HAW HAMBURG

Bachelorarbeit

Erik Raik Engelhardt

Realisation einer FPGA-basierten psychometrischen Authentifizierung mittels Touchscreen

Fakultät Technik und Informatik
Department Informations- und Elektrotechnik

Faculty of Computer Science and Engineering
Department Information and Electrical Engineering

Erik Raik Engelhardt

Realisation einer FPGA-basierten psychometrischen Authentifizierung mittels Touchscreen

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang Bachelor of Science Elektro- und Informationstechnik
am Department Informations- und Elektrotechnik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr.-Ing. Robert Fitz
Zweitgutachter: Prof. Dr. Pawel Buczek

Eingereicht am: 26.04.2019

Erik Raik Engelhardt

Thema der Arbeit

Realisation einer FPGA-basierten psychometrischen Authentifizierung mittels Touchscreen

Stichworte

FPGA, SoC, Biometrie, psychometrische Authentifizierung, Touchscreen, maschinelles Lernen, Klassifikation, Zynq-7000

Kurzzusammenfassung

Bei der Eingabe einer klassischen Pin auf einem Touchscreen wird nur ein Bruchteil der zur Verfügung stehenden Daten verwendet. Diese Arbeit beschreibt die Implementierung einer Methode, welche die genaue Position jeder Berührung und das Zeitverhalten der Eingabe berücksichtigt. Dabei wird der Nutzer in einem Feldtest in 95% der Fälle erfolgreich erkannt. Angreifer hingegen konnten sich auch mit Kenntnis über das Verfahren, die Pin und das erwartete Eingabeverhalten nur in 7% der Fälle authentifizieren.

Erik Raik Engelhardt

Title of Thesis

Realization of an FPGA based psychometric authentication using touchscreens

Keywords

FPGA, SoC, biometrics, psychometric authentication, touchscreen, machine learning, classification, Zynq-7000

Abstract

When entering a pin code on a touchscreen only a fraction of the available data is utilized. In this thesis the implementation of a methods which takes the exact position of every touch and the time characteristic into account is presented. A field test shows that users are correctly recognized in 95% of the cases. Even with knowledge about the method, pin code and the expected input behavior attackers are only able to authenticate themselves in 7% of the cases.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungen

ADC Analog to Digital Converter. 53, 55, 60, 61

AMBA Advanced Microcontroller Bus Architecture. 21

AXI Advanced Extensible Interface. 12, 20–22, 43, 63

BCD Binary-Coded Decimal. 79, 80

CAN Controller Area Network. 12

CLB Configurable Logic Blocks. 12

CPHA Clock Phase. x, 21

CPOL Clock Polarity. x, 21

CS Chip Select. 21

DSP Digital Signal Processor. 12

EER Equal Error Rate. x, 5–8

FAR False Acceptance Rate. x, 5, 6

FMC FPGA Mezzanine Card. 39

FN Flase Negative. 28

FP False Positive. 28

FPC Flexible Printed Circuit. 43, 59

FPGA Field Programmable Gate Array. 12

FRR False Rejection Rate. x, 5, 6

GPIOs General-Purpose Inputs/Outputs. 9, 12, 39

HDl Hardware Description Language. 12

HDMI High Definition Multimedia Interface. 9, 39, 45, 63

I2C Inter-Integrated Circuit. 12

IP-Core Intellectual Property Core. vii, 22, 24, 64, 72

ISR Interrupt Service Routine. 81, 83–85, 87

k-NN k-Nearest Neighbors. x, 7, 32, 33

LSB Least Significant Bit. 60

LVDS Low Voltage Differential Signaling. 39

MISO Master in - Slave out. 21, 55, 60

ML Machine Learning. 26, 32

MLP Multilayer Perceptron. 7

MOSI Master out - Slave in. 21, 55, 60

MSB Most Significant Bit. 60

OLED Organic Light-Emitting Diode. 9, 63, 64

PCIe Peripheral Component Interconnect Express. 12

PL Programmable Logic. 12, 20, 43, 45, 47, 53, 56, 62–64, 72, 101

PS Processing System. 12, 43, 45, 47–50, 53, 63, 64, 68–70, 72, 73, 76, 77, 80, 81, 101

RBF radiale Basisfunktion. 35

RF Random Forest. 7

SAR Successive Approximation Register. 53, 55

SCLK SPI Clock. 21

SD Secure Digital. 9, 42, 43, 49, 83

SoC System on Chip. 9

SPI Serial Peripheral Interface. vii, 12, 20–22, 45, 60

SVM Support Vector Machine. viii, x, 6, 35, 36

TN True Negative. 28

TP True Positive. 28

TTSOP Thin Shrink Small Outline Package. 53

UART Universal Asynchronous Receiver Transmitter. 12

VGA Video Graphics Array. 39

VHDL Very High Speed Integrated Circuit Hardware Description Language. 69

1 Einleitung

Die Anforderungen an Authentifizierungsverfahren sind ebenso vielfältig wie ihre Anwendungsgebiete [24] [4]. Ein perfektes Authentifizierungsverfahren wäre absolut sicher, mit keinem Aufwand für den Nutzer verbunden und stünde jedem Nutzer gleichermaßen zur Verfügung. Da dies allerdings nicht realistisch ist, wurden im Laufe der Jahre etliche Authentifizierungsverfahren entwickelt. Alle mit ihren eigenen Vor- und Nachteilen.

Generell gibt es drei Wege einen Nutzer zu authentifizieren. Durch die Kenntnis einer geheimen Information, den Besitz einer Sache oder durch die Einzigartigkeit des Benutzers selbst [54]. Nehmen wir an Alice möchte Bob eine vertrauliche Information zukommen lassen. Bevor sie dies tut möchte Sie aber sicherstellen, dass es sich bei ihrem Gesprächspartner auch wirklich um Bob handelt. Kennt Alice Bob bereits und findet das Treffen in Person statt, kann sich Alice auf die körperlichen Merkmale Bobs verlassen und Ihn anhand dieser authentifizieren. Treffen sich die beiden zum ersten Mal, haben aber einen gemeinsamen Bekannten, der Bob eine spezielle Karte als Beweis seiner Authentizität gegeben hat, könnte Alice Bob über den Besitz dieser Karte authentifizieren. Findet das Gespräch am Telefon statt, könnte Bobs Kenntnis über ein zuvor vereinbartes Passwort als Beweis seiner Authentizität genutzt werden.

Bereits an diesem Beispiel lässt sich abschätzen, dass es vermutlich kein perfektes Verfahren gibt, um Bob zu authentifizieren. Das Aussehen von Bob ist eine öffentliche Information. Mit entsprechendem Aufwand wäre es einem Angreifer möglich die körperlichen Merkmale Bobs so zu kopieren, dass Alice den Angreifer für Bob hält. Auch denkbar wäre, dass Bob einen Zwillingsbruder hat, welcher sich als Bob ausgibt. Zudem wäre es möglich, dass sich Alice und Bob schon eine Zeit lang nicht mehr gesehen haben und Bob sich in der Zwischenzeit so verändert hat, dass er nicht mehr von Alice erkannt wird. Daran lässt sich eins der bestimmenden Merkmale einer biometrischen Authentifizierung erkennen: Die Authentifizierung erfolgt in der Regel nicht sicher sondern lediglich mit einer bestimmten Wahrscheinlichkeit. Doch auch die Authentifizierung durch Besitz und

durch Wissen ist nicht perfekt. Der Besitz kann entwendet werden oder verloren gehen und auch das Wissen kann vergessen und abgehört oder erraten werden.

Der Vorteil gegenüber der biometrischen Authentifizierung ist jedoch, dass man sowohl den Besitz als auch das Wissen ersetzen kann. Heutzutage wird beispielsweise häufig der Fingerabdruck zum Entsperren des Smartphones verwendet. Die Einzigartigkeit des menschlichen Fingerabdrucks scheint wie dafür gemacht zu sein, als Merkmal für eine Authentifizierung zu dienen. Es ist zwar möglich einen Fingerabdruck zu kopieren und damit das Authentifizierungsverfahren zu überlisten [18] [29] [44], dies ist aber wesentlich aufwendiger, als die Eingabe einer Pin zu beobachten und somit das geheime Wissen zu erlangen [25]. Der entscheidende Nachteil einiger biometrischer Authentifizierungsverfahren ist nun allerdings, dass es keine Möglichkeit gibt ein kompromittiertes Merkmal, in gleicher Weise wie eine kompromittierte Pin, zu ersetzen. Eine Pin ist schnell geändert. Einen neuen Fingerabdruck oder eine neue Netzhaut sind schwieriger zu beschaffen.

Ein weiteres Problem, welches unweigerlich auftaucht, ist die Sicherheit der Betriebssysteme. Ein Authentifizierungsverfahren kann noch so sicher sein, wenn das zugrunde liegende Betriebssystem weitgehende Schwachstellen hat, bietet dies dem Angreifer gegebenenfalls einen Weg um das Authentifizierungsverfahren einfach zu umgehen. Besonders im Bereich der mobilen Endgeräte, wie Smartphones, Tablets und ähnlichem, gibt es immer wieder erhebliche Sicherheitslücken, welche entsprechende Authentifizierungsverfahren gefährden können [26] [42].

1.1 Ziel der Arbeit

Viele biometrische Merkmale sind schwer zu ersetzen und Authentifizierungsverfahren, welche auf der Eingabe einer Pin beruhen, relativ leicht durch das Ausspähen eben jener umgehbar. Daher wird in dieser Arbeit eine Kombination aus beiden Verfahren untersucht. Ziel der Bachelorarbeit ist es zu untersuchen, ob die Einbeziehung des Eingabeverhaltens bei der Eingabe einer herkömmlichen Pin auf einem Touchscreen zur Verbesserung der Sicherheit betragen kann. Im Idealfall:

- Könnte sich der rechtmäßige Benutzer bei *jedem Versuch* durch Eingabe seiner vermeintlich geheimen Pin und seinem, im Idealfall einzigartigem, Eingabeverhalten authentifizieren.

- Könnte sich ein Angreifer, obwohl er Kenntnis über die vermeintlich geheime Pin und über das Eingabeverhalten des Nutzers hat, in *keinem Versuch* authentifizieren.

In einem realistischerem Szenario könnte sich der Benutzer in ausreichend vielen Fällen authentifizieren und ein Angreifer würde mehr Versuche brauchen, als vom System erlaubt sind, woraufhin das System den Account des Benutzers sperren würde und sich dieser eine neue Pin errichten könnte, um die kompromittierte zu ersetzen.

Das Verfahren soll die Vorteile des erschwerten Kopieren eines biometrischen, in diesem Fall spezieller eines psychometrischen Merkmals, mit dem leichtem Austausch einer Pin kombinieren. Hierfür sollen die Eingaben mehrerer Nutzer aufgezeichnet werden, um anschließend ein geeignetes Verfahren zu finden und zu implementieren. Aufgrund der erwähnten Sicherheitsbedenken mobiler Betriebssysteme soll dieses Verfahren möglichst hardwarenah implementiert werden.

1.2 Aufbau der Arbeit

Zunächst soll in Kapitel 2 ein Überblick über ähnliche und verwandte Arbeiten gegeben werden. Anschließend wird in Kapitel 3 ein Überblick über die verwendeten Technologien und Verfahren gegeben. Daraufhin werden in Kapitel 4 die Anforderungen an das System und das Authentifizierungsverfahren konkretisiert. Nach dem Entwickeln eines Konzeptes in Kapitel 5 wird in Kapitel 6 die Implementierung der notwendigen Funktionen sowie die Erfassung der Nutzerdaten, das Finden eines geeigneten Authentifizierungsverfahren und der Test dieses Verfahrens beschrieben. Zum Ende der Arbeit werden in Kapitel 7 die Ergebnisse zusammengetragen und potentielle zukünftige Entwicklungen aufgezeigt.

2 Einordnung der Arbeit

Das Verwenden des Bedien- beziehungsweise Tippverhaltens von Nutzern zur Authentifizierung ist ein bereits viel erforschtes Gebiet. Aus diesem Grund soll nun im Folgenden ein Überblick über bereits geleistete Arbeiten gegeben werden. Anschließend wird dargestellt wie sich diese Arbeit in das Forschungsgebiet einordnen lässt.

2.1 Vorangegangene Arbeiten

Aufgrund der Vielzahl an Arbeiten zu diesem Thema kann lediglich ein Einblick in den aktuellen Forschungsstand gegeben werden. Die Auswahl der vorgestellten Arbeiten soll einen geeigneten Überblick verschaffen.

An Introduction to Biometric Recognition [4]
In ihrer Arbeit aus dem Jahr 2004 geben Anil K. Jain *et al.* einen Überblick über verschiedene Gebiete der biometrischen Authentifizierung und fassen ihre Vor- und Nachteile sowie Stärken und Schwächen zusammen. Dabei betrachten sie sieben Kriterien zur Bewertung der verschiedenen biometrischen Merkmale. Ihre Befunde zum Tippverhalten lauten wie folgt:

- Es besteht die Hypothese, dass jeder Mensch ein charakteristisches Eingabeverhalten an einer Tastatur besitzt.
- Es wird vermutet, dass das Tippverhalten nicht einzigartig ist, jedoch genug charakteristische Informationen enthält, um eine Identifikation von Personen zu ermöglichen.
- Es wird erwartet, da es sich um ein psychometrisches Merkmal handelt, dass es bei einigen Personen starke Schwankungen im Tippverhalten gibt.

- Das Tippverhalten einer Person kann unbemerkt während der Eingabe durch einen Angreifer beobachtet werden.

Biometric Authentication: A Review [20]
Debnath Bhattacharyya *et al.* vergleichen in ihrer Arbeit aus dem Jahr 2009 verschiedene biometrische Authentifizierungsverfahren. Speziell beim Thema Tippverhalten kommen sie zum Schluss, dass es sowohl zur Identifikation von erfahrenen Schreibkräften, als auch von Personen, die unerfahren im Schreiben an einer Tastatur sind, geeignet sei. Des weiteren vermuten sie, dass das Implementieren eines solchen Verfahrens günstig sein sollte, da lediglich die Installation eines Software-Paketes notwendig sei. In einem Vergleich mit anderen biometrischen Merkmalen anhand der Kriterien Equal Error Rate (EER)[1], False Acceptance Rate (FAR) und False Rejection Rate (FRR) (s. Tabelle 2.1) ist erkennbar, dass das Tippverhalten durchaus als Merkmal für eine Identifizierung geeignet ist. Betrachtet man nur die EER ist das Tippverhalten nur unwesentlich schlechter, als die Erkennung mittels Fingerabdruck. Allerdings ist die Anzahl der Testpersonen bei der Studie zum Fingerabdruck wesentlich größer als bei der Studie zum Tippverhalten. Um einen wirklich Aussagekräftigen Vergleich durchführen zu können, bräuchte man Studien mit einer ähnlichen Anzahl an Testpersonen, da man vermuten könnte, dass mit steigender Anzahl an Testpersonen auch die Zahl der Personen steigt, welche ein ähnliches Tippverhalten aufweisen.

Keystroke dynamics as a biometric for authentication [38]
Fabian Monrose und Aviel D. Rubin beschäftigten sich mit der Analyse des Tippverhaltens von Benutzern bei der Passworteingabe an einer klassischen Tastatur mit dem Ziel die Benutzer aufgrund spezifischer Charakteristika unterscheiden zu können. Sie schlussfolgern, dass obwohl das Tippverhalten als psychometrisches Merkmal inhärente Einschränkungen mit sich bringt, wie beispielsweise die Abhängigkeit von der Umgebung beziehungsweise Umwelt, es dennoch in Verbindung mit einer traditionellen Passworteingabe die Sicherheit eines Systems verbessern kann. Für ihre Datensätze von 63 Benutzern konnten sie eine Genauigkeit von bis zu 92.14% erreichen.

[1] Die Equal Error Rate (EER) eine Kennzahl für die Güte eines biometrisches Authentifizierungsverfahrens. Er ist dann gegeben, wenn die Wahrscheinlichkeit, dass ein Benutzer fälschlicherweise abgelehnt wird (False Rejection Rate) und die Wahrscheinlichkeit, dass ein Angreifer fälschlicherweise nicht abgelehnt wird (False Acceptance Rate) gleich sind [56].

Biometrik	EER	FAR	FRR	Testpersonen	Kommentare
Gesicht	NA	1%	10%	37437	Unterschiedliche Lichtverhältnisse, Indoor/Outdoor
Fingerabdruck	2%	2%	2%	25000	Rotation und übermäßige Hautverzerrungen
Handgeometrie	1%	2%	2%	129	Mit Ringen und ungenauer Platzierung
Iris	0.01%	0.94%	0.99%	1224	Indoor Umgebung
Tippverhalten	1.8%	7%	0.1%	15	Datenerfassung über sechs Monate
Stimme	6%	2%	10%	30	Textabhängig und Mehrsprachig

Tabelle 2.1: Vergleich verschiedenen biometrischer Merkmale anhand von EER, FAR und FRR [20]

Keystroke dynamics enabled authentication and identification using triboelectric nanogenerator array [61]
Changsheng Wu *et al.* untersuchten das Eingabeverhalten an einem eigens entwickelten triboelektrischen Eingabegerät (s. Abb. 2.1). Mit diesem waren sie auch in der Lage den Druck der Eingabe als zusätzliches Merkmal für die Nutzererkennung zu verwenden. Mit einer entsprechend trainierten SVM waren Sie in der Lage eine EER von 1.15% zu erreichen.

