitts-Stromverbrauchswerte zu einem zusammengefasst werden. Dies benötigt derzeit noch Menschliches eingreifen, da das Bestimmen und Abschätzen der Parameter immer ein Abwägen von Vor- und Nachteilen ist.
Wird im zweiten Schritt der Normalzustand mithilfe einer Simulation bestimmt, muss trotzdem das Ergebnis der Simulation in die Frequenzen und dem mittleren Stromverbrauch umgewandelt werden. Wird der Normalzustand durch das Messen des Normalverbrauchs ermittelt, solle wie in Kapitel 5.4 und Kapitel 6.2.1 beschrieben, vorgegangen werden.
Auch für den dritten Schritt, bei dem der auf Anomalien zu untersuchende Test durchgeführt wird, sind die beiden Kapitel zu beachten.
Nachdem auch in den Schritten 4 und 5 das Kapitel 5.4 beachtet wurde, sollte ein ganzer Durchlauf vollendet sein. Nun kann immer wieder von Schritt 3 neugestartet werden, um den gleichen Test mit verschiedenen Parametern durchzuführen, um auch diese auf Anomalien zu überprüfen.

7 Praktisches Beispiel

Im praktischen Teil wird eine Electronic control unit (ECU) von Opel benutzt, um ihre Zustände zu bestimmen. Aus vorherigen Arbeiten wird ein Skript bereitgestellt, welches es ermöglicht Befehle über den Controller Area Network (CAN)-Bus auf das Gerät zu übertragen und so den Zustand des Geräts zu verändern. Dadurch konnte an einer echten Steuereinheit getestet werden.

7.1 Voraussetzungen/Pythonbibliothek

Für die digitale Analyse ist es unerlässlich Messwerte von einem Messgerät digital zu erhalten. Da innerhalb des Versuchs ein Oszilloskop verwendet wird, um den Stromverbrauch zu bestimmen, ist ein Teil der Arbeit eine Bibliothek für das Steuern und Abfragen von Messinstrumenten zu erstellen und später einzusetzen. Des Weiteren sollte es möglich sein durch die erstellte Bibliothek das verfügbare Netzteil anzusteuern.
Da sowohl das Netzteil (Rigol DP811A) als auch das verwendete Oszilloskop (Rigol DS1054Z) über die VISA Spezifikation verfügen und über die SCPI-Kommandos gesteuert werden können, hat man sich dafür entschieden die Geräte über diese Schnittstellen zu steuern. Dafür wird die PyVisa-Py und PyVisa Python Bibliothek verwendet um eine einfach zu benutzende, hardwareabstrahierende und umfassende Bibliothek zur Steuerung der Instrumente zu erstellen. Das Klassendiagramm in Abbildung 6 stellt den Aufbau der Bibliothek dar.
Die abstrakte PS-Klasse soll von den spezifischen geräteabhängigen Klassen geerbt werden und so Grundfunktionalitäten immer auf die gleiche Weise bereitstellen. Die PS-Klasse enthält die Memberfunktionen ”On” und ”Off” zum An- und Ausschalten von Ports an den Instrumenten. ”SetVoltage” und ”SetCurrent” sind Funktionen für Netzteile, um eine auszugebende Spannung und maximale Stromstärke zu setzen. Analog dazu geben die ”GetVoltage” und ”GetCurrent” Funktionen die derzeitige Spannung und Stromstärke zurück. Sicherheitsmaßnahmen wie Überspannungsschutz und Überlastschutz können durch ”SetOverVoltageProtection” und ”SetOverCurrentProtection” gesetzt werden. Durch ”StartPowerTrace” können Oszilloskope eine Messung starten. Sollte eine Funktionalität bei einem Gerät nicht verfügbar sein (z.B. das Starten einer Messung bei einem Netzteil), so wird eine Fehlermeldung zurückgegeben, welche übermittelt, dass diese Funktion nicht verfügbar ist.

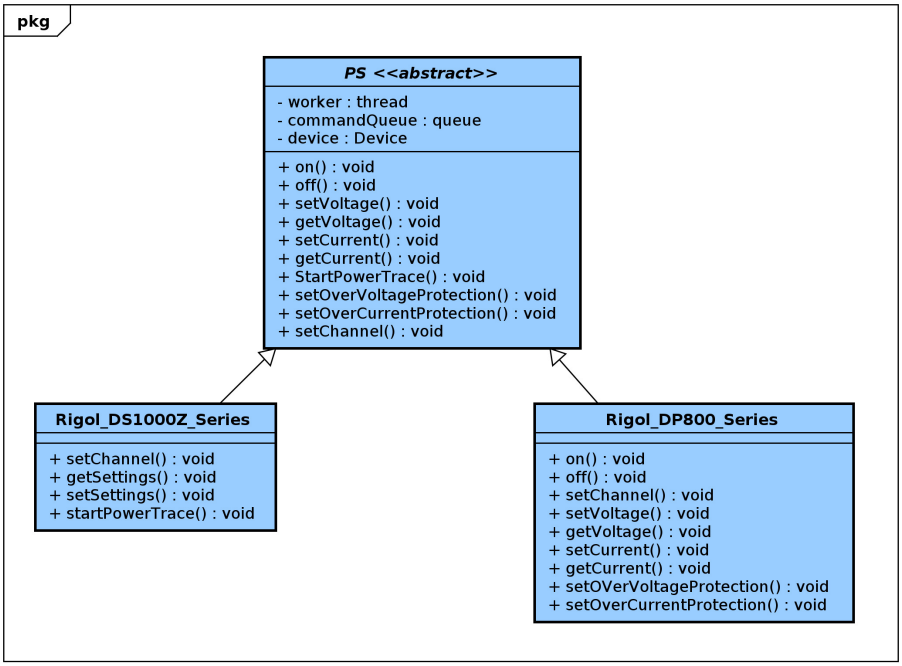


Abbildung 6: Vereinfachtes Klassendiagramm der Instrumentenbibliothek

Da zwischen dem Versenden der Kommandos und dem vollständigen Ausführen der Aktion des Gerätes einige Minuten vergehen können, kann dies dem Benutzer der Bibliothek Schwierigkeiten bereiten. Während eine Aktion ausgeführt wird, blockiert die PyVisa-Py Library in den meisten Fällen. So ist es z.B. nicht einfach eine Messung zu starten und im gleichen Programm Befehle an ein Steuergerät zu senden, um die Veränderung zu messen. Aus diesem Grund wurde die Bibliothek parallelisiert. Befehle welche ausgeführt werden sollen, werden in eine Warteschlange gebracht. Diese Warteschlange wird dann in einem weiteren Thread abgearbeitet. Die Rückgabewerte werden dann in einer Klasse abgelegt, welche im Vornherein bereits beim Aufrufen der jeweiligen Funktion der PS-Klasse zurückgegeben werden. Diese Rückgabeklasse hat eine Methode zum Erhalten des Wertes, wobei diese solange blockiert bis der Wert verfügbar ist.
Die Klassen Rigol_DS1000Z_Series und Rigol_DP800_Series sind Oberklassen der namentlich passenden Geräte der Marke Rigol. Dabei führen diese Klassen die Grundfunktionalitäten für die jeweilige Serie ein, ohne dabei Gerätespezifische Merkmale zu beachten. Das Einschränken der Funktionalität übernehmen die spezifischeren Klassen wie DS1054Z oder DP811A welche von den Oberklassen erben und deren Funktionalitäten einschränken (z.B. limitieren sie die Anzahl der möglich einzugebenden Porte auf die Anzahl der

physikalisch verfügbaren Porte). Dieser Aufbau der Oberklassen und spezifischeren Klassen wird ermöglicht durch eine große Ähnlichkeit zwischen den Geräten.
Beim Erstellen der Oberklassen wurden die ”Programming Guide”s der jeweiligen Serie verwendet. [3] Die Rigol_DS1000Z_Serie-Klasse implementiert die Funktionen On, Off, SetSettings, GetSettings, SetChannel und StartPowertTrace für die gleichnamige Serie von Oszilloskopen.
”On” und ”Off” schalten die jeweilig gegebenen Kanäle an (On) oder aus (Off). Dies ist in der Hinsicht nützlich, da bei mehr aktivierten Messkanälen die Speichertiefe pro Messkanal kleiner ist. Folglich begrenzt dies die maximale Dauer und/oder die maximal mögliche Abtastrate. ”SetChannel” setzt die Kanäle, die in den folgenden Funktionen verwendet werden sollen. Dieser Befehl ist nicht notwendig, wenn man die Kanäle bei jedem Funktionsaufruf mit übergibt.
”StartPowerTrace” startet eine Messung und gibt nach dem Empfangen der Messung die Werte zurück. Die ”GetSettings” und ”SetSettings” Funktionen beschaffen oder setzten die Einstellungen des Oszilloskops. Eine dieser beiden Funktionen sollten vor einem ”StartPowerTrace” aufgerufen werden. Geschieht das nicht wird innerhalb eines ”StartPowerTrace”-Aufrufs die ”GetSettings”-Funktion aufgerufen, was zu Verzögerungen zwischen dem Senden des Kommandos und dem Starten der Messung führt.
Mithilfe von ”GetSettings” werden die aktuellen Einstellungen des Oszilloskops geladen. Sollte der Parameter ”Read-only” nicht auf True gesetzt sein, so werden die vertikalen Einstellungen durch das Auslösen des ”Auto”-Modes eingerichtet.
”SetSettings” gibt dem Anwender die Möglichkeit die drei horizontalen Einstellungen zu setzen als auch für jeden Kanal seine eigenen vertikalen Einstellungen zu übernehmen. Zu den horizontalen Einstellungen stehen das Zeitfenster ”timewindow” in Sekunden, die Abtastgeschwindigkeit ”Samplerate” in Hertz und Speicherberreich ”Memorydepth” welches die Anzahl der Traces für jeden aktiven Kanal angibt (nur ein Wert kann für alle Kanäle gesetzt werden). Die vertikalen Einstellungen sind das ”Voltoffset” für den Offset des zu messenden Bereichs und die ”Voltscale” welche die vertikale Skalierung setzt.
Für diese drei Funktionen existiert jeweils eine Verbose Option. Sie lässt innerhalb der Konsole Debugginginformationen ausgeben, sollte als Hilfe während