Touchalytics: On the Applicability of Touchscreen Input as a Behavioral Biometric for Continuous Authentication [28]
Mit dem Thema der andauernden Authentifizierung bei der Benutzung eines Smartphones beschäftigten sich Mario Frank *et al.* Sie suchten nach einem Verfahren, welches das Verhalten des Benutzers nach der Entsperrung des Smartphones durchgängig beobachtet und bewertet. Hierfür extrahierten sie 30 Merkmale aus den Touchscreen-Logdaten und trainierten einen Klassifikator. Dieser hatte im Median eine EER von 0% während einer einzelnen Sitzung, 2% bis 3% für aufeinander folgende Sitzungen und 4% für Sitzungen, welche mehr als eine Woche auseinander lagen. Sie kamen zu dem Schluss, dass sich diese Methode nicht als alleiniges Authentifizierungsverfahren eignet, jedoch sehr wohl um die Zeit bis zum Sperren des Bildschirms zu verlängern.

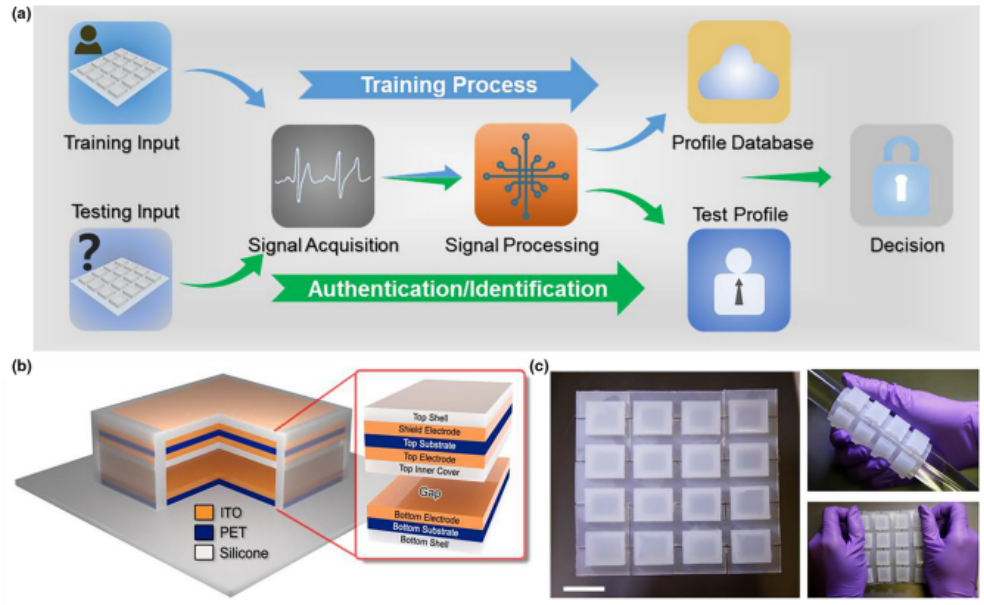

Abbildung 2.1: Zwei-Faktoren-Authentifizierungssystem (a) Systemübersicht. (b) Schema einer einzelnen Triboelektrischen Taste. (c)Bilder der triboelektrischen Tastatur. [61]

Introducing touchstroke: keystroke-based authentication system for smartphones [33]

Für lange Zeit beschäftigte sich die Analyse des Tippverhaltens mit der Eingabe auf klassischen Tastaturen. Georgios Kambourakis *et al.* beschäftigten sich in ihrer Arbeit mit dem Eingabeverhalten auf dem Touchscreen eines Smartphones. Dafür entwickelten sie eine Anwendung, welche auf dem Android-Betriebssystem lief und die Geschwindigkeit der Eingabe, die Distanz zwischen den Eingaben sowie die Pausen- und Haltezeiten bei der Eingabe eines Passwortes aufzeichnete. Basierend auf Testdaten testeten sie drei Klassifizierungsverfahren: Multilayer Perceptron (MLP), k-NN und Random Forest (RF). Der MLP-Klassifikator schied bereits aufgrund des begrenzten Speichers aus. In zwei Szenarien erreichten sie mit dem Random Forest-Klassifikator eine EER von 26% und mit dem k-Nearest Neighbors-Klassifizierer eine EER von 13.6%.

Weitere Arbeiten

Es existieren zahlreiche weitere Arbeiten, welche sich mit dem Thema des Eingabeverhaltens als Merkmal für ein Authentifizierungsverfahren beschäftigen. Darunter *Identification of User Behavioral Biometrics for Authentication using Keystroke Dynamics and Machine Learning* [36], *A machine learning approach to keystroke dynamics based user authentication* [43], *On the discriminability of keystroke feature vectors used in fixed text keystroke authentication* [12], *Improving Accuracy, Applicability and Usability of Keystroke Biometrics on Mobile Touchscreen Devices* [16], *An evaluation of one-class and two-class classification algorithms for keystroke dynamics authentication on mobile devices* [5] und *Passcode Keystroke Biometric Performance on Smartphone Touchscreens is Superior to that on Hardware Keyboards* [31].

2.2 Ziel dieser Arbeit

Die Eignung des Tippverhaltens als biometrisches Merkmal wurde bereits ausgiebig an klassischen Tastaturen sowie an Smartphones untersucht. Ziel dieser Arbeit in Bezug auf die bereits erlangten Erkenntnisse besteht einerseits darin, die Eignung des Eingabeverhaltens an einem Touchscreen als biometrisches Merkmal zu bestätigen, sowie verschiedene Klassifizierungsverfahren zu testen um eine möglichst hohe Genauigkeit zu erreichen. Des Weiteren wurden die Algorithmen in den gesichteten Arbeiten stets als Softwarelösung auf dem Betriebssystem des jeweiligen Smartphones aufgesetzt. Wie bereits in Kapitel 1 erwähnt bedeutet dies eine zusätzliche Schwachstelle im Authentifizierungsverfahren, welche potentiell ausgenutzt werden könnte. Aus diesem Grund soll das Authentifizierungsverfahren in dieser Arbeit in einem abgeschlossenen System implementiert werden, welches sich prinzipiell zwischen jeder Art von Touchscreen und dem dahinter befindlichen Endgerät schalten ließe.

3 Grundlagen

In diesem Kapitel werden für die Verständnis dieser Arbeit notwendige Grundlagen erläutert, wobei ein Grundwissen der Elektro- und Informationstechnik vorausgesetzt wird.

3.1 Zedboard

Das Zedboard der Firma Avnet (s. Abb. 3.1) ist ein Entwicklungs- beziehungsweise Evaluationsboard für den *Xilinx Zynq-7000 All Programmable System on Chip (SoC)*. Es wurde entwickelt, um eine schnelle Prototypen- und Proof-Of-Concept-Enwicklung zu ermöglichen [9].

Der generelle Aufbau des Zedboard ist in Abbildung 3.2 zu sehen. Die Zentrale Komponente des Zedboards ist der *Zynq XC7Z020-CLG484 SoC*. Um eine schnelle Prototypenentwicklung zu ermöglichen befinden sich außerdem diverse Speicher- und Input-/Output-Möglichkeiten auf dem Zedboard. Auf eine Aufzählung aller Komponenten wird an diesem Punkt verzichtet, eine vollständige Beschreibung ist in dem *Hardware User's Guide* [11] zu finden. Besondere Erwähnung aufgrund der Relevanz für diese Arbeit sollen an dieser Stelle finden:

- der High Definition Multimedia Interface (HDMI) Ausgang mit dediziertem HDMI Transmitter,
- das Secure Digital (SD)-Karten Interface,
- das Organic Light-Emitting Diode (OLED)-Display,
- die General-Purpose Inputs/Outputs (GPIOs) sowie
- die PMOD-Anschlüsse.

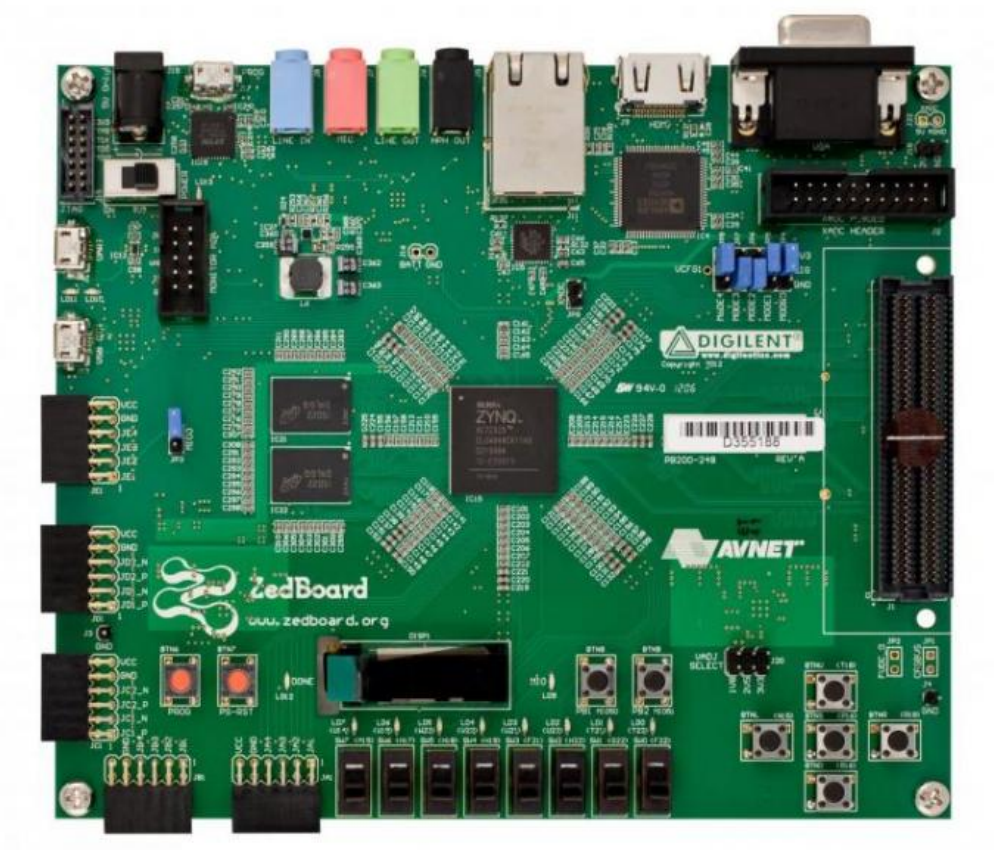

Abbildung 3.1: Zedboard Frontansicht [9]

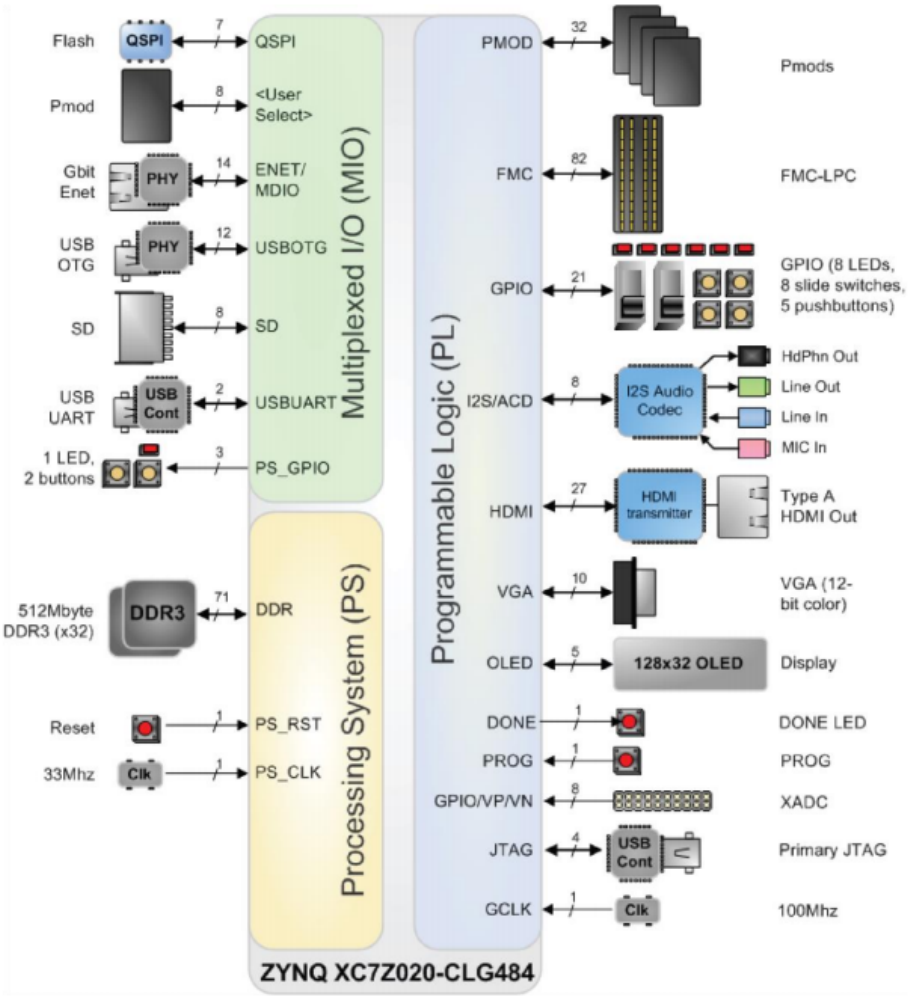

Abbildung 3.2: Zedboard Komponentenübersicht [10]

Wie bereits aus Abbildung 3.2 erkennbar, besteht der Zynq Chip im wesentlichem aus zwei Seiten: der Programmable Logic (PL) und dem Processing System (PS). Diese Aufteilung ist in Abbildung 3.3 genauer aufgeschlüsselt.

Die Programmable Logic besteht aus Configurable Logic Blocks (CLB), Digital Signal Processor (DSP)-Blöcken, programmierbaren I/O-Blöcken, Peripheral Component Interconnect Express (PCIe)-Blöcken, seriellen Transceiver sowie zwei 12-Bit Analog-Digital-Umsetzer.

Das Processing System (PS) ist um die zwei zentralen *ARM Cortex - A9* Prozessoren aufgebaut und beinhaltet unter anderem Serial Peripheral Interface (SPI)-, Inter-Integrated Circuit (I2C)-, Controller Area Network (CAN)- und Universal Asynchronous Receiver Transmitter (UART)-Schnittstellen sowie General-Purpose Inputs/Outputs (GPIOs), On-Chip Speicher und externe Speicherschnittstellen.

Die Kommunikation zwischen Processing System (PS) und Programmable Logic (PL) erfolgt über das Advanced Extensible Interface (AXI), welches in Kapitel 3.3.2 genauer beschrieben wird.

Die Architektur des Zyqns erlaubt die Implementierung von benutzerdefinierter Logik in der PL und benutzerdefinierter Software im PS. Dadurch stehen dem Nutzer die Vorteile beider Entwurfsarchitekturen zur Verfügung. Während es sich anbietet hardwarenahe oder sehr rechenintensive Anwendungen mit Hilfe einer Hardwarebeschreibungssprache (englisch Hardware Description Language (HDl)) zu entwickeln, ist die Entwicklung über eine Hochsprache auf dem PS im allgemeinen wesentlich schneller, da auf zahlreiche Bibliotheken zugegriffen werden kann.

Die Integration von PS und PL auf einen Chip bietet im Vergleich mit einer Zwei-Chip-Lösung (z.B. einen Mikrocontroller und einem Field Programmable Gate Array (FPGA)) entscheidende Vorteile in Bezug auf Bandbreite, Latenz und Energiebedarf [65].

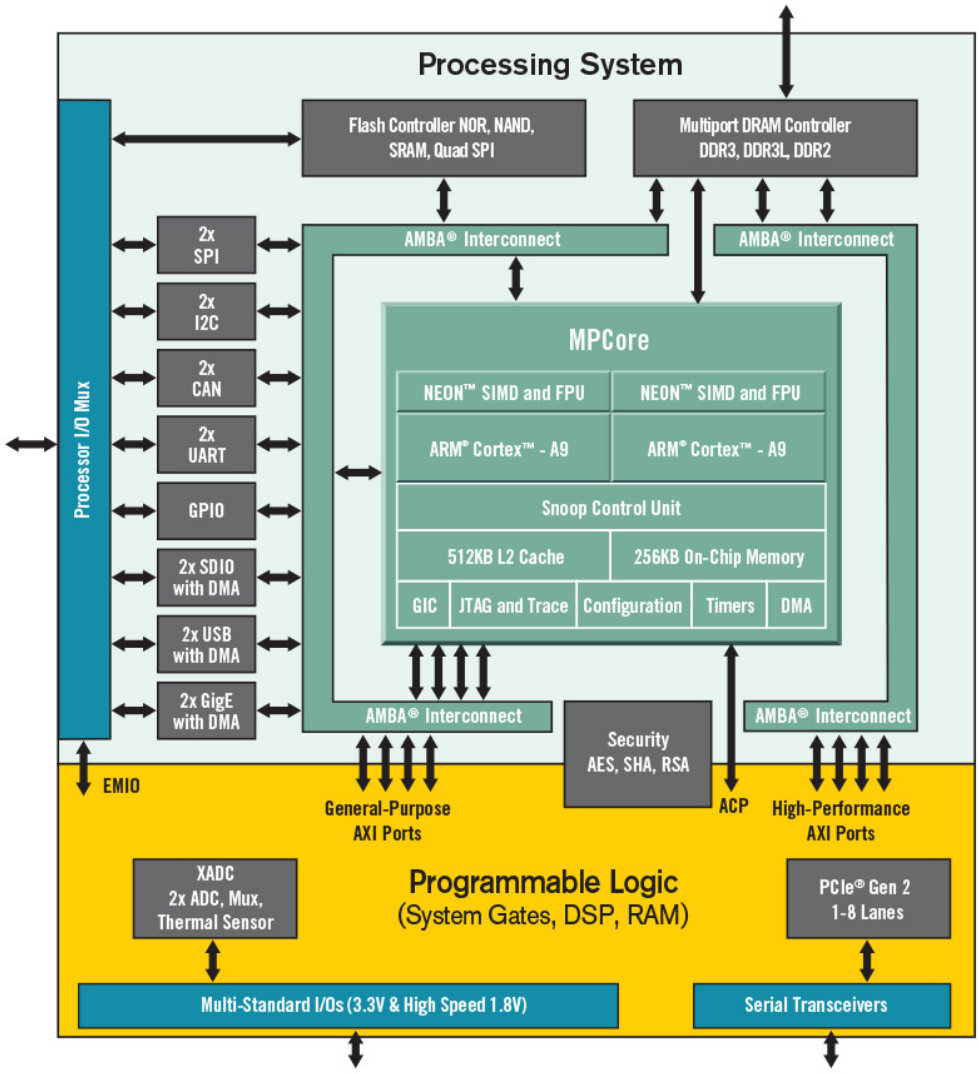

Abbildung 3.3: Übersicht der Zynq-7000 Architektur [64]

3.2 Touchscreen-Technologien

Für die Analyse des Eingabeverhaltens an einen Touchscreen ist es notwendig sich mit deren grundsätzlicher Funktionsweise vertraut zu machen. Da es zahlreiche Varianten an Touchscreen-Technologien gibt, ist es sinnvoll diese gegenüberzustellen um im weiteren Verlauf die für diese Arbeit geeignete Variante bestimmen zu können. Neben den im Folgenden vorgestellten Varianten existieren noch weitere, weniger verbreitete. Dazu zählen unter anderem optische, Infrarotgitter und Biegewellen Touchscreens. Auf diese wird aufgrund ihrer geringen Relevanz nicht weiter eingegangen. Weitere Informationen zu diesen Technologien sind beispielsweise unter [17] zu finden.

3.2.1 Resistive Touchscreens

Resistive Touchscreens reagieren im eigentlichen Sinne nicht auf Berührung sondern auf Druck. Sie bestehen aus einer flexiblen oberen Lage, meist Polyethylenterephthalat (PET), und einer starren unteren Lage, meist Glas. Diese zwei Lagen sind mit einem transparent, leitenden Material beschichtet. In der Regel ist dies Indiumzinnoxid (englisch indium tin oxide, *ITO*). Die Schichten werden von nichtleitenden, kleinen Ausbuchtungen auseinander gehalten (s. Abb. 3.4). Diese Ausbuchtungen sollen verhindern, dass zwischen den Lagen, ohne äußere Einflüsse, eine leitende Verbindung entsteht. Die Dichte beziehungsweise Anzahl dieser Ausbuchtungen bestimmt den Aktivierungsdruck des Touchoverlays. Wird dieser überschritten entsteht eine leitende Verbindung zwischen den Lagen, welche messtechnisch erkannt werden kann. [14]

3.2.1.1 4-Draht Anschluss

Die einfachste Variante ein resistives Touchoverlay zu beschalten ist der 4-Draht- (englisch 4-Wire-) Anschluss. Dabei werden je zwei Anschlüsse an den leitenden Lagen angebracht.

Um bei Aktivierung des Touchoverlays die X und Y Koordinaten zu ermitteln, wird an je ein Paar der gegenüberliegenden Anschlüsse die Betriebsspannung angelegt. Da aufgrund des Drucks die beiden leitenden Lagen des Touchoverlays verbunden sind, kann an einem der verbleibenden Anschlüsse eine Spannung abgegriffen werden, welche zur X beziehungsweise Y Koordinate proportional ist (s. Abb. 3.5 und 3.6). [30]

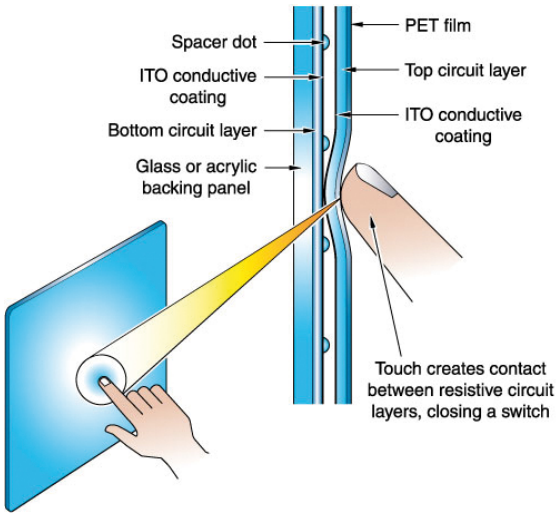

Abbildung 3.4: Lagen eines resistiven Touchoverlays [17]

Im Idealfall wäre der Zusammenhang zwischen abgegriffener Spannung und Koordinate der Berührung über das gesamte Touchoverlay linear. Bei realen Touchoverlays ist dies gerade an den Rändern nur bedingt der Fall. Wie gut ein Touchoverlay in diesem Sinne ist, gibt die Eigenschaft *Linearität* an, welche in den meisten Datenblättern zu finden ist. In der Regel ist sie in Prozent angegeben und beschreibt die Abweichung vom idealen Verlauf.

Ein Nachteil des 4-Draht Anschlusses ist, dass zur Bestimmung von einer der Koordinaten ein linearer Spannungsverlauf auf der äußeren leitenden Lage benötigt wird. Da diese sich auf der flexiblen Lage des Touchoverlays befindet, welche während des Gebrauchs immer wieder verformt wird, kommt es früher oder später zu mikroskopischen Rissen in dieser leitenden Schicht. Dadurch werden deren elektrischen Eigenschaften verändert, was sich negativ auf die Genauigkeit und Linearität des Touchoverlays auswirkt. [14]

3.2.1.2 5-Draht Anschluss

Dieser Nachteil kann mit der Verwendung des 5-Draht- (englisch 5-Wire-) Anschlusses vermieden werden. Hierbei werden vier der Drähte mit je einer Ecke der hinteren leitenden Schicht verbunden (Anschlüsse A, B, C und D in Abb. 3.7) und der fünfte mit der vorderen leitenden Schicht.

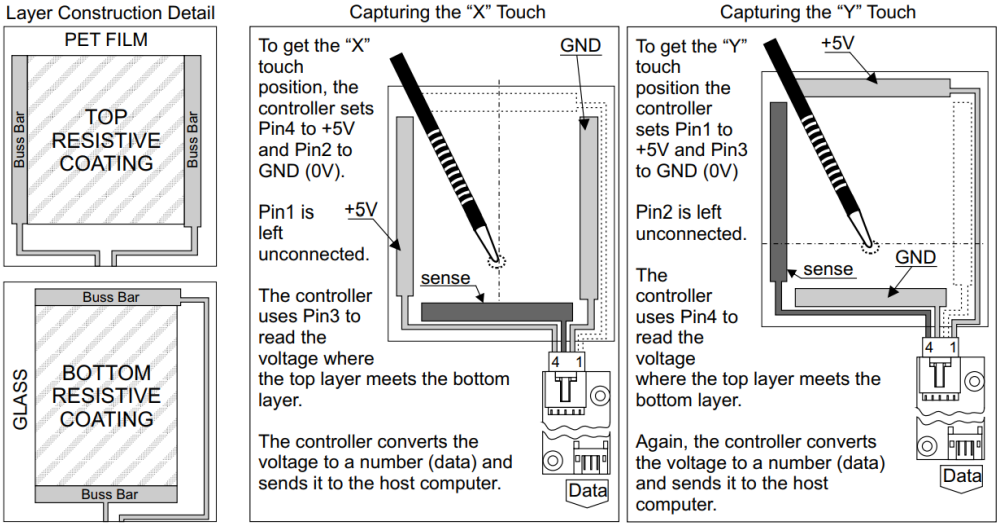

Abbildung 3.5: 4-Draht Anschluss eines resistiven Touchscreens [30]

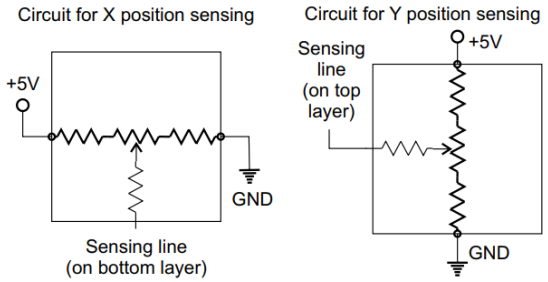

Abbildung 3.6: Schaltbild: 4-Draht Anschluss eines resistiven Touchscreens [30]

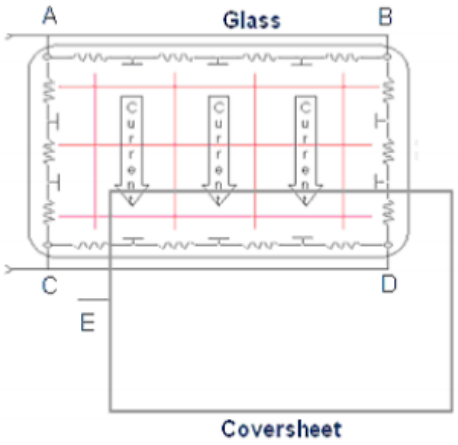

Abbildung 3.7: 5-Draht Anschluss eines resistiven Touchscreens [14]

Zum Messen der X Koordinate wird die Betriebsspannung an die Ecken A und C angelegt und Masse an die Ecken B und D. Die zur X Koordinate proportionale Spannung kann nun an E gemessen werden. Zum Messen der Y Koordinate wird entsprechend vorgegangen (Betriebsspannung an A und B, Masse an C und D). Bei dieser Art des Anschlusses wird die hintere leitende Schicht immer für das Einprägen eines linearen Spannungsabfalls genutzt und die vordere lediglich zum Messen der Spannung.

Beschädigungen der vorderen leitenden Lage können so zwar immer noch auftreten, diese haben dann allerdings keinen Einfluss mehr auf die Linearität. [14]

3.2.2 Kapazitive Touchscreens

Kapazitive Touchscreens reagieren im allgemeinen auf die Berührung mit einem leitenden Objekt, beispielsweise mit einem Finger. Im Vergleich zu resistiven Touchoverlays besitzen sie eine deutlich bessere Lichtdurchlässigkeit. Es gibt verschiedene Varianten von kapazitiven Touchscreens, welche sich grob in die Kategorien der oberflächenkapazitiven und der projiziert kapazitiven Technologien einteilen lassen. [14]

3.2.2.1 Oberflächenkapazitive Technologie

Bei einem Oberflächen kapazitiven Touchscreen wird eine Spannung an alle Ecken angelegt, sodass sich auf der Oberfläche ein gleichmäßiges elektrisches Feld bildet. Es wird

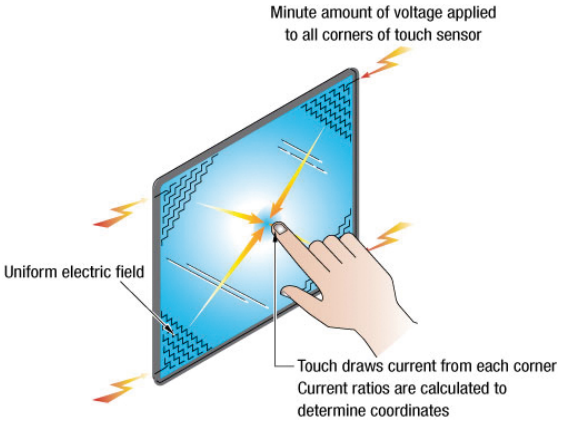

Abbildung 3.8: Aufbau eines oberflächenkapazitiven Touchscreens [17]

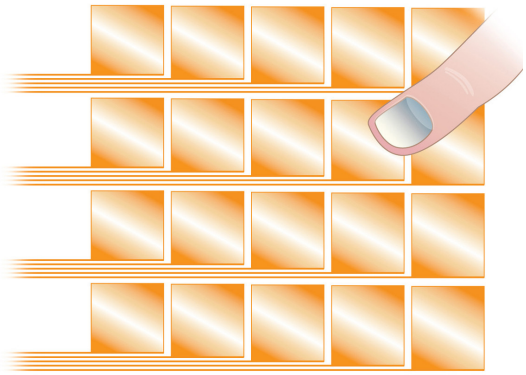

Abbildung 3.9: Aufbau eines eigenkapazitiven Touchscreens mit Multi-Pad [17]

auf die Oberfläche eine gleichmäßig verteilte Ladung aufgebracht. Bei der Berührung mit einem elektrisch leitendem Objekt wird diese Ladung nun abgeleitet. Ein Strom fließt von den vier Ecken über den Kontaktpunkt. Dabei ist der Strom, welcher von einem Anschluss aus fließt, umgekehrt proportional zu der Entfernung von diesem Anschluss zum Kontaktpunkt. Die vier Eckströme werden gemessen und über deren Verhältnis wird die Position des Kontaktpunktes bestimmt. [17]

Mit dieser Technologie kann nur ein Kontaktpunkt aufgelöst werden und eine Beschädigung der Oberfläche hat eine Beeinträchtigung der Funktion zur Folge.

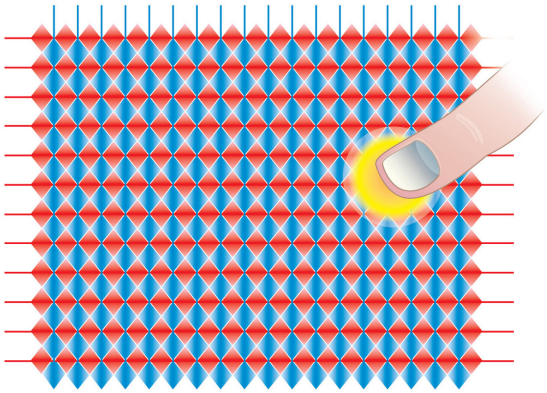

Abbildung 3.10: Aufbau eines gegenkapazitiven Touchscreens mit Zeilen und Spalten [17]

3.2.2.2 Projiziert kapazitive Technologie

Bei projiziert kapazitiven Touchscreens befindet sich eine passive schützende Schicht zwischen der aktiven Lage und dem Finger. Dadurch wird die Widerstandsfähigkeit erhöht. Die aktive Lage besteht aus einem Raster von Elektroden, welche entweder als einzelne Elektroden ausgeführt werden (Multi-Pad, s. Abb. 3.9) oder in Zeilen und Spalten angeordnet sind (s. Abb. 3.10) [17]. Von der Vielzahl an projiziert kapazitiven Technologien wird nun genauer auf die gegenkapazitive Ausführung eingegangen.