[3] Rigol DS1000Z Serie: http://beyondmeasure.rigoltech.com/acton/attachment/1579/f-0386/1/-/-/-/-/DS1000Z_Programming%20Guide_EN.pdf (Zuletzt Besucht: 15.12.2019)
Rigol DP800 Serie: http://beyondmeasure.rigoltech.com/acton/attachment/1579/f-03a1/1/-/-/-/-/DP800%20Programming%20Guide.pdf (Zuletzt Besucht: 15.12.2019)

der Entwicklung mit der Bibliothek genutzt und später deaktiviert werden.
Die Rigol_DP800_Series Klasse implementiert die Funktionen ”On”, ”Off”, ”SetChannel”, ”SetVoltage”, ”GetVoltage”, ”SetCurrent”, ”GetCurrent”, ”SetOverVoltageProtection”, ”SetOverCurrentProtection” für die gleichnamige Serie von Netzteilen.
Die ”SetChannel”-Funktion ist dafür zuständig einen Standard Wert für die folgenden Funktionen zu setzen. Wie bei der Rigol_DS1000Z_Series-Klasse ist diese Funktion auch hier nicht unbedingt notwendig. Der Kanal kann auch während den anderen Funktionsaufrufen als Parameter mitgegeben werden.
Die Funktionen ”On” und ”Off” schalten die ausgewählten Ports an oder aus.
Die gewünschte Spannung kann durch ”SetVoltage” gesetzt werden, während mit ”GetVoltage” die aktuelle Spannung abgefragt werden kann.
Mit ”GetCurrent” wird die derzeitige Stromstärke vom Netzteil abgefragt. Die maximale nutzbare Stromstärke kann über ”SetCurrent” gesetzt werden.
Die DP800 Serie von Rigol enthält Schutzmechanismen, welche die maximal einzustellende Spannung und Stromstärke limitieren. Diese Schutzmechanismen können über ”SetOverVoltageProtection” für die Spannung und ”SetOverCurrentProtection” für die Stromstärke gesetzt werden.
Diese Bibliothek ermöglicht somit mit diesen Funktionalitäten eine einfache Steuerung. Dies sollte das folgende Beispiel zeigen:

```
# Connect to a DS1054Z Oscilloscope
av = AbstractVisa.Rigol_DS1054Z("123.245.123.245")
# Set the measurement time to 1.0 seconds,
# with 100 000 Hz samplerate for channel 1,2,3,4
av.SetSettings(timewindow=1.0,
            Samplerate=100000,
            ChannelNumber=[1,2,3,4])
# Start the measurement
x = av.StartPowerTrace()
# Simulating work
for i in range(0, 100):
      print("Simulating work - Part " + str(i + 1) + "/100")
      time.sleep(0.15)
# Get the data and save each channel to a csv file
y = x.GetValue()
for data in y:
      print("Saving Measurement of channel ", data.Channel)
      Path = r"\tmp\_data" + str(data.Channel) + ".csv"
      data.SaveToDataToCSV(Path)
```

Hierbei wird eine Verbindung zu einem Rigol DS1054Z Oszilloskop aufgebaut. Die Einstellungen Abtastrate, zu messendes Zeitintervall und verwendete Kanäle werden gesetzt und die Messung wird gestartet. Zwischen dem Starten der Messung und dem Abrufen der Ergebnisse wird eine Arbeit simuliert. Zuletzt werden die vier entstandenen Messungen jeweils in CSV-Dateien abgespeichert.

7.2 Aufbau

Zum Messen wird ein Oszilloskop der Marke Rigol (DS1054Z) benutzt. Da dieses nur mit Sonden für die Spannungsmessung geliefert wurde, kann der Stromverbrauch nur indirekt gemessen werden. Aus Kostengründen hat man sich entschieden die Stromstärke über einen Shuntwiderstand zu messen und mit der Spannung zu verrechnen, um den Stromverbrauch zu erhalten. Anfangs wurde dafür ein 0.33Ω Widerstand genutzt, der in Reihe zum Verbraucher geschaltet wurde. Während Unterschiede im Verbrauch festzustellen waren, wurde sich dafür entschieden einen größeren Widerstand zu wählen, um die Spannungstiefen besser erkennen zu können und eine genauere Messung durchzuführen. Da die verwendete ECU relativ robust ist [4], wurde der Widerstand durch einen 10Ω Widerstand ersetzt.

Die testdurchführende Einheit ist ein Raspberry Pi 4 mit CAN-Shield. Dieser startet die Aufzeichnung des Oszilloskops und sendet Testdaten an die oben angeführte ECU. Außerdem zeichnet der Raspberry Pi (RPI) auf, welche Antworten er bekommt, sodass bereits daran zu erkennen ist in welchem Zustand sich das Gerät befindet. Weiterhin wird ein Relais vom Raspberry Pi gesteuert, um den Stromkreislauf der ECU zu unterbrechen, wenn sie sich in einen Zustand befindet, der es nicht ermöglicht weitere Nachrichten zu empfangen oder auf sie zu reagieren (die ECU gilt dann als verglitched und muss neugestartet werden). Als Netzteil wurde anfangs ein Rigol DP811A vorhergesehen. Da dieses zum Anfang der ausführenden Phase nicht verfügbar war, wurde sich für die lokal vorhandenen Labornetzteile entschieden. Des Weiteren wurde eine Kamera an den RPI angeschlossen, um den Verlauf des Tests auch außerhalb des Labors beobachten zu können. Fehler konnten so auch außerhalb des Labors gelöst werden.