Projiziert gegenkapazitive Touchscreens machen sich die Gegenkapazität zwischen den Schnittpunkten einzelner Zeilen und Spalten zu Nutze. Damit kann jeder Schnittpunkt während eines Messdurchlaufes einzeln gemessen werden. Somit ist es möglich mehrere simultane Kontaktpunkte zu erkennen. Dabei wird in jedem Messdurchlauf die Kapazität jedes Messpunktes gemessen. Wird ein leitender Gegenstand in die Nähe eines Schnittpunktes gebracht, verringert sich die Kapazität zwischen der entsprechenden Zeile und Spalten (s. Abb. 3.11). Unterschreitet diese Kapazität einen Grenzwert, wird dies als Berührung interpretiert. Die Auflösung des Touchscreens kann bei diesem Verfahren sehr einfach durch Interpolation der einzelnen Messwerte verbessert werden. [17]

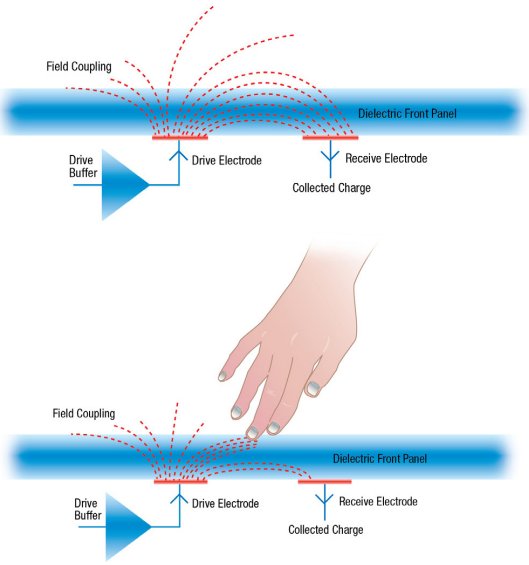

Abbildung 3.11: Funktionsweise eines projiziert gegenkapazitiven Touchscreens [17]

3.3 Bussysteme

In diesem Kapitel werden die in der Implementierung verwendeten Bussysteme vorgestellt. Das Serial Peripheral Interface (SPI) wird dabei für die Kommunikation zwischen dem Touchscreen-Anschlussboard und dem Programmable Logic (PL) Teil des Zedboards verwendet. Das Advanced Extensible Interface (AXI) wird für die Kommunikation zwischen dem PL- und PS-Teil des Zedboards verwendet.

3.3.1 Serial Peripheral Interface - SPI

SPI ist ein serielles, synchrones, vollduplexfähiges Master-Slave basiertes Interface, welches vorwiegend für die Kommunikation zwischen Mikrocontrollern untereinander oder die Kommunikation zwischen Mikrocontrollern und externer Peripherie eingesetzt wird. Für SPI existiert kein festgeschriebener Industriestandard. Im folgendem werden die weit verbreiteten Konventionen und die allgemeine Funktionsweise dieses Interfaces dargestellt.

Eine SPI Konfiguration besteht aus genau einem Master und mindestens einem Slave. Dabei werden mindestens die vier Signale SPI Clock (SCLK), Chip Select (CS), Master

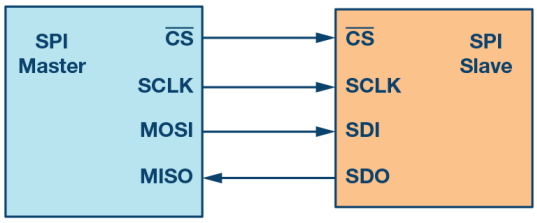

Abbildung 3.12: SPI Konfiguration mit einem Slave [21]

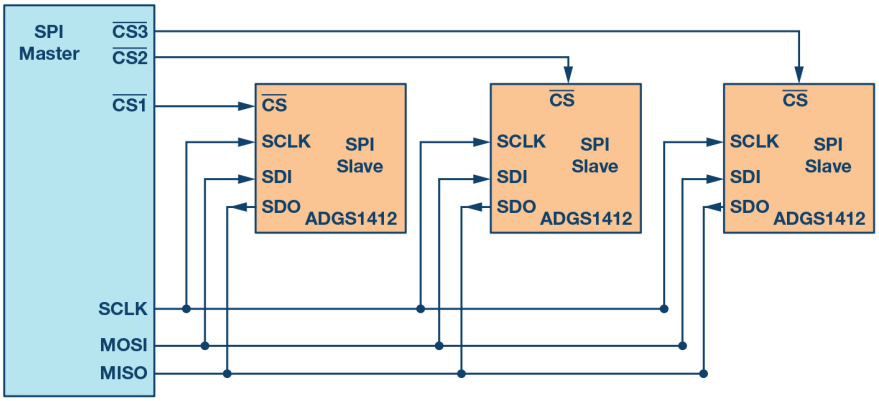

Abbildung 3.13: SPI Konfiguration mit mehreren Slaves [21]

out - Slave in (MOSI) und Master in - Slave out (MISO) benötigt. Ist mehr als ein Slave vorhanden werden zusätzliche CS Leitungen benötigt (s. Abb. 3.13).

Die Kommunikation kann nur vom Master aus initiiert werden. Durch einen Lowpegel an der entsprechenden CS Leitung wird der entsprechende Slave ausgewählt. Der Master generiert ein Clock-Signal, welches über die SCLK Leitung an die Slaves übertragen wird. Diese werden somit auf den Master synchronisiert. Auf der MOSI Leitung werden die Daten seriell vom Master zum Slave übertragen. Gleichzeitig können Daten vom Slave zum Master auf der MISO Leitung übertragen werden.

Das Shiften der Daten auf den Bus und das Einlesen der Daten vom Bus wird bei steigenden oder fallenden Flanken des SCLK Signal durchgeführt. Der Pegel des SCLK Signals im Idle Zustand kann entweder High oder Low sein. Die möglichen Kombinationen in Abhängigkeit von Clock Polarity (CPOL) und Clock Phase (CPHA) sind in Tabelle 3.1 zu finden. Die genauen Timing-Spezifikationen sind in jedem Fall einem entsprechenden Handbuch beziehungsweise Datenblatt zu entnehmen. Abbildung 3.14 zeigt beispielhaft die Übertragung von je einer Nachricht vom Master zum Slave und umgekehrt. Gezeigt ist hier der Mode 0. Die gestrichelten grünen Linien zeigen den Beginn und das Ende

SPI Mode	CPOL	CPHA	SCLK Pegel im Idle Zustand	Ausgabe der Daten bei Flanke von SCLK	Einlesen von Daten bei Flanke von SCLK
0	0	0	Low	fallend	steigend
1	0	1	Low	steigend	fallend
2	1	0	High	fallend	steigend
3	1	1	High	steigend	fallend

Tabelle 3.1: SPI Modi in Abhängigkeit von CPOL und CPHA

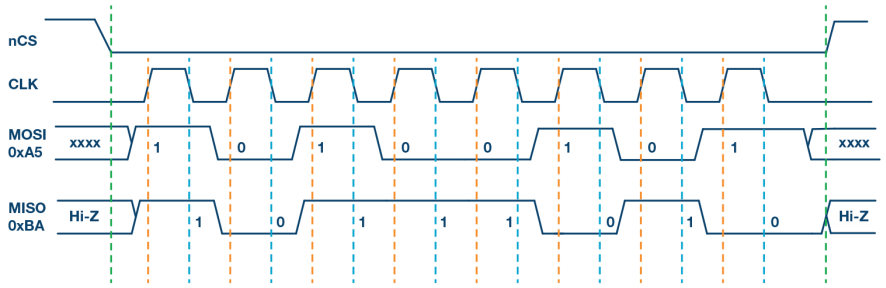

Abbildung 3.14: Beispiel einer Nachrichtenübertragung mit SPI [21]

der Übertragung. Die steigenden Flanken, bei denen die Daten eingelesen werden, sind durch die orangen Linien gekennzeichnet. Mit den blauen Linien sind die fallenden Flanken gekennzeichnet. Bei diesen werden die Daten auf den Bus geschrieben. [21] [48]

3.3.2 Advanced eXtensible Interface - AXI

Das Advanced Extensible Interface (AXI) ist ein Bestandteil der ARM Advanced Microcontroller Bus Architecture (AMBA)-Familie und wird unter anderem von Xilinx für die Kommunikation zwischen verschiedenen Funktionsblöcken auf einem Chip eingesetzt. Es handelt sich um einen weit verbreiteten Industriestandard.

Es existieren verschiedene Versionen dieses Standards. Zur Zeit ist die Version AXI4 am verbreitetsten. Diese wird auch in der Implementierung verwendet und im Folgenden kurz vorgestellt.

Das AXI Interface verbindet einen einzelnen AXI-Master mit einem einzelnen AXI-Slave. Mehrere Master und Slaves können mit AXI-Interconnect Blöcken zusammengeschaltet werden (s. Abb. 3.15).

Es wird zwischen AXI4, AXI4-Lite und AXI4-Stream Interfaces unterschieden. AXI4 und AXI4-Lite Interfaces werden beide für Memory Mapped I/Os verwendet. AXI4-Stream Interfaces werden für kontinuierliche Datenübertragung verwendet. Der Unterschied zwischen Memory Mapped I/O und kontinuierlicher Datenübertragung besteht im wesentlichen darin, dass die Datenmenge pro Transaktion, welche beim Memory Mapped I/O übertragen wird, begrenzt ist. Es muss immer die Lese- beziehungsweise Schreibadresse übertragen werden. Während sich ein AXI4-Stream Interface bei kontinuierlichen Datenflüssen, wie etwa den Videodaten einer Kamera anbietet, werden AXI4 und AXI4-Lite Interfaces für das Auslesen von Eventdaten oder dem Schreiben von Kontrolldaten verwendet.In den Abbildungen 3.16 und 3.17 sind beispielhaft ein Schreib- und ein Lesevorgang für ein AXI4- beziehungsweise AXI4-Lite-Interface visualisiert. [62]

In Abbildung 3.18 ist beispielhaft ein von Vivado zur Verfügung gestellter IP-Core mit AXI4-Lite Interface zu sehen. Dieser kann über den Slave-AXI-Port S_AXI mit einem Master verbunden werden. In diesem Beispiel können die am Port $GPIO$ anschließbaren Ausgänge dann vom AXI-Master geschrieben und gelesen werden. In Tabelle 3.2 sind die Signale des AXI-Interfaces beschrieben.

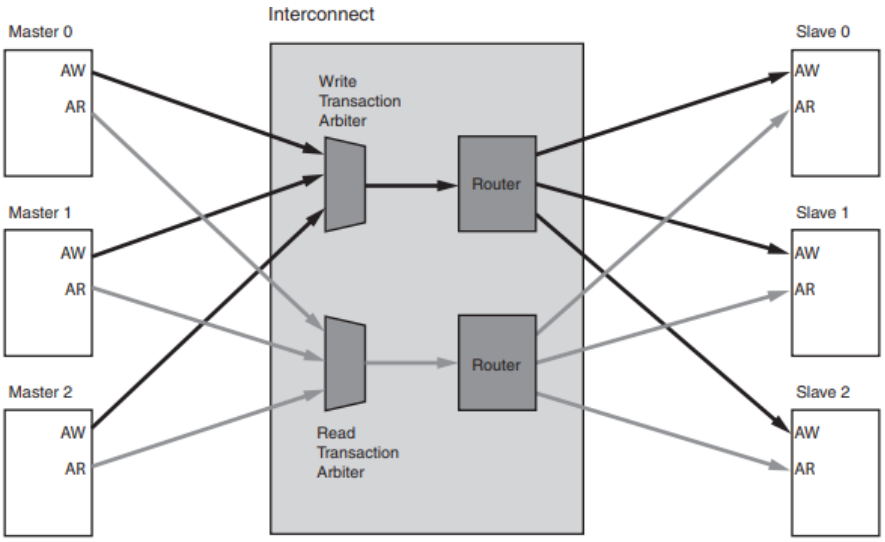

Abbildung 3.15: Verbinden von AXI Mastern und Slaves über ein AXI-Interconnect [62]

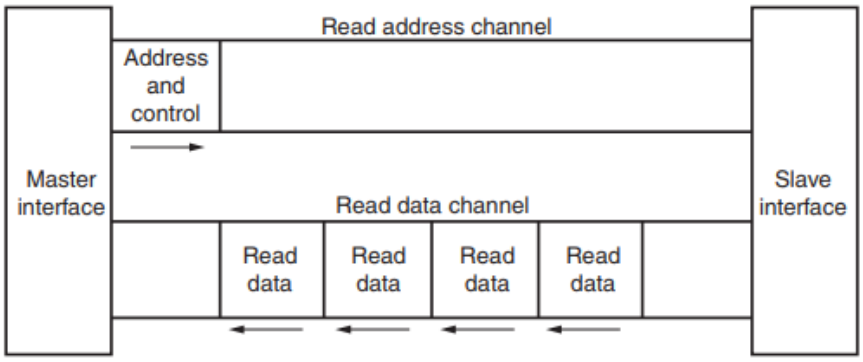

Abbildung 3.16: Auslesen von Slave-Daten über das AXI4-Interface [62]

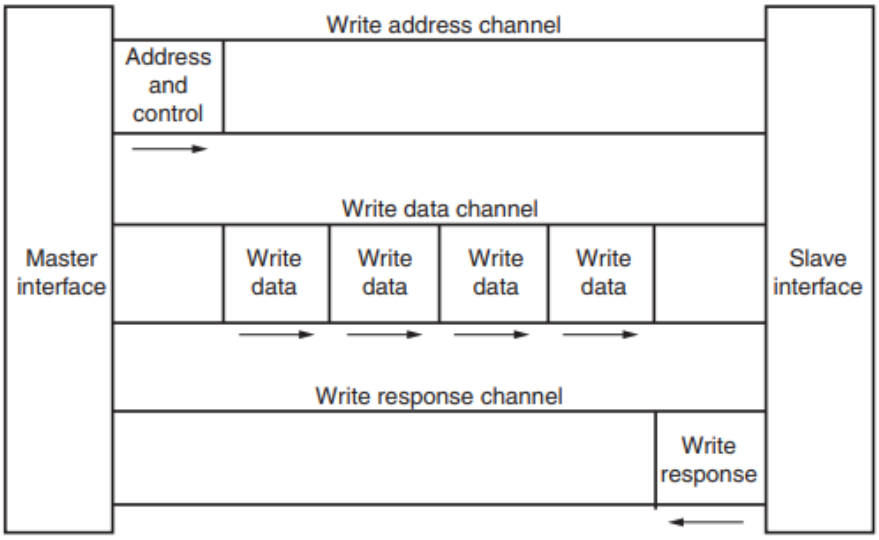

Abbildung 3.17: Schreiben von Daten in den Slave-Speicher über das AXI4-Interface [62]

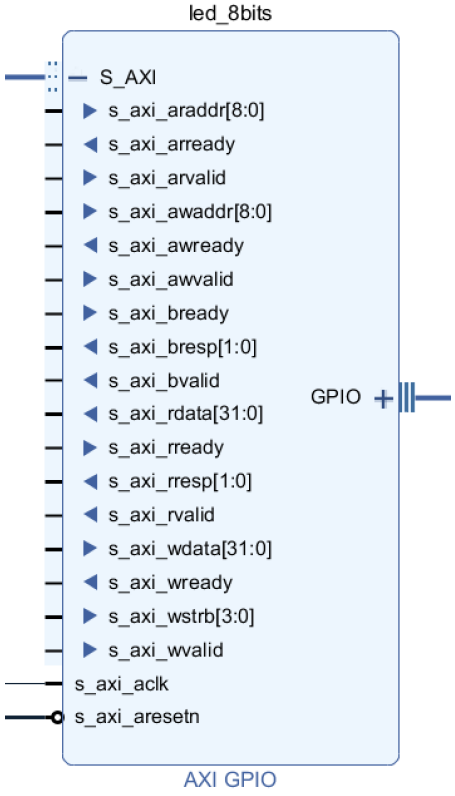

Abbildung 3.18: Beispiel eines IP-Cores mit AXI4-Lite Interface

Signalname	Beschreibung
araddr[8:0]	Leseadresse: Wird vom Master gesetzt und gibt an welche Register gelesen werden sollen.
arready	Wird vom Slave gesetzt. Teilt dem Master mit, dass der Slave bereit für eine Adresseingabe und die dazugehörigen Kontrollsignale ist.
arvalid	Wird vom Master gesetzt. Gibt dem Slave an, dass die anliegende Leseadresse und die Kontrollsignale gültig sind.
awaddr[8:0]	Schreibadresse: Wird vom Master gestzt und gibt an in welches Register geschrieben werden soll.
awvalid	Wird vom Master gesetzt. Gibt dem Slave an, dass die anliegende Schreibadresse und die Kontrollsignale gültig sind.
bready	Wird vom Master gesetzt. Gibt dem Slave an, dass der Master bereit ist Schreibantwort zu erhalten.
bresp[1:0]	Schreibantwort: Wird vom Slave gesetzt und teilt dem Master mit wie der Schreibvorgang verlaufen ist.
rdata[31:0]	Lesedaten.
rready	Wird vom Master gesetzt. Gibt dem Slave an, dass die Lesedaten und die Leseantwort empfangen werden können.
rresp[1:0]	Leseantwort: Wird vom Slave gesetzt und teilt dem Master mit wie der Lesevorgang verlaufen ist.
rvalid	Wird vom Slave gestzt. Teilt dem Master mit, dass sich die angeforderten Lesedaten auf dem Kanal befinden.
wdata[31:0]	Schreibedaten
wready	Wird vom Slave gestzt. Teilt dem Master mit, dass der Slave breit ist die Schreibdaten zu empfangen.
wstrb[3:0]	„Strobes“.Wird vom Master gestzt. Gibt an welches Byte des Datenbusses gültige Daten enthält.
wvalid	Wird vom Master gestzt. Gibt an, dass die Schreibdaten und Strobes (wstrb) gültig sind.
aclk	AXI-Taktsignal
aresetn	AXI-Resetsignal

Tabelle 3.2: Signalbeschreibung des AXI4-Lite-Interfaces [6]

3.4 Grundlagen der Klassifizierung

Da es sich bei dem Authentifizierungsalgorithmus, welcher in dieser Arbeit verwendet wird, um einen Klassifizierungsalgorithmus handelt, werden in diesem Kapitel kurz die Grundlagen der Klassifizierung beschrieben.

Klassifizierung ist der Vorgang der Einordnung eines Datenpunktes in eine Klasse. Diese Klassen (engl. Classes) werden auch Labels oder Ziele (engl. Targets) genannt. Klassifikatoren sind trainierte Modelle, welche über eine Funktion f (dem Klassifizierungsalgorithmus) aus den Eingangsdaten x (den Attributen oder Merkmalen des Datenpunktes) die diskrete Ausgangsvariable y (die Klasse des Datenpunktes) bestimmen sollen. [8]

Die Klassifizierung gehört zu den Supervised Machine Learning (ML) Algorithmen. Diese laufen im Grunde alle gleich ab (s. Abb. 3.19). Man benutzt Trainingsdaten (Training Data Input) und die dazugehörigen Label (Target Output) und trainiert basierend auf diesen Daten ein Modell. Dieses Modell kann dann für neue Daten, für welche noch keine Label existieren, eine Vorhersage treffen und diesen eines der Label zuordnen. [8]

Die Anwendungsbereiche der Klassifizierung sind vielfältig. So können Klassifikatoren eingesetzt werden um zu entscheiden, ob es sich bei einer E-Mail um Spam handelt oder nicht [8]. Sie können verwendet werden um die Art einer Pflanze vorherzusagen [66], oder um anhand des Eingabeverhaltens vorherzusagen, welche Person diese Eingabe durchgeführt hat [43].

Haben wir beispielsweise Daten aus den zwei Klassen **A** und **B** (s. Abb. 3.20), können wir versuchen die Klassen durch eine Gerade voneinander zu trennen. Sollen wir nun für einen neuen Datenpunkt vorhersagen, ob er von Klasse **A** oder Klasse **B** stammt, basiert unsere Entscheidung einfach darauf auf welcher Seite der Geraden der Punkt liegt.

Bereits an diesem Beispiel ist zu erkennen, dass die Klassifizierung in den vielen Fällen kein Problem ist, welches perfekt gelöst werden kann. In unserem Beispiel konnten wir die Gerade nicht so legen, dass alle Punkte der Klasse **A** auf einer und alle Punkte der Klasse **B** auf der anderen Seite liegen. Dies könnte viele verschiedene Gründe haben. Vielleicht sind die Trainingsdaten verunreinigt oder haben Ausreißer. Es könnte aber auch sein, dass die Klassen sich über die zwei Merkmale, welche wir betrachten, nicht eindeutig unterscheiden lassen.

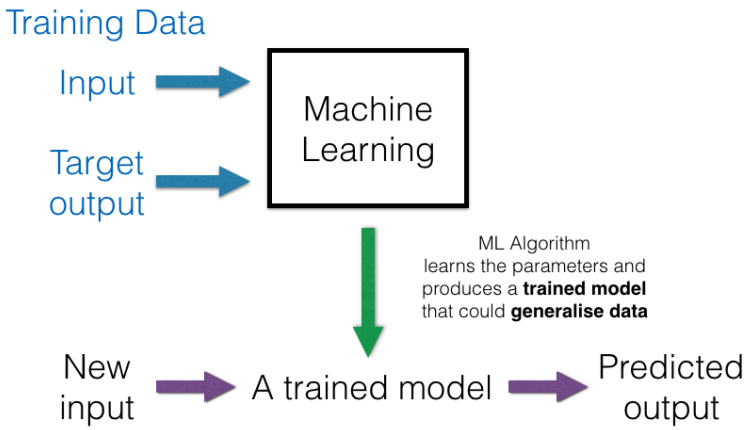

Abbildung 3.19: Allgemeiner Ablauf des *Supervised Machine Learnings* [66]

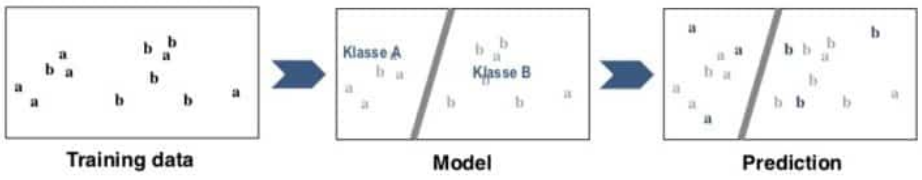

Abbildung 3.20: Beispiel eines Klassifikators [50]

3.4.1 Bewertung von Klassifikatoren

Damit die Klassifikatoren leichter bewertet werden können, werden meist nicht alle Daten zum trainieren des Modells benutzt. Stattdessen werden zum Beispiel nur 60% der Daten verwendet. Anschließend wird das Modell auf die restlichen Daten angewendet und die vom Modell vorhergesagten Klassen mit den korrekten Klassen verglichen. [8]

In Abbildung 3.21 ist dieses Verfahren zu sehen. Die Daten der beiden Klassen (**Rot** und **Blau**) werden aufgeteilt in die Trainingsdaten (Kreise) und die Testdaten (Dreiecke). Mit den Trainingsdaten wurden dann verschiedene Klassifikatoren trainiert. Die Entscheidungsgrenzen der jeweiligen Modelle sind durch die Farbe des Hintergründe gekennzeichnet. Anschließend werden die Klassifikatoren auf die Testdaten (Dreiecke) angewendet um vorherzusagen, ob es sich um Punkte aus der Klasse **Rot** oder **Blau** handelt. Die mittlere Genauigkeit der Modelle ist in der unteren rechten Ecke dargestellt.

Die **Genauigkeit** (engl. **Accuracy**) gibt an in wie viel Prozent der Fälle das Modell die Klasse des Datenpunktes korrekt vorhergesagt hat. Diese Charakteristik ist allerdings nicht in allen Fällen aussagekräftig. Wird beispielsweise ein Modell mit sehr ungleich verteilten Daten trainiert, können wir über die Accuracy nicht zwangsweise beurteilen, wie geeignet unser Modell ist. Nehmen wir an wir hätten einen Datensatz mit 9900 Punkten der Klasse **Rot** und 100 Punkten der Klasse **Blau**. Ein Klassifikator welcher *alle* Datenpunkte unabhängig von deren Merkmalen als **Rot** vorhersagen würde, hätte eine Accuracy von 99%. Betrachtet man also nur diese Charakteristik würde man zu dem Schluss kommen, dass der Klassifikator „gut“ funktioniert. Aus diesem Grund ist es notwendig weitere Charakteristiken zu betrachten.

Etabliert und aussagekräftig sind beispielsweise die **Precision** (deut. Präzision) und der **Recall** (deut. Erinnerungsvermögen) [45]. Diese können für alle Klassen eines Modells einzeln angegeben werden. Für die Berechnung dieser Werte sind die vier Werte: True Positive (TP), False Positive (FP), True Negative (TN) und Flase Negative (FN) notwendig. Bleiben wir bei unserem Beispiel mit den Klassen **Rot** und **Blau** und betrachten die Klasse **Rot** sind dies:

- True Positive (TP): Die Testpunkte, welche das Modell als **Rot** vorgesagt hat und die auch wirklich aus der Klasse **Rot** sind,
- False Positive (FP): Die Testpunkte, welche das Modell als **Rot** vorgesagt hat, die aber *nicht* aus der Klasse **Rot** sind,

- True Negative (TN): Die Testpunkte, welche das Modell als nicht **Rot** vorgesagt hat und die auch wirklich *nicht* aus der Klasse **Rot** sind,
- Flase Negative (FN): Die Testpunkte, welche das Modell als nicht **Rot** vorgesagt hat, die aber aus der Klasse **Rot** sind.

Da unser Beispiel lediglich die Klassen **Rot** und **Blau** hat, könnte man natürlich anstelle von „nicht aus Klasse **Rot**“ auch einfach „aus Klasse **Blau**“ sagen. Besteht das Problem allerdings aus mehr als zwei Klassen ist die Beachtung dieser Formulierung wichtig.

Die **Precision** eines Modells für eine Klasse berechnet als

$$P = \frac{TP}{TP + FP},$$

der **Recall** als

$$R = \frac{TP}{TP + FN}.$$

In Worten bedeutet dies für die Klasse **Rot**:

- Precision: Wie viel Prozent der Datenpunkte, welche das Modell als **Rot** vorgesagt hat sind auch wirklich aus der Klasse **Rot**?
- Recall: Wie viel Prozent der Datenpunkte, die wirklich aus der Klasse **Rot** sind, hat das Modell auch als solche erkannt?

Für das vorliegende Beispiel mit den 10000 Datenpunkten und der Genauigkeit von 99% ergeben sich die Werte in Tabelle 3.3. Anhand dieser Werte ist eindeutig zu erkennen, das unser Modell nicht geeignet ist. Precision und Recall für die Klasse **Rot** sind zwar fast perfekt, für die Klasse **Blau** jedoch unbrauchbar.

Klasse	Precision	Recall
Rot	0.99	1
Blau	0	0

Tabelle 3.3: Precision und Recall für Einführungsbeispiel

3.4.2 Klassifizierungsalgorithmen

Im Folgendem werden fünf der am weit verbreitetsten Klassifizierungsalgorithmen vorgestellt.

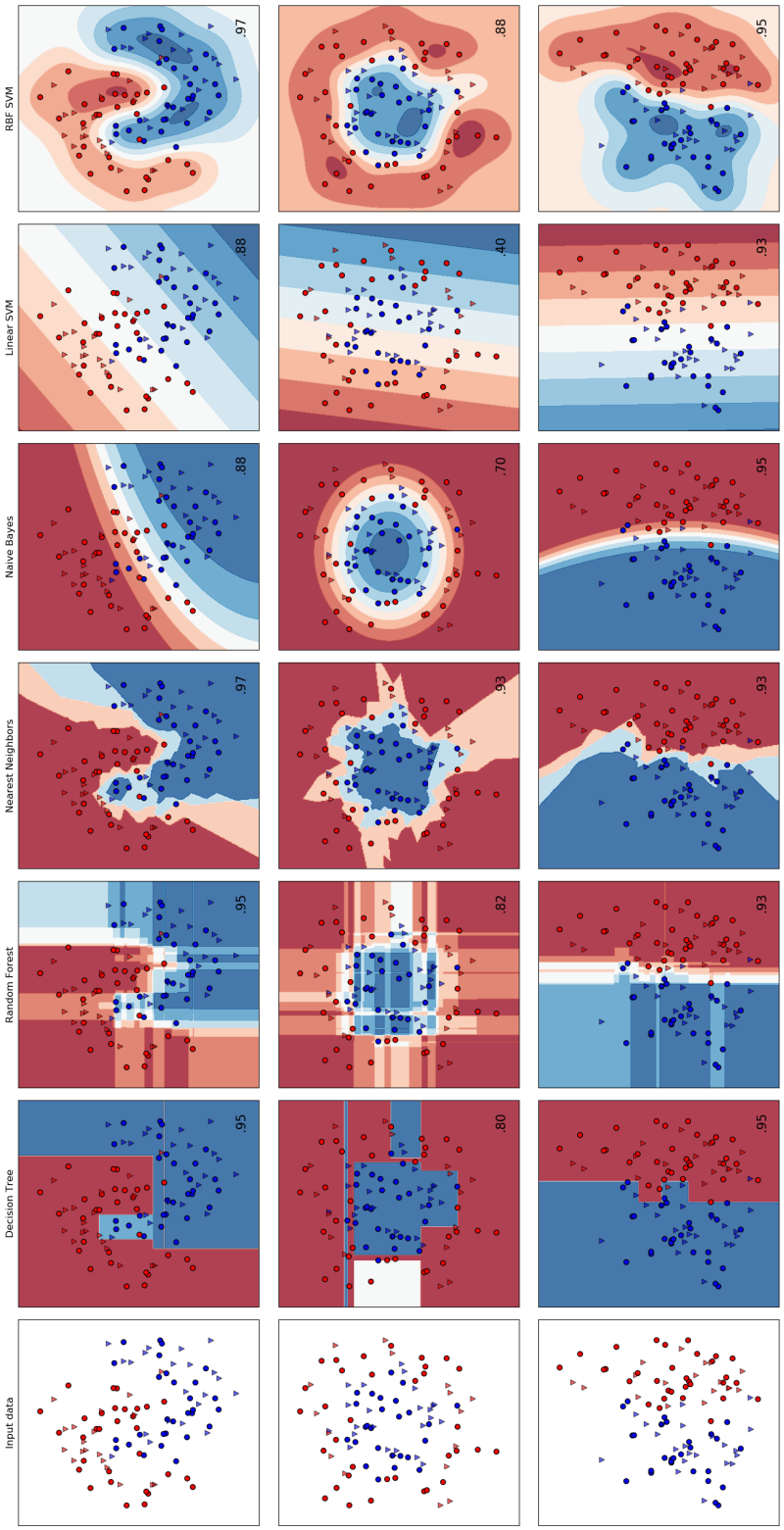

Abbildung 3.21: Verschiedene Klassifizierungsalgorithmen im Vergleich. Klassifizierung von drei künstlichen Datensätzen mit jeweils zwei Klassen (Rot und Blau). Training der Modelle mit 60% der Datenpunkte (Kreise), Test mit 40% der Datenpunkte (Dreiecke). Mittlere Genauigkeit in der unteren rechten Ecke. Basierend auf [51].

In Abbildung 3.21 sind Visualisierungen dieser Algorithmen für drei künstliche Datensätze zu sehen. Dabei steht jede Zeile für einen der Datensätze und die Spalten für einen der Algorithmen. Eine Ausnahme ist die erste Spalte. In dieser sind die Datensätze an sich dargestellt. Betrachtet man nun also eine einzelne Zeile kann die Performance der verschiedenen Algorithmen für einen Datensatz vergleichen. Betrachtet man eine Spalte kann man die Performance eines Algorithmus für verschiedene Datensätze vergleichen. In der unteren rechten Ecke ist die mittlere Genauigkeit des Modells angegeben.

3.4.2.1 Decision Tree

Der *Decision Tree*, zu Deutsch Entscheidungsbaum, ist in der zweiten Spalte der Abbildung 3.21 zu sehen. Er basiert, wie der Name vermuten lässt, auf einer Baumstruktur (s. Abb. 3.22). Dabei wird die Trainingsmenge an den Entscheidungsknoten (engl. Decision Nodes) anhand des bedeutungsvollsten Merkmals aufgeteilt. Dies wird wiederholt, bis alle Trainingsdatenpunkte in ihre entsprechende Klasse aufgeteilt wurden oder eine einstellbare maximale Tiefe des Entscheidungsbaums erreicht wurde. Es besteht auch die Möglichkeit eine minimale Anzahl von Datenpunkten festzulegen, welche in jedem der Blätter (engl. Leaf Nodes) vorhanden sein muss. Dies kann sinnvoll sein um ein Overfitting, also ein Überanpassen, des Modells zu verhindern [66]. Ein Beispiel für ein solches Overfitting ist in Abbildung 3.21, Zeile 1, Spalte 2 zu sehen. Der Decision Tree teilt die rechte Blaue Fläche erneut, um den *einen* Trainingspunkt der Klasse **Rot** korrekt einzuordnen. Betrachtet man die Trainingsdaten mit dem Auge, scheint diese Aufteilung nicht sinnvoll. Vor- und Nachteile des Decision Trees sind in Tabelle 3.4 zusammengefasst[1].

[1] Greedy-Algorithmen oder zu deutsch gierige Algorithmen lösen schrittweise. Dabei wird bei jedem Schritt die, zu diesem Zeitpunkt, optimale Entscheidung getroffen. Dies garantiert allerdings nicht, dass die Lösung global gesehen optimal ist [57].