[4] Simple Tests zeigten, dass ein Bereich von 7-15 Volt für die ECU möglich war

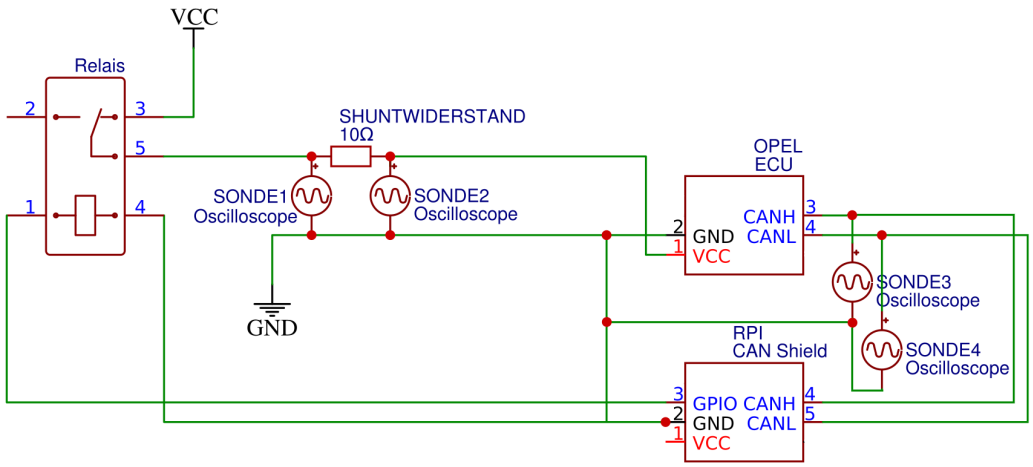


Abbildung 7: Schaltbild des Versuchsaufbau besteht aus einem Raspberry Pi mit CAN-Shield, einer ECU, einem Relais und einem Oszilloskop mit vier Sonden

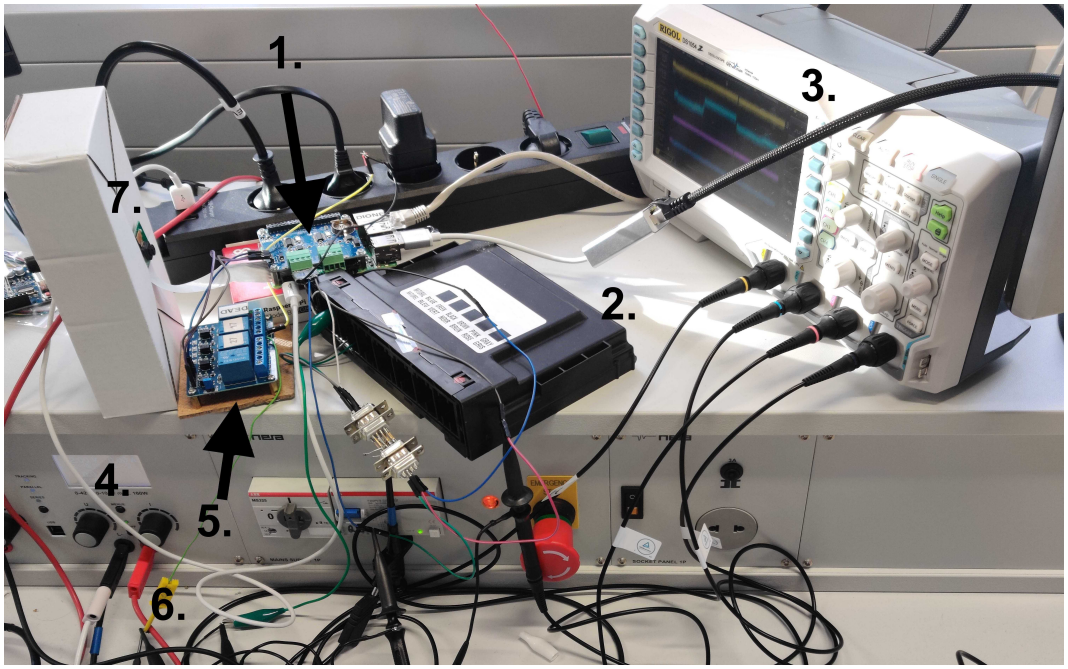


Abbildung 8: Versuchsaufbau in der Praxis: 1. RPI mit CAN-Shield, 2. Steuergerät von Opel (Identifikationsmerkmale wurden entfernt), 3. Oszilloskop, 4. Labornetzteil, 5. Relais, 6. Widerstand, 7. Observationskamera

7.3 Versuchsdurchlauf

Der erste Schritt in Richtung Anomalieerkennung, ist das Untersuchen der Steuereinheit. Mithilfe des RPI's und einem bereitgestellten Testskript zum Senden und Empfangen von Nachrichten, welche zu Codeexecution auf dem Gerät führten, kann mit der ECU agiert werden.
Während man jetzt schon eigenen Code auf dem Steuergerät hätte ausführen können und das mit anderen Arten von Code vergleichen könnte, fand sich ein viel interessanterer Fall zum Untersuchen. Codeexecution funktioniert in diesem Fall durch das Laden eines Codestücks in den RAM und dem Sprung zu diesem Code. Die Idee ist es denn Sprung auszuführen ohne Code bereitzustellen. Das sollte dazu führen, dass die ECU unsinnige Sachen ausführt und nach einiger Zeit, vom Watchdog zurückgesetzt werden sollte. Etwas viel interessanteres passiert aber des Öfteren. Das Steuergerät erhält die Nachricht und geht in einen ”verglitchten” Zustand der einen deutlich kleineren Stromverbrauch hat und es unmöglich macht dem Gerät neue Befehle zu senden. Bereits mit den bloßen Augen kann man sehen wie sich der Zustand des Systems anhand des Stromverbrauchs sichtbar verändert. Weiterhin kann anhand des Rückgabewertes, erkannt werden wie sich die ECU verhält. So ergeben sich Zustandswechsel, welche wie in Tabelle 3 charakterisiert werden können.
Da nun die Grundlagen klar sind, können nun Daten automatisiert erhoben werden. Dies geschieht über ein Pythonskript, welches die Bibliothek in Kapitel 7.1 nutzt. Dafür wird nach folgendem Schema vorgegangen:

1. Verbindung zum Oszilloskop herstellen und Einstellungen setzen
2. Einstellungen und Zeitstempel des Starts speichern
3. Starten der Messung
4. Kommando an ECU senden
5. Zeitstempel und Antwort speichern, während die Antwort verwendet wird, um den Zustandswechsel nach Tabelle 3 zuzuordnen
6. Springe 3 mal zu Nr.4
7. Beende die Messung und speichere das Gemessene ab
8. Sollte der Speicher des Speicherorts noch nicht voll sein springe zu Nr.2
9. Beende das Programm

Zustandswechsel	Charakterisierung
Glitched	Derzeitig ist die ECU in einem verglitchtem Zustand, in welchem sie nicht antwortet. Das Relais wird nun ausgelöst, um den Stromkreis zu unterbrechen und wieder zu schließen
sleepy	Die ECU war im Stromsparmodus und sollte nach dieser Nachricht wieder aktiv sein
sent_msg_answered	Eine Nachricht wurde gesendet und erfolgreich Empfangen und beantwortet. Es ist jetzt sehr wahrscheinlich, dass die ECU verglitched ist
sent_msg_unanswered	Eine Nachricht wurde gesendet und erfolgreich Empfangen, aber nicht beantwortet. Der vorherige Zustand ist wahrscheinlich nicht verändert worden.
sent_inf	Ein Fehler in der Kommunikation führte zum unendlichen Senden von Nachrichten. Der vorherige Zustand ist wahrscheinlich nicht verändert worden.

Tabelle 3: Aussagen, welche über die Zustände getroffen werden können, wenn diese Zustandswechsel anhand der Nachrichten erkannt werden

Somit werden so viele Messungen gestartet bis der Speicher des Geräts voll ist. Außerdem werden innerhalb von einer Messung immer drei Tests durchgelaufen. Dies liegt an der verwendeten ECU. Eine Messung vom Oszilloskop herunterzuladen, dauert bis zu 20 Minuten, da die Übertragungsgeschwindigkeit vom Oszilloskop limitiert ist. Das Steuergerät besitzt einen Stromsparmodus, der nach einigen Minuten aktiviert wird und durch eine Nachricht auf dem CAN-Bus wieder deaktiviert werden kann. Würde man also nur einen Test in einer Messung durchführen, würde man es schaffen die ECU aufzuwecken, dies aufzuzeichnen und beim Herunterladen der Daten vom Oszilloskop wieder das Steuergerät schlaffen legen. Deswegen werden drei Tests in einer Messung aufgezeichnet.
Gespeichert wird jede Messung mit ihren Daten in einem Ordner, welche fortlaufend nummeriert werden. Die Messdaten selbst werden in vier einzelnen Dateien im CSV-Format gespeichert. Die Metadaten werden in einer Datei namens ECU_Resp.info abgespeichert. Darin enthalten sind die Start- und Endzeiten der durchgeführten Tests, deren erkannte Antwort (wie in Tabelle 1) und zwischendurch auch der freie Arbeitsspeicher. Weiterhin werden die vollständigen Antworten der ECU aufgezeichnet und in einzelnen Dateien abgelegt.

Während das Automatisierungsskript dafür gedacht war bis zum Volllaufen des Speichers zu arbeiten, gab es einen Memoryleak innerhalb einer benutzten Bibliothek. Aus diesem Grund konnten meist nur 6-7 Messungen hintereinander durchlaufen werden, bevor das Programm wegen fehlendem Arbeitsspeichers abstürzte. Deswegen kommt es vor, dass zwischen den Messungen auch mehrere Tage vergehen.
Weitere Probleme gab es anfangs mit den erhaltenen Daten, da diese nicht den echten Werten entsprachen. Dies wurde später auf die Umrechnung der Rohbytes in Floats in der eigenerstellten Bibliothek zurückgeführt und gelöst werden. Aber das bedeutet, dass die Messungen erneut durchzuführen und die alten Daten zu verwerfen sind. Zum Schluss ist ein Satz von 40 erfolgreichen Messungen entstanden.

7.4 Messergebnisanalyse

Die Messergebnisanalyse wird hauptsächlich in Matlab durchgeführt. Dies liegt an der breiten Auswahl an verfügbaren Tools der Signalverarbeitung innerhalb von Matlab. Da Matlab in anderen Bereichen der Programmierung nicht sehr komfortable ist, werden Rohdaten als auch die Ergebnisse mit Pythonskripte überarbeitet.
So sind die Start- und Endzeiten ohne Zusatzinformationen aus der Metadatei genommen und in einer neuen Datei im CSV-Format abgespeichert worden, damit sie in Matlab einfacher zu nutzen sind. Des Weiteren werden die Zustände, welche anhand der empfangenen Antworten erkannt wurden, genauso in eine Datei geschrieben.
Aus den Messdaten werden zuerst nur die Messwerte verwendet, welche die Zustände ”sleepy”, ”sent_msg_answered” und ”glitched” beim Empfangen der Antworten festgestellt hatten. Die Messdaten werden eingeladen und verarbeitet. Da der CAN-Bus ein Differentiell nutzt, um Daten zu übertragen werden die zwei Messwertreihen (je Leitung) voneinander abgezogen. Zu sehen ist das Ergebniss in Abbildung 9.