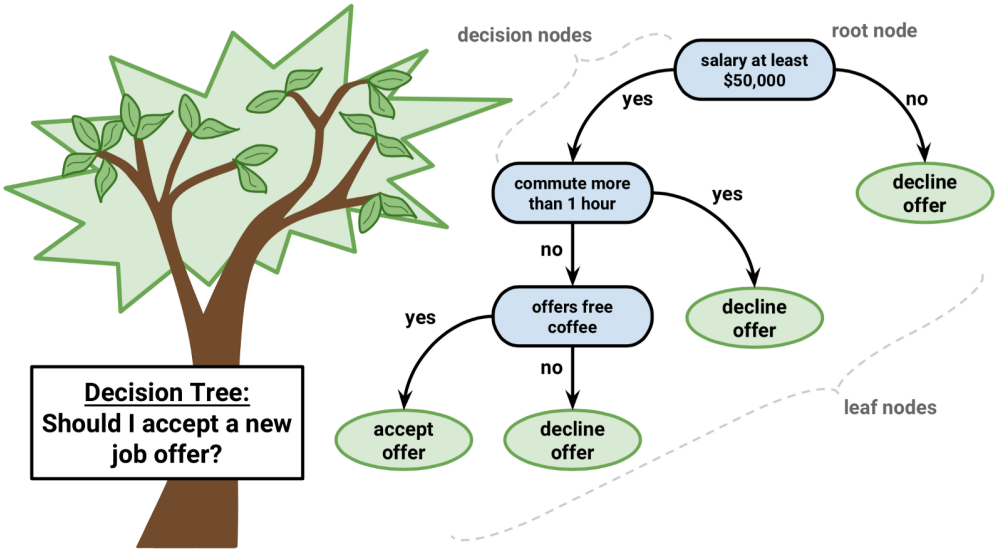

Abbildung 3.22: Funktionsweise eines Decision Tree Klassifikators [8]

Vorteile	Nachteile
Einfach zu verstehen und zu visualisieren. Aussagekräftige Merkmale können schnell identifiziert werden.	Anfällig für Overfitting. Da es sich um einen Greedy-Algorithmus handelt ist eine optimale Lösung nicht garantiert.
Relativ unempfindlich gegenüber Ausreißern.	Bei kontinuierlichen Variablen besteht die Gefahr, dass Informationen verloren gehen.
Es werden keine Annahmen über die Daten getroffen.	

Tabelle 3.4: Vor- und Nachteile des Decision Trees [66]

3.4.2.2 Random Forest

Der *Random Forest*, zu Deutsch „Zufallswald", ist in der dritten Spalte der Abbildung 3.21 zu sehen. Er gehört zur Gruppe der Ensemble-Algorithmen und besteht aus mehreren Decision Trees. Während des Trainierens des Modells werden die Trainingsdaten in mehrere Untergruppen aufgeteilt. Für jede dieser Untergruppen wird dann ein Decision Tree trainiert. Soll ein neuer Datenpunkt klassifiziert werden, wird dieser von allen trainierten Decision Trees bewertet. Anschließend wird die Klasse des Datenpunktes durch eine Mehrheitsentscheidung der Decision Trees bestimmt. Daher der Namensteil *Forest*. Der Namensteil *Random* oder „Zufall" beruht darauf, das beim Trainieren der einzelnen Decision Trees nicht, wie zuvor beschrieben, eine optimale Aufteilung in jedem Entscheidungspunkt getroffen wird. Stattdessen werden leicht suboptimale, von Pseudozufallsgeneratoren bestimmte, Entscheidungen getroffen [23]. Dies kann ein Overfitting des Modells verhindern. Vor- und Nachteile des Random Forests sind in Tabelle 3.5 zusammengefasst.

Vorteile	Nachteile
Relativ resistent gegen Overfitting.	Aufgrund der Zufallskomponente haben Nutzer relativ wenig Einfluss darauf wie das Modell konkret arbeitet.
Geeignet um große Datensätze mit einer Vielzahl an Merkmalen zu klassifizieren.	Eine Hohe Anzahl von Decision Trees könnte den Algorithmus zu langsam für Echtzeitanwendungen machen.

Tabelle 3.5: Vor- und Nachteile des Random Forests [66]

3.4.2.3 k-Nearest-Neighbors

Der k-Nearest Neighbors, zu Deutsch „k-nächste-Nachbarn", ist in der vierten Spalte der Abbildung 3.21 zu sehen. Um einen neuen Datenpunkt zu klassifizieren die *k-nächsten*[2] Datenpunkte des Trainingsdaten betrachtet. Der neue Datenpunkt wird dann als die Klasse klassifiziert, welche dabei am häufigsten auftritt (s. Abb. 3.23). Der Algorithmus

[2] k kann dabei jede natürliche Zahl größer gleich Eins sein.

entscheidet also basiert auf der Ähnlichkeit zu Datenpunkten, welche er bereits kennt. [15] Vor- und Nachteile des k-Nearest Neighbors sind in Tabelle 3.6 zusammengefasst.

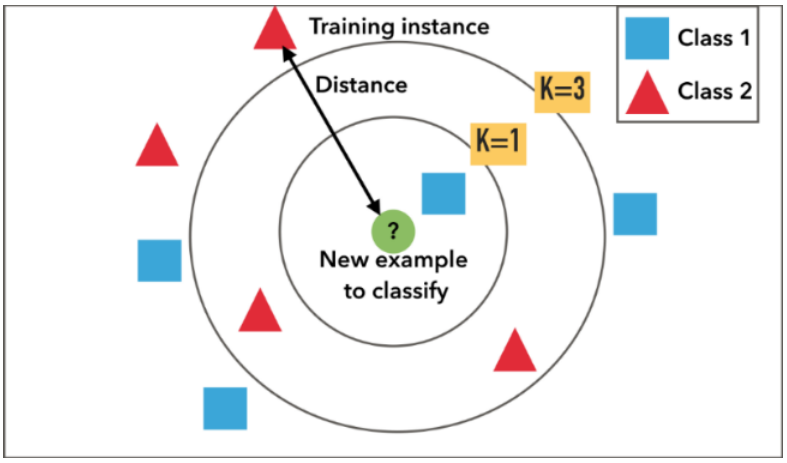

Abbildung 3.23: Funktionsweise eines k-NN Klassifikators. Der neue Datenpunkt wird für $k = 1$ als Klasse **1** erkannt und für $k = 3$ als Klasse **2** [15].

Vorteile	Nachteile
Keine Annahmen über die Daten (unter anderem nützlich bei nichtlinearen Daten).	Rechenaufwendig, da für die Klassifizierung eines neuen Datenpunktes alle bereits bekannten Datenpunkte überprüft werden müssen.
Unempfindlich gegenüber Ausreißern.	Viel Speicher notwendig, da alle bereits bekannten Datenpunkte gespeichert werden müssen.
Einfach zu verstehen.	Empfindlich gegenüber der Skalierung der einzelnen Merkmale. Normalisieren der Merkmale sinnvoll.

Tabelle 3.6: Vor- und Nachteile des k-Nearest Neighbors [66] [15]

3.4.2.4 Naive Bayes

Der Naive Bayes Klassifikator ist in der fünften Spalte der Abbildung 3.21 zu sehen. Er basiert auf dem Satz von Bayes und wird als *naive* bezeichnet, da er annimmt, dass alle Merkmale eines Datenpunktes gleich wichtig und statistisch unabhängig sind [66]. Mit Hilfe des Satzes von Bayes ist es möglich die bedingte Wahrscheinlichkeit $P(c|x)$ aus

den Wahrscheinlichkeiten $P(c)$, $P(x)$ und $P(x|c)$ auszurechnen Es müssen also folgende Wahrscheinlichkeiten bekannt sein: die Wahrscheinlichkeit, dass ein Datenpunkt einer Klasse angehört, die Wahrscheinlichkeit, dass ein Datenpunkt ein bestimmtes Merkmal hat und die Wahrscheinlichkeit, dass ein Datenpunkt einer bestimmten Klasse ein bestimmtes Merkmal hat. Nun können wir für neue Datenpunkte deren Merkmale wir kennen die Wahrscheinlichkeit berechnen, dass sie einer bestimmten Klasse angehören [37]. Die Wahrscheinlichkeit $P(c|x)$ errechnet sich als:

$$P(c|x) = \frac{P(x|c) \cdot P(c)}{P(x)}.$$

Da angenommen wird, dass die Merkmale statistisch unabhängig sind lässt sich die Wahrscheinlichkeit $P(x|c)$ auf die einzelnen Merkmale aufteilen:

$$P(x|c) = P(x_1|c) \cdot P(x_2|c) \cdot P(x_3|c) \cdot \ldots P(x_n|c).$$

Eine ausführlichere Beschreibung mit Beispielen ist unter [37] zu finden. Die Vor- und Nachteile des Naive Bayes Klassifikators sind in Tabelle 3.7 zusammengefasst.

Vorteile	Nachteile
Relativ hohe Genauigkeit.	Annahme von statistisch unabhängigen Merkmalen ist in echten Problemen so gut wie nie gegeben.
Gute Genauigkeit auch mit wenig Trainingsdaten.	Für gute Ergebnisse sollten kontinuierliche Merkmale gleichverteilt sein.
Gut für große Datensätze skalierbar.	Wenn ein kategorisches Merkmal in den Trainingsdaten eine Wahrscheinlichkeit von 0 hat, in den Testdaten aber nicht, ist eine Klassifizierung nicht möglich.

Tabelle 3.7: Vor- und Nachteile des Naive Bayes Klassifikators [66]

3.4.2.5 Support Vector Machine

Zwei Support Vector Machines (SVMs), zu Deutsch Stützvektor-Maschinen, sind in der sechsten und siebten Spalte der Abbildung 3.21 zu sehen. SVMs versuchen eine oder mehrere Hyperebenen zu finden, welche die Klassen eindeutig trennen und damit möglichst viel Abstand zu den Datenpunkten lassen. Die Datenpunkte welche am nächsten an der

Hyperebene liegen werden Stützvektoren genannt. In Abbildung 3.24 ist ein Beispiel mit zwei Klassen dargestellt. Die Hyperebene H_1 ist nicht geeignet, da sie die Klassen nicht eindeutig voneinander trennt. Die Hyperebene H_2 trennt die Klassen zwar voneinander, lässt aber nur wenig Abstand zu den Datenpunkten. H_3 ist optimal, da es die Klassen mit möglichst großem Abstand voneinander trennt. [66]

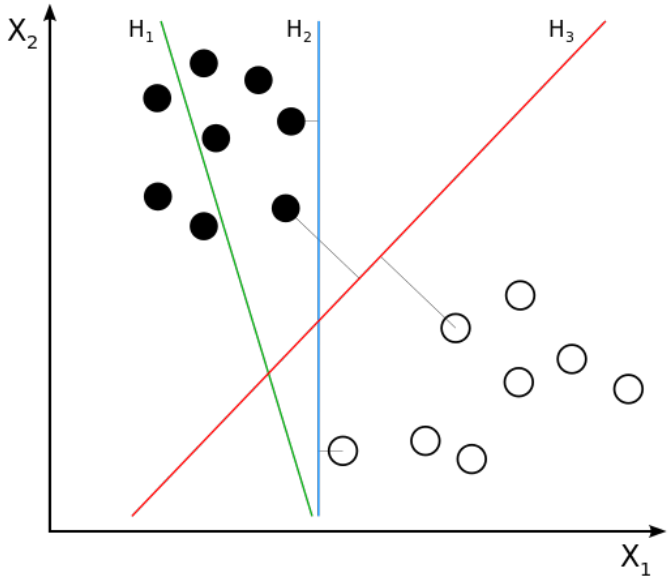

Abbildung 3.24: Beispiele für Hyperebenen einer SVMs. [66]

Lassen sich die Klassen nicht linear trennen (s. Abb. 3.25), bietet es sich an einen sogenannten Kernel-Trick zu verwenden. Dabei werden die Datenpunkte in einen höherdimensionalen Raum transformiert, in welchem sie linear trennbar sind. Eine der verbreitetsten Kernel-Funktionen ist die radiale Basisfunktion (RBF) [58]. In Abbildung 3.21 ist der Nutzen des Kernel-Tricks besonders gut in Zeile Zwei, Spalte Sechs und Sieben zu sehen. Während mit einer linearen SVM nur eine Genauigkeit von 40% erreicht werden konnte, erreicht die SVM mit radiale Basisfunktion (RBF)-Kernel eine Genauigkeit von 88%.

Die Vor- und Nachteile der SVM sind in Tabelle 3.8 zusammengefasst.

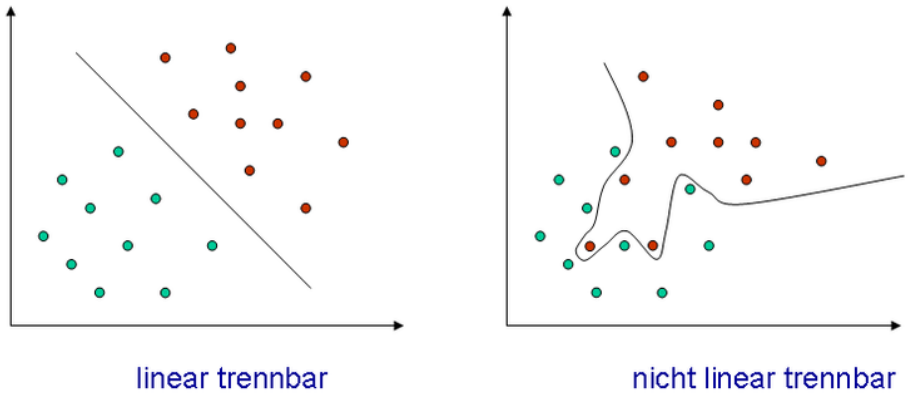

Abbildung 3.25: Linear trennbare und nicht linear trennbare Klassen [59]

Vorteile	Nachteile
Sehr effizient auch in höheren Dimensionen / bei Klassen mit vielen Merkmalen. Speichereffizient, da lediglich die Stützvektoren und einige Parameter gespeichert werden müssen.	Relativ lange Trainingszeit bei großen Datensätzen.

Tabelle 3.8: Vor- und Nachteile der Support Vector Machine [66]

4 Anforderungen

Im folgenden werden die Anforderungen an das System beschrieben, welche im weiteren Verlauf die Wahl der Hardware und das Design der Software beeinflusst haben. Zur Erinnerung: es soll ein Authentifizierungsverfahrens implementiert werden, welches das Eingabeverhalten eines Benutzers an einem Touchscreen auswertet. Ziel ist es, dass sich ein Angreifer, obwohl er das Passwort oder die Pin kennt, nicht anmelden kann.

4.1 Allgemeine Anforderungen

Aufgrund der zeitlichen Begrenzung der Arbeit sowie der Tatsache, dass das System lediglich als Proof-of-Concept gedacht ist, ergeben sich die folgenden allgemeinen Anforderungen:

1. Die Komponenten sollten so ausgewählt werden, dass die Entwicklungszeit möglichst gering ist.
2. Auch bei der Software ist eine kurze Entwicklungszeit wichtiger als eine möglichst einfache Skalierbarkeit.
3. Die Langlebigkeit der Komponenten ist nicht von außerordentlicher Wichtigkeit.
4. Da lediglich ein Exemplar des Systems geplant ist, spielt die Kostenoptimierung keine große Rolle.

4.2 Funktionale Anforderungen

Um eine Implementierung des psychometrischen Authentifizierungsverfahrens entsprechend testen zu können müssen folgende Funktionen gegeben sein:

1. Die Eingabe eines klassischen geheimen Wissens über einen Touchscreen.
    - Dies könnte unter anderem die Eingabe eines Passwortes, einer Pin oder eines Musters sein.
    - Es ist erstrebenswert bei der Eingabe nicht nur die Position der Berührung zu erfassen, sondern noch möglichst viele andere Daten. Anhand dieser könnten die Benutzer eventuell unterschieden werden. Dazu könnten zählen:
        - die Dauer jeder Berührung,
        - die Pause zwischen den Berührungen,
        - der Druck der Berührung,
        - die Form der Kontaktfläche.
2. Die Ausgabe der notwendigen Visualisierung über ein zum Touchscreen passendes Display. Diese beinhaltet:
    - Für den Fall einer Passworteingabe eine vollständige Tastatur.
    - Für den Fall einer Pineingabe ein Feld mit den Zahlen Null bis Neun.
    - Für den Fall einer Mustereingabe eine Begrenzung, innerhalb welcher das Muster eingegeben werden muss.
    - Eventuell weitere Informationen, wie eine Meldung über den Erfolg oder Misserfolg der Authentifizierung.
3. Das Abspeichern der Nutzereingaben, um das Anlernen des Systems beziehungsweise das Optimieren eines Authentifizierungsalgorithmus zu ermöglichen
4. Ein Authentifizierungsalgorithmus welcher alle verfügbaren Daten auswertet und entscheidet ob die Anmeldung erfolgreich war oder nicht. Also ob es sich um den rechtmäßigen Benutzer handelt oder um einen Angreifer.

4.3 Anforderungen an den Touchscreen

Es stellt sich die Frage nach der Art des Touchscreens (s. Kapitel 3.2) und der Größe. Im Vergleich ist ein resistives Touchscreen im 4-Draht-Anschluss für dieses System am geeignetsten, da es am einfachsten auszuwerten ist und damit die kürzeste Entwicklungszeit verspricht. Die Vorteile eines kapazitiven Touchscreens (Multi-Touch, längere Lebenszeit und höhere Transparenz) wären zwar sicherlich nicht hinderlich, können jedoch nicht den höheren Entwicklungsaufwand, die höheren Kosten und die schlechtere Auswahl im Einzelhandel aufwiegen. Die Größe des Touchscreens ist so zu wählen, dass die Eingabe eines Passwortes oder einer Pin problemlos möglich ist. Auf einen unnötig großen Touchscreen ist jedoch zu verzichten, da dies keinen Mehrwert bieten würde und die Linearität von resistiven Touchscreens mit zunehmender Größe immer schwieriger zu kontrollieren wird. Des Weiteren ist darauf zu achten, dass auch ein Display in der entsprechenden Größe erhältlich ist. Aufgrund dieser Anforderungen würde sich eine Größe von 7" anbieten.