Durch das abziehen der Spannung $U_{s,i}$ an jedem Zeitpunkt i nach dem Shuntwiderstand von der Netzspannung $U_{n,i}$ erhält man die Spannungsdifferenz $U_{d,i}$. Mithilfe der Spannungsdifferenz und der Größe des Shuntwiderstands R_s kann die Stromstärke I bestimmt werden ($I_i = U_{d,i} : R_s$).
Der Stromverbrauch P_i kann über die Zeit somit mit der Formel 7 berechnet werden. Abbildung 10 zeigt das Ergebnis dieser Berechnung.

$$P_i = U_{n,i} * I_i \tag{7}$$

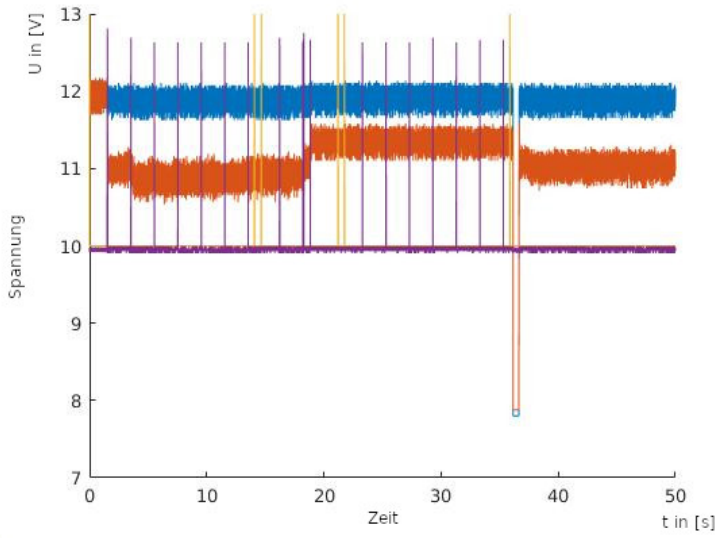


Abbildung 9: Rohdaten einer Messung, wobei Blau die Spannung vor dem Shuntwiderstand und Orange nach dem Widerstand darstellen. Lila stellt die Spannung des CAN-Busses dar (ein Offset zur schöneren Darstellung wurde aufgerechnet). Gelb steht für den Start und das Ende der Tests. Y = Spannung in Volt, X = Zeit in Sekunden

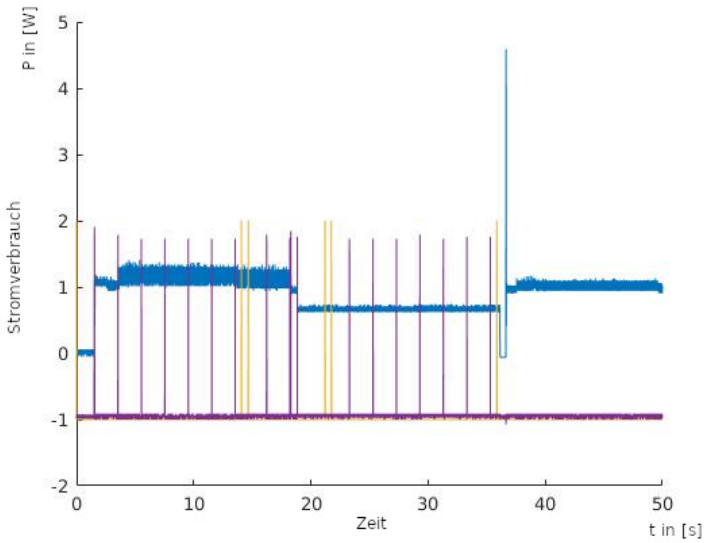


Abbildung 10: Stromverbrauch einer Messung(blau) der ECU, Lila und Gelb wie in (a) nur mit einem Offset von -1. Für den Stromverbrauch gilt: Y = Stromverbrauch in Watt, X = Zeit in Sekunden

Die Messungergebnisse werden in Zeitintervalle mithilfe der gesendeten CAN-Nachrichten und Zeitpunkten aus den Metadaten gewählt. Dabei wird der Start einer CAN-Nachricht verwendet, um den Startzeitpunkt eines Intervalls festzulegen. Da CAN-Nachrichten die Hauptursache von Zustandswechsel in diesen Versuchen sind, ist diese Methodik der Zeitintervallsetzung besonders attraktiv.
Außerdem wird für den ersten Analyseversuch aus den 40 Messreihen 15 ausgewählt. Alle 15 Messungen meldeten die Zustände sleepy, sent_msg_answered und Glitched in genau dieser Reihenfolge, zurück. Das vereinfachte das Auswerten der Daten.
Jeder einzelne Teil wird mithilfe der Fouriertransformation untersucht, wobei die minimale Frequenz für Zustand 1 und 3 auf 150 und für Zustand 2 auf 250 gesetzt wird. Für die Minimalhöhe der Spitzen in der Fouriertransformation wird 80 für den Zustand 1 und 175 für Zustände 2 und 3 gewählt. Dies wurde für alle 15 Messungen wiederholt.
Nach der Analyse sind die Ergebnisse wie folgt dargestellt:
Alle Frequenzen der Teile jedes einzelnen vorher erkannten Zustands wird für jede Messung vereint. Somit erhält man eine Darstellung von 15 Gruppen an erkannten Frequenzwerten für jeden Zustand.

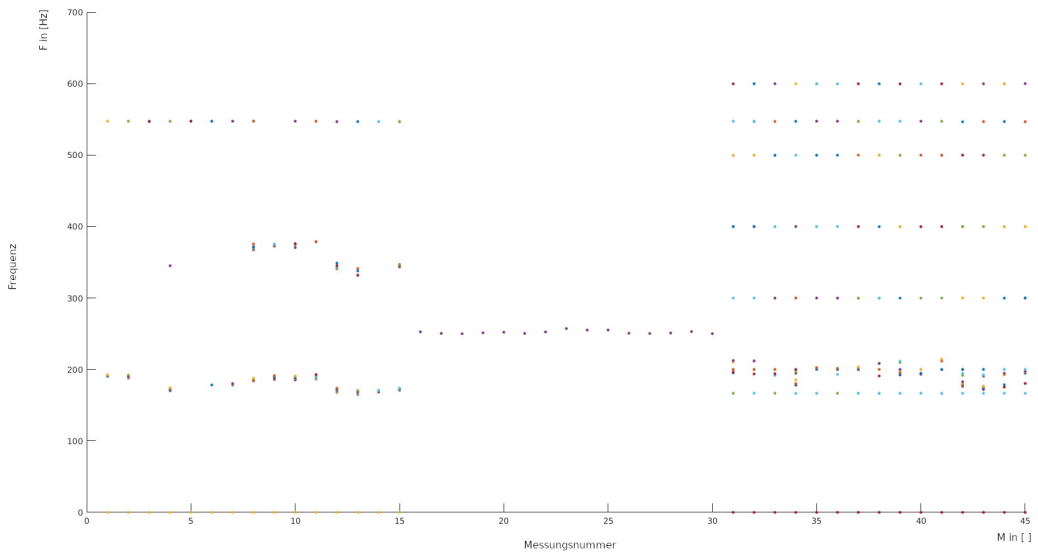


Abbildung 11: Erstes Ergebnis der Zustandsanalyse, F = Frequenzen in Hz, Messungsnummer = $M - (Zustandsnummer - 1) * 15$

Beispielhaft bedeutet ein Wert bei X = 5 und Y = 550, dass bei der Messung 5 im Zustand 1 eine Frequenz von 550 Hz erkannt wurde. Oder als weiteres Beispiel könnte der Wert bei X = 17 und Y = 250 als Messung 2 in Zustand 2 mit einer erkannten Frequenz von 250 Hz interpretiert werden.
Bereits eine grobe Betrachtung ergibt eine klare Spaltung der Daten in drei Bereiche. Jeder Bereich stellt einen Zustand dar. Somit können die erkannten Zustände bereits relativ gut unterschieden werden.
Hier wird versucht einen Normalzustand zu bestimmen indem 15 Messungen von jeweils drei Tests durchgeführt, analysiert und einzeln dargestellt wurden. Innerhalb des ersten Zustands ist eine Tendenz zur 547 Hz und 185 Hz zu erkennen. Die Frequenzen zwischen 350 und 400 Hz scheinen nicht immer vorhanden zu sein und sind relativ weit verstreut.
Zustand 2 scheint nur eine erkannte Frequenz bei 252 Hz zu enthalten.
Die Frequenzen 600, 547, 500, 400, 300 und 166 Hz setzen sich stabil in allen Messungen für Zustand 3 fort. Nur im Bereich 175 bis 210 Hz springen die Frequenzen.
Wenn also beispielhaft zum jetzigen Zeitpunkt Zustand 3 als Normalzustand erwartet wird, aber die Frequenzen aus Zustand 1 eintreten, kann auf eine Anomalie geschlossen werden. Was zu zeigen war.

8 Fazit, Nutzen und Ausblick auf weitere Arbeiten

Innerhalb dieser Arbeit konnte erörtert werden, dass der Stromverbrauch durch die Analyse auf Operationen, Zustände und somit Anomalien rückführbar ist. Es ist möglich Anomalien an dem Stromverbrauch zu erkennen, aber eine leere Menge an erkannten Anomalien sagt nicht aus, dass keine Anomalien aufgetreten sind.
Für diese Aussage wurde zuerst die Theorie hinter der Quelle des Stromverbrauchs ermittelt, wobei man zu dem Schluss kam, dass Operationen zu einem Stromverbrauch führen. Eine Rückführung vom Stromverbrauch zu Operationen ist nicht eindeutig möglich. Trotzdem können die Stromverbrauchswerte verglichen werden und bei Unterschieden auf abweichende Operationen geschlossen werden.
Weiterhin wurde die Messung des Stromverbrauchs und die Analyse durch die Fouriertransformation und arithmetische Mittel beschrieben, um die Messungen in einen vergleichbaren Ausgangspunkt zu bringen.
Ein Normalverbrauch konnte mithilfe von Simulationen oder messtechnisch ermittelt werden. Durch die Analyse und dem Vergleichen des Normalverbrauchs und den Testmessungen, konnten Anomalien festgestellt werden.
Die Schritte der Anomalieerkennung wurden für die Automatisierung untersucht und dargestellt.
Innerhalb des praktischen Beispiels wurde die Theorie angewendet. Dafür wurde ein Versuchsaufbau aufgebaut, worin ein Oszilloskop, ein Opel Steuergerät als zu testendes System und ein RPI, als mess- und teststartendes System eingesetzt wurden. In den folgenden Messungen wurde bereits die selbsterstellte Python Bibliothek ”PS” verwendet. Die Messungen wurden analysiert und das Ergebnis bestätigte die Theorie.