4.4 Anforderungen an das Display

Betrachten wir die Anschlussmöglichkeiten an das Zedboard (a. Abb. 3.2), haben wir die Wahl zwischen einen High Definition Multimedia Interface (HDMI)- und einem Video Graphics Array (VGA)-Display. Alternativ könnte auch über die GPIOs ein 40-Pin Low Voltage Differential Signaling (LVDS) Display angeschlossen werden. Betrachtet man die Verfügbarkeit in der Größe 7" scheidet die VGA Variante aus. Die Entscheidung zwischen HDMI und 40-Pin-LVDS Display fällt aufgrund der Entwicklungszeit. Für das 40-Pin-LVDS-Display müsste ein Anschlussboard für den FPGA Mezzanine Card (FMC)-Anschluss [47] (s. Abb. 3.2) entwickelt werden. Daher eignet sich ein HDMI-Display am besten für dieses System.

4.5 Anforderungen an die Programmierplattform

Eine Überdimensionierung der Programmierplattform ist kein Problem, da das System lediglich als Proof-of-Concept gedacht ist. Aufgrund der vielseitigen Anschlussmöglichkeiten, der hohen Rechenleistung und der flexiblen Möglichkeiten der Programmierung wurde das Zedboard als Programmierplattform gewählt. Zu dieser Entscheidung beigetragen hat zudem die Tatsache, dass im Fachbereich bereits Erfahrungen mit dem

Zedboard gesammelt wurden und eine umfassende Dokumentation [11] sowie zahlreiche Tutorials [19] existieren.

4.6 Anforderungen an den Authentifizierungsalgorithmus

Für die Authentifizierung sollen verschiedene Klassifikatoren trainiert und bewertet werden. Das Trainieren soll nicht auf dem Zedboard sondern in einer Hochsprache auf einem PC erfolgen. Dadurch sollte es möglich sein in kürzerer Zeit verschiedene Klassifikatoren zu Testen, zu Bewerten und zu Vergleichen. Nach der Bewertung in Software soll der erfolgversprechendste Algorithmus auf dem Zedboard implementiert werden. Die Performance soll in einem Feldtest überprüft werden.

Weitere Fragen, welche sich zum Thema Authentifizierungsalgorithmus stellen sind:

1. Darf der Algorithmus beim Trainieren auf die Daten aller Nutzer zurückgreifen oder nur auf die des eigentlichen Benutzers? Handelt es sich also um eine Klassifikation oder um eine Novelty (deut. Neuheiten) und Outlier (deut. Ausreißer) Erkennung[1]?
2. Soll der Algorithmus nur zwischen **Nutzer** und **Angreifer** unterscheiden können oder soll er bei der Anmeldung vorhersagen können welche der Testpersonen die Eingabe durchgeführt hat. Handelt es sich also um ein Zwei- oder Mehrklassen-Problem?
3. Welche Art von geheimen Wissen soll abgefragt werden?
4. Wie lang soll das Passwort oder der Pin-Code sein? Wie viele Merkmale stehen dem Algorithmus zur Verfügung?
5. Sollen vorgegebene Pins/Passwörter eingegeben werden oder sollen sie von den Testpersonen selbst gewählt werden?

Da das Problem der Klassifizierung das einfachere ist, wird zunächst auf dieses der Fokus gelegt. Da man aber durchaus argumentieren könnte, dass in einem echten Szenario nicht immer Daten potenzieller Angreifer zur Verfügung stehen, sollte auch das Problem Novelty und Outlier Erkennung betrachtet werden.

[1] Die Novelty und Outlier Erkennung ist ebenso wie die Klassifizierung ein Problem des maschinellen Lernens. Hierbei sind allerdings nur Daten einer Klasse bekannt. Der Algorithmus soll für neue Datenpunkte entscheiden, ob sie dieser Klasse angehören oder nicht. Weiterführende Informationen sind unter [46] zu finden.

Bei einer Authentifizierung handelt es sich im Allgemeinen um ein Zwei-Klassen-Problem. Solange ein Angreifer nicht als berechtigter Nutzer erkannt wird, ist man in der Regel zufrieden. Sollte es sich bei dem Angreifer natürlich um einen bekannten unautorisierten Benutzer handeln, wäre es wünschenswert, wenn der Algorithmus erkennen könnte welcher Benutzer es genau war. Auf dieser Problemstellung liegt allerdings nicht der Fokus dieser Arbeit.

Bei der Frage Pin oder Passwort ist auch der Entwicklungsaufwand das ausschlaggebende Argument. Da es wesentlich leichter ist ein Feld mit 10 Zahlen, als eine Tastatur mit 104 Tasten zu implementieren, fällt die Wahl auf den Pin-Code.

Bezüglich der Länge der Eingabe bietet es sich an, verschiedene Längen zu überprüfen. Hierdurch kann auch abgeschätzt werden wie viele Merkmale notwendig sind, um eine hinreichend gute Klassifizierung durchführen zu können.

Um möglichst viele Eingaben für jeden der Pins zu erheben, bietet es sich an möglichst wenig verschiedene zu verwenden. Drei Pins mit jeweils unterschiedlicher Länge lassen eine Beurteilung der Wichtigkeit der Länge zu und maximieren die Eingaben pro Person pro Pin. Damit jede Testperson die gleichen Startbedingungen hat, ist es sinnvoll die Pincodes vorzugeben.

4.7 Anforderungen an die Bedienung

Die Bedienung muss nicht zwangsweise selbsterklärend sein, da während der Feldtest immer eine Einweisung vom Entwickler durchgeführt werden kann. Es sollte die Möglichkeit bestehen zwischen den drei Pincodes umzuschalten. Des weiteren bietet es sich an den Benutzer, welcher gerade die Eingabe durchführt, einstellen zu können. So könne diese Informationen während der Anlernphase mit abgespeichert werden und muss nicht manuell gepflegt werden.

4.8 Anforderung an die Datenspeicherung

Die Daten müssen schnell und zuverlässig abgespeichert werden. Zudem muss das Speichermedium auch einfach von einem PC ausgelesen werden können. Dies ist notwendig, da die Nutzerdaten für das Trainieren und Bewerten der Klassifikatoren vom Zedboard auf

einen PC übertragen werden müssen. Bei diesen Anforderungen und den Möglichkeiten des Zedboard bietet sich eine Secure Digital (SD)-Karte an.

5 Konzept

Bevor im nächsten Kapitel die genaue Implementierung beschrieben wird, soll an dieser Stelle das grundsätzliche Konzept vorgestellt werden. Hierfür wird in einem ersten Schritt der Aufbau des Systems beschrieben, bevor in einem zweiten Schritt das Konzept für die Testphasen beschrieben wird.

5.1 Systemaufbau

Der grundsätzliche Aufbau des Systems ist in Abbildung 5.1 zu sehen. Die zentrale Komponente bildet das Zedboard. Auf dem Zedboard befindet sich der Zynq Chip, welcher wiederum aus der Programmable Logic (PL) und dem Processing System (PS) besteht. Der Austausch von Daten zwischen PL und PS erfolgt über ein AXI-Interface. Die auf dem Zedboard befindlichen Taster, LEDs und Schalter, sowie das OLED-Display sind direkt mit der PL-Seite des Zynqs verbunden. Auch das externe HDMI Display und der Touchscreen sind über den HDMI-Anschluss respektive den PMOD-Anschluss mit der PL-Seite des Zynqs verbunden. Wie zu sehen, ist der Touchscreen nicht direkt mit dem Zedboard verbunden, sondern über ein Anschlussboard. Die SD-Karte des Zedboards ist direkt mit der PS-Seite des Zynqs verbunden. Schlussendlich ist noch der PC mit aufgeführt, welcher auch in der Lage sein muss die SD-Karte zu lesen.

5.1.1 Touchscreen

Bei dem verwendeten Touchscreen handelt es sich um ein resistives Touchscreen im 4-Draht-Anschluss in der Größe 7" (s. Abb. 5.2). Die Anschlüsse der vier Elektroden des Touchscreen sind mit einem Flexible Printed Circuit (FPC)-Kabel nach außen geführt. Um dieses an das Zedboard anzuschließen ist ein Anschlussboard nötig.

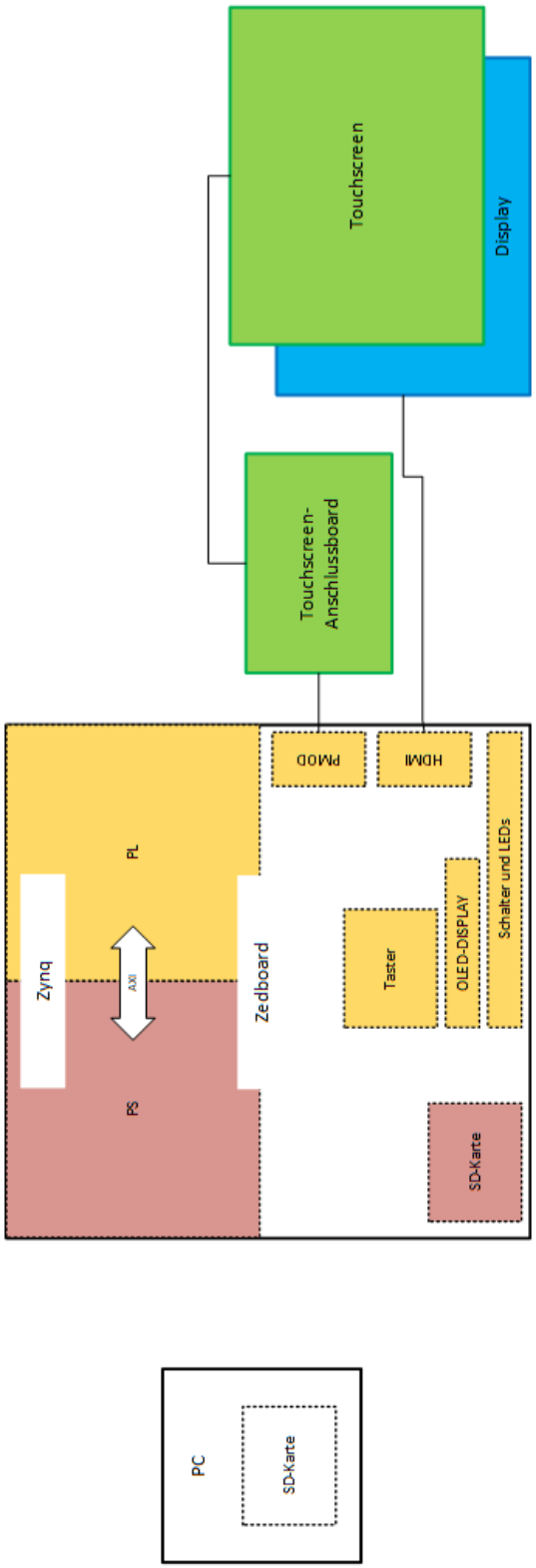

Abbildung 5.1: Grundsätzlicher Aufbau des Systems mit Zedboard, Touchscreen, Display und PC

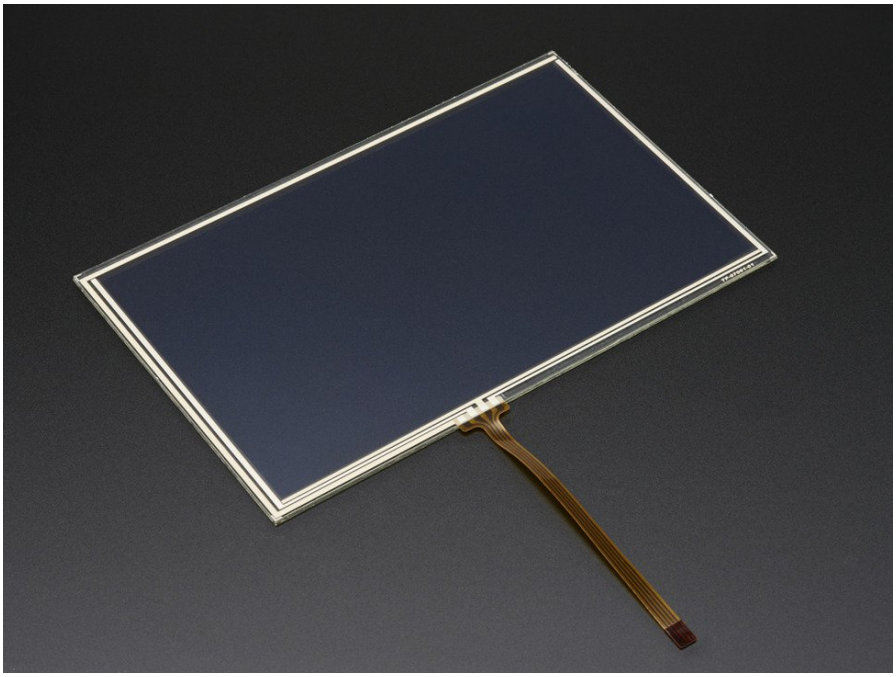

Abbildung 5.2: Verwendetes resistives Touchoverlay, 7", 4-Draht-Anschluss [2]

Auf diesem **Anschlussboard** befindet sich außerdem der Touchscreen-Treiber-IC. Das Anschlussboard ist über einen PMOD-Stecker mit dem Zedboard verbunden. Die Kommunikation zwischen dem Zedboard und dem Treiber-IC erfolgt über SPI. Der Treiber-IC soll erkennen, wenn der Touchscreen berührt wird und dies an das Zedboard melden. Dieses soll dann das Auslesen der Koordinaten dieser Berührung veranlassen. Im Anschluss sollen die Messwerte, falls es sich als notwendig herausstellt, noch bereinigt werden. Danach können die Messwerte visualisiert und zur Authentifizierung verwendet werden.

Die Kommunikation soll dabei mit dem Treiber-IC über die PL laufen. Diese wiederum stellt die Messwerte dem PS zur Verfügung. Dort findet die weitere Verarbeitung der Daten statt.

5.1.2 Display

Das Display (s. Abb. 5.3) ist über HDMI direkt mit dem Zedboard verbunden. Es hat eine zum Touchscreen passende Größe von 7" und eine Auflösung von 800×480 Pixel. Auf dem Display sollen folgende Informationen und Elemente dargestellt werden:

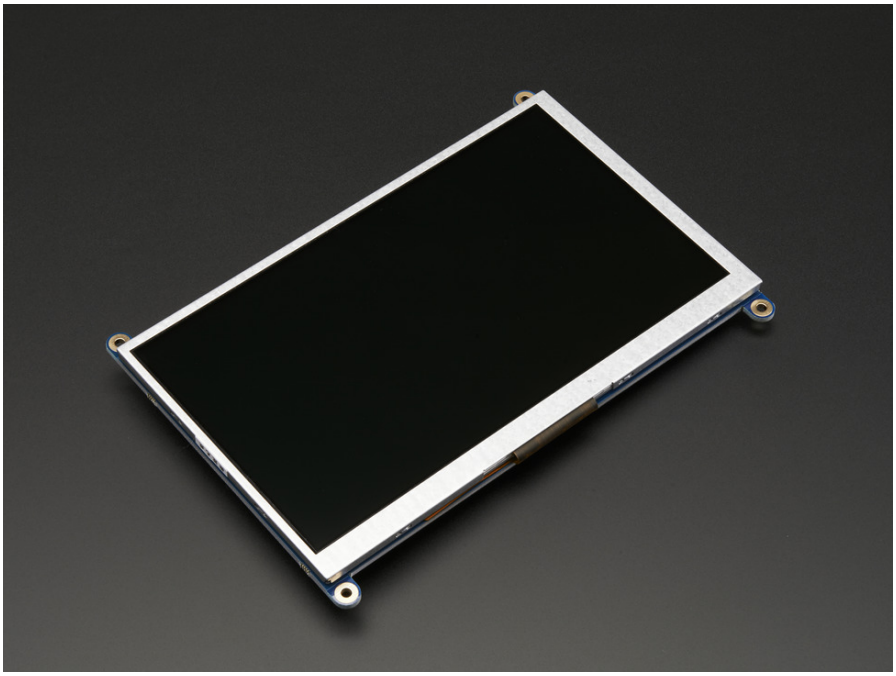

Abbildung 5.3: Verwendetes HDMI Display, 7", Auflösung: 800 × 480 Pixel [1]

- das Pinfeld mit den Zahlen Null bis Neun und je einem Feld für das Bestätigen und das Abbrechen der Eingabe,
- einer Anzeige an welcher Stelle die letzte Berührung des darüber liegenden Touchscreens gemessen wurde,
- welcher Nutzer gerade ausgewählt ist,
- welcher Account gerade ausgewählt ist,
- die zum Account gehörige Pin,
- eine Anzeige die angibt wie viele Stellen der Pin bereits eingegeben wurden,
- die Information ob die Authentifizierung erfolgreich war oder nicht.