8.1 Nutzen

Wie der Titel dieser Arbeit schon andeutet, kann die Testautomatisierung von der Thematik der automatisierten Anomalieerkennung Gebrauch machen. Dadurch wäre es möglich eine Schnittstelle auf alle möglichen Eingaben zu testen und zu erkennen ob und auf welche Weise das System auf diese Eingabe reagiert. Somit kann das Fuzzing durch eine weitere Rückgabewert erweitert werden.
Eine weitere Anwendung könnte die Anomalieerkennung in der Schadsoftwareerkennung oder Hardwarefehlererkennung finden. Unter der Voraussetzung, dass ein System eine vordefinierte Software nutzt, kann jede Abweichung von dieser Software auf Fehler der Hardware oder Schadsoftware zurückgeführt werden. Dies wäre in den Bereichen der Automobilbranche und der Internet of Things (IoT)-Branche besonders interessant, da hier die Software meist die Voraussetzungen erfüllen und die immer größere Vernetzung der Geräte in diesen Branchen einen größeren Anspruch auf Sicherheit genügen müssen. Durch eine Minimierung der Kosten und Größe eines Anomalieerkennungssystems ist es vorstellbar, dass ein solches System in dieser Branche eingesetzt werden kann.

8.2 Ausblick auf weitere Arbeiten

Da innerhalb des Machine Learnings die Thematik der Pattern Recognition (zu Deutsch Mustererkennung) ein relativ großer Themenbereich ist, ist es vorstellen die Zustandserkennung damit zu erweitern, um noch präziser die Zustände und somit Anomalien zu erkennen.
Des Weiteren könnten noch Forschungen in Richtung des Wertebereichs von Störgrößen und messbaren Stromverbräuchen durchgeführt werden, um die Parameterbestimmung zu vereinfachen. Auch hier könnten Aspekte des Machine Learnings eingesetzt werden, um die Parameterbestimmung weiter zu automatisieren.

Literatur

[AS04] Eran Tromer Adi Shamir. Acoustic cryptanalysis on nosy people and noisy machines. 2004. http://cs.tau.ac.il/~tromer/acoustic/ec04rump/, Letzter Zugriff: 16.12.2019.

[Foua] IVI Foundation. *Standard Commands for Programmable Instruments (SCPI)*. VERSION 1999.0, 05.1999, http://www.ivifoundation.org/docs/scpi-99.pdf, Zuletzt besucht: 23.12.2019.

[Foub] IVI Foundation. *VPP-4.3: The VISA Library*. Revision 5.8, 17.10.2017, http://www.ivifoundation.org/downloads/Class%20Specifications/vpp43_2017-10-17.pdf, Zuletzt besucht: 25.10.2019.

[KJJR11] Paul Kocher, Joshua Jaffe, Benjamin Jun, and Pankaj Rohatgi. Introduction to differential power analysis. *Journal of Cryptographic Engineering*, pages 5–27, Apr 2011. https://doi.org/10.1007/s13389-011-0006-y, Letzter Zugriff: 06.12.2019.

[LCC08] Thanh-Ha Le, Cécile Canovas, and Jessy Clédière. An overview of side channel analysis attacks. In *Proceedings of the 2008 ACM Symposium on Information, Computer and Communications Security*, ASIACCS '08, page 33–43, New York, NY, USA, 2008. Association for Computing Machinery. https://doi.org/10.1145/1368310.1368319, Letzter Zugriff: 11.01.2020.

[LFK19] Chao Luo, Yunsi Fei, and David Kaeli. Side-channel timing attack of rsa on a gpu. *ACM Trans. Archit. Code Optim.*, 16(3):32:1–32:18, August 2019. http://doi.acm.org/10.1145/3341729, Letzter Zugriff: 16.12.2019.

[Par14] Rainer Parthier. *Messtechnik: Grundlagen und Anwendungen der elektrischen Messtechnik*. Springer Fachmedien Wiesbaden, 2014.

[SM07] Thomas Popp Stefan Mangard, Elisabeth Oswald. *Power analysis attacks : revealing the secrets of smart cards*. Springer, 2007. https://www.regensburger-katalog.de/s/ubrfhr/de/2/1035/BV036480550, Letzter Zugriff: 21.12.2019.

[Sti09] Leonhard Stiny. *Handbuch aktiver elektronischer Bauelemente*. Franzis Verlag, 2009.

[WMM+17] Satohiro Wakabayashi, Seita Maruyama, Tatsuya Mori, Shigeki Goto, Masahiro Kinugawa, and Yu-ichi Hayashi. Poster: Is active electromagnetic side-channel attack practical? In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 2587–2589, New York, NY, USA, 2017. ACM. https://doi.org/10.1145/3133956.3138830, Letzter Zugriff: 28.12.2019.