In Abbildung 5.4 ist das geplante Layout für das Display zu sehen. Unter dem Schriftzug *ACC* soll angezeigt werden, welcher Account gerade ausgewählt ist. Also Null, Eins oder Zwei, je nachdem welche der vorgegebenen Pins der Benutzer gerade eingeben soll. Unter dem Schriftzug *USR* (User) soll angezeigt werden, welcher Benutzer gerade ausgewählt

ist. Hierfür wird jeder Testperson eine Nummer zugeordnet. Dies soll im späteren Verlauf die Pflege der Daten erleichtern.

In der oberen linken Ecke ist die Zahlenfolge *12345678* zu sehen. Diese ist nur exemplarisch. Im späteren Verlauf soll hier eine der Drei vorgegebenen Pins angezeigt werden. Die Quadrate unter dieser Zahlenfolge zeigen an, wie viele Stellen der Pin bereits eingegeben wurden. Zu Beginn der Eingabe ist keins der Quadrate sichtbar. Im Laufe der Eingabe erscheinen von links nach rechts nacheinander schwarze Quadrate. Dies ist gedacht um dem Benutzer die Bedienung zu erleichtern. So kann beispielsweise auch festgestellt werden, ob eine Berührung des Touchscreens auch wirklich erkannt wurde, oder ob aus Versehen mehrere Eingaben durchgeführt wurden. Nach dem Prüfen der Eingabe durch den Authentifizierungsalgorithmus werden diese Quadrate entweder grün, wenn die Authentifizierung erfolgreich, war oder rot, wenn die Eingabe abgelehnt wurde.

Die Anzeige der letzten Berührung soll durch ein blaues Quadrat an der entsprechenden Stelle angezeigt werden. Im Layout ist dies exemplarisch für eine Berührung des oberen linken Bereiches des Feldes der Zwei dargestellt.

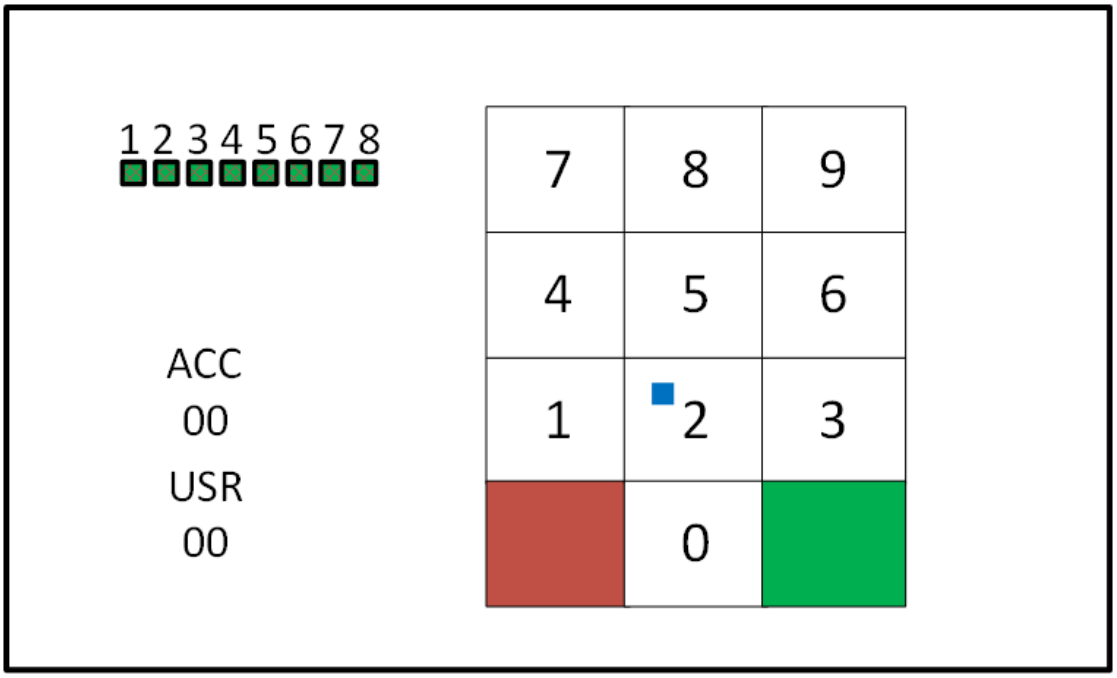

Abbildung 5.4: Grober Entwurf des Display-Layouts

Die Bild- und Signal-generierung soll in der PL-Seite des Zynqs stattfinden. Die Steuerung des Inhaltes, also welche Pin, welcher Nutzer- und Accountnummer angezeigt werden soll,

erfolgt über das Anwenderprogramm im PS. Auch die Position für die letzte Berührung soll im PS berechnet werden. Dies ist notwendig da Touchscreen und Display nicht perfekt aufeinander liegen. Zudem kann es durch nicht lineare Verläufe im Touchscreen zu Verzerrungen kommen. In Abbildung 5.5 ist dies exemplarisch visualisiert. Dargestellt ist kein reelles System sondern lediglich eine zur Verdeutlichung gedachte Grafik. Sie ist wie folgt zu interpretieren: Auf einem Display wird der blaue Kreis angezeigt. Auf diesem Display befindet sich ein Touchscreen. Nun wird der blaue Kreis, welcher durch den Touchscreen hindurch sichtbar ist, auf eben diesem nachgezeichnet. Bei einem idealem Touchscreen würde man erwarten, dass die Messwerte auch wieder einen perfekten Kreis ergeben. Dadurch das Touchscreens in der Regel aber nicht ideal sind und auch nicht perfekt auf dem Display platziert sind, kommt es zu Offsets, Drehungen und Skalierungen. Die Messwerte stimmen nicht mit der Ausgabe auf dem Display überein. Dies bedeutet in Folge auch, dass zur Anzeige der Position einer Berührung, diese erst aus den Messdaten ausgerechnet werden muss.

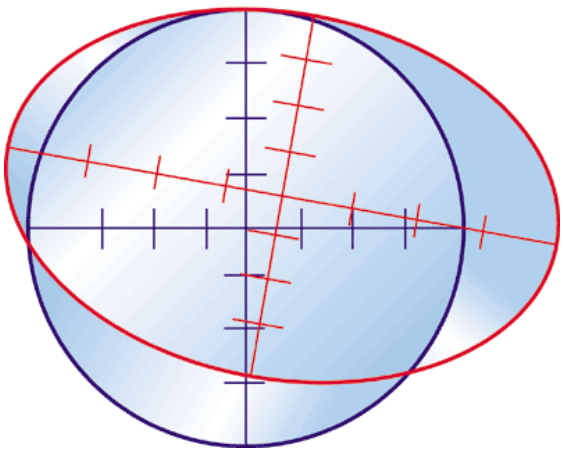

Abbildung 5.5: Ein Kreis auf einem Display und die dazu äquivalenten Messwerte eines Touchscreens [52]

5.1.3 Menüführung und Bedienung

Die Menüführung soll in dem PS des Zynqs stattfinden und über die Taster sowie das OLED-Display des Zedboards bedient werden. Es sollen folgende Funktionen implementiert werden:

- Auswahl des Benutzers,

- Auswahl des Accounts,
- Starten der Benutzereingabe,
- Abbruch der Benutzereingabe (während der Eingabe und nach Beendigung),
- Speichern der Messwerte nach Beendigung der Pineingabe.

5.1.4 Speicherung der Messwerte

Die Speicherung der Messwerte erfolgt über das PS. Nach Abschluss der Pineingabe und Bestätigung über die Taster werden die Messwerte der SD-Karte gespeichert. Anschließend können diese manuell auf den PC übertragen werden Auf diesem findet die weitere Auswertung statt.

Es sollen folgende Werte abgespeichert werden:

- Der Tag an welchem die Eingabe stattfand,
- Der Nutzer welcher die Eingabe durchgeführt hat,
- Der Account für welchen die Eingabe durchgeführt wurde
- Die Länge der Pin welche eingegeben wurde[1],
- Die Messwerte für die Vier bis Acht Berührungen, mit denen die Pin eingegeben wurde. Dabei sollen die folgenden fünf Werte erfasst werden:
  - X-Wert der Touch-Messung. Dieser entspricht der horizontale Position der Berührung.
  - Y-Wert der Touch-Messung. Dieser entspricht der vertikalen Position der Berührung
  - Z-Wert der Touch-Messung. Dieser entspricht der Kontaktfläche der beiden Touchscreenlagen während der Berührung.
  - Dauer der Berührung.

[1] Dies ist zugegebenermaßen redundant, da diese Information bereits in der Nummer des Accounts steckt. Aus Gründen der Bequemlichkeit wurde entschieden diese Information dennoch mit abzuspeichern.

- Pause vor der Berührung. Also die Zeit zwischen dem Ende der vorangegangenen und dem Beginn der folgenden Berührung.

5.1.5 Authentifizierungsalgorithmus

Die Authentifizierung soll von einem Klassifikator durchgeführt werden. Hierfür sollen verschiedene Modelle auf dem PC trainiert und bewertet werden. Nach einem Vergleich der Klassifikatoren soll der erfolgversprechendste auf dem PS des Zynqs implementiert werden.

5.2 Vorgehen

Aufgrund der Thmatik macht es Sinn verschiedene Phasen für diese Arbeit zu definieren. dies sind:

1. Nutzerdatenerhebung
2. Training und Simulation der Klassifikatoren
3. Validieren der Klassifikatoren im Feldtest

5.2.1 Nutzerdatenerhebung

Während dieser Phase sollen möglichst viele Nutzer die Pins **6649**, **283764** und **15948755** so oft wie möglich eingeben. Die Messwerte dieser Eingaben sollen wie zuvor beschrieben gespeichert werden und auf einen PC übertragen werden. Die Pins wurden mehr oder weniger willkürlich gewählt. Es wurde jedoch darauf geachtet möglichst alle Entfernungen zu erfassen. So kommt die Eingabe von zwei gleichen Zahlen hintereinander vor, aber ebenso die Eingabe von zwei gegenüberliegenden Zahlen (beispielsweise **37**).

5.2.2 Training und Simulation der Klassifikatoren

Während der zweiten Phase sollen mit den in Phase 1 erfassten Daten verschiedene Klassifikatoren getestet werden. Hierfür sollen die Daten in Trainings- und Testdaten aufgeteilt werden. Die Modelle werden auf die Trainingsdaten optimiert. Anschließend werden sie auf die Testdaten angewandt.

Die erste Variante welche geprüft werden soll, ist die Zwei-Klassen-Klassifizierung. Hierbei muss für jede Person und jeden Account ein eigener Klassifikator trainiert werden. In diesem Fall soll unterscheiden werden, ob die Eingabedaten von dem autorisierten Benutzer stammen oder nicht. Für einen Klassifikator welcher auf den Benutzer 1 und Account 1 trainiert wird, sind beispielsweise nur die Eingabedaten aller Nutzer für den Account 1 relevant. Des Weiteren würden alle Eingaben welche nicht von Benutzer 1 stammen in einer Klasse zusammengefasst werden. Dieser Klassifikator würde dann nur unterscheiden zwischen **Benutzer 1** und **nicht Benutzer 1**.

5.2.3 Validieren der Klassifikatoren im Feldtest

In einer letzten Phase soll mindestens einer der in Phase 2 trainierten und simulierten Klassifikatoren auf dem System implementiert werden. In einem Feldtest sollen Testpersonen den Pin, auf welchen der Klassifikator trainiert wurde, eingeben. Im Falle eines Zwei-Klassen-Klassifikators entscheidet dieser dann, ob die Eingabe vom autorisierten Benutzer oder einem Angreifer stammt. In dieser Phase ist es auch erstrebenswert möglichst viele Testpersonen teilnehmen zu lassen.

Vor den Versuchen werden die Testpersonen darauf hingewiesen, wie dieses Authentifizierungssystem funktioniert. Es wird ihnen Erklärt, dass auf die Position, Fläche und Dauer der Eingaben, sowie auf die Pausen zwischen den Eingaben geachtet wird. Des Weiteren wird Ihnen mitgeteilt in welche Pin sie eingeben müssen. Zusätzlich wird ihnen aber erläutert auf welche Person der Klassifikator trainiert ist.

Nach den ersten 10 Versuchen haben die Testpersonen zusätzlich die Möglichkeit den Benutzer beliebig oft bei der Eingabe der Pin zu beobachten. Diese Möglichkeit wird ihnen gewährt, da es auch in echten Anwendungsfällen durchaus denkbar wäre, dass ein Angreifer den Benutzer bei der Eingabe der Pin beobachtet. Es wäre sogar denkbar, dass er den Benutzer dabei filmt. In diesem Fall hätte er die Möglichkeit sich die Eingabe beliebig oft anzuschauen.

Während des Test soll erfasst werden wie oft es der Benutzer und jeder Angreifer geschafft hat sich anzumelden und bei welchem Versuch der erste Erfolg verzeichnet wurde.

6 Implementierung

In diesem Kapitel wird auf die genaue Umsetzung des in Kapitel 5 vorgestellten Konzepts eingegangen. Dabei wird zunächst auf das Design des Anschlussboards (s. Abb. 5.1) eingegangen. Anschließend werden die in der Programmable Logic (PL) implementierten Komponenten erklärt bevor die Programmierung des Processing System (PS) im Detail beschrieben wird. Aus Gründen der Übersichtlichkeit wird die Auswertung der Nutzerdaten und der anschließende Vergleich und Test der Klassifikatoren sowie die Kalibrierung des Touchscreens in einem seperaten Unterkapitel behandelt.

6.1 Touchscreen-Anschlussboard

Die Aufgabe des Anschlussboards ist es den Touchscreen zu überwachen und eine Berührung an die PL zu melden. Nach Aufforderung durch die PL soll das Anschlussboard die Position der Berührung messen, digitalisieren und an die PL schicken. Herzstück des Anschlussboards ist der Touchscreen-Controller *TSC2046E* von Texas Instruments.

6.1.1 Grundlagen zum Touch-Controller TSC2046E

Der IC ist in verschiedenen Packungen verfügbar. Um ein einfaches Verbauen auf der Platine zu gewährleisten wurde die Thin Shrink Small Outline Package (TTSOP) Packung (s. Abb. 6.1) verwendet.

Der *TSC2046E* verfügt über 16 Pins. Eine Übersicht über deren Funktionen ist in Tabelle 6.1 gegeben.

In Abbildung 6.2 ist der interne Aufbau des Chips dargestellt. Die vier Anschlüsse des Touchscreens (linke Seite) sind über einen Multiplexer auf den 12 bit Successive Approximation Register (SAR) Analog to Digital Converter (ADC) geführt. Die Messwerte werden über das serielle Interface (rechte Seite) an einen Master übertragen. Dies ist in

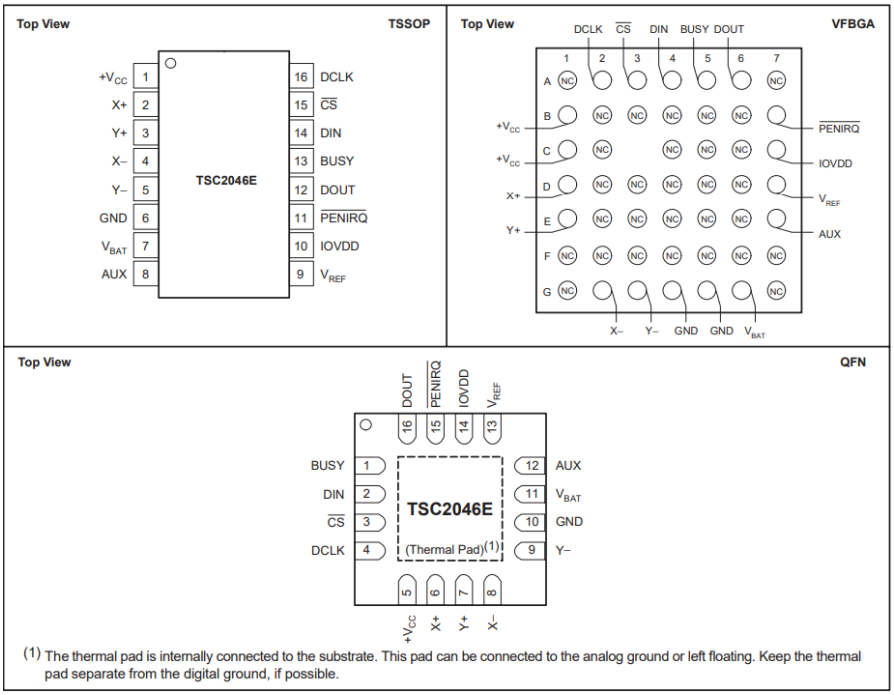

Abbildung 6.1: Verfügbare Packungen des Touch-Controllers *TSC2046E* [49]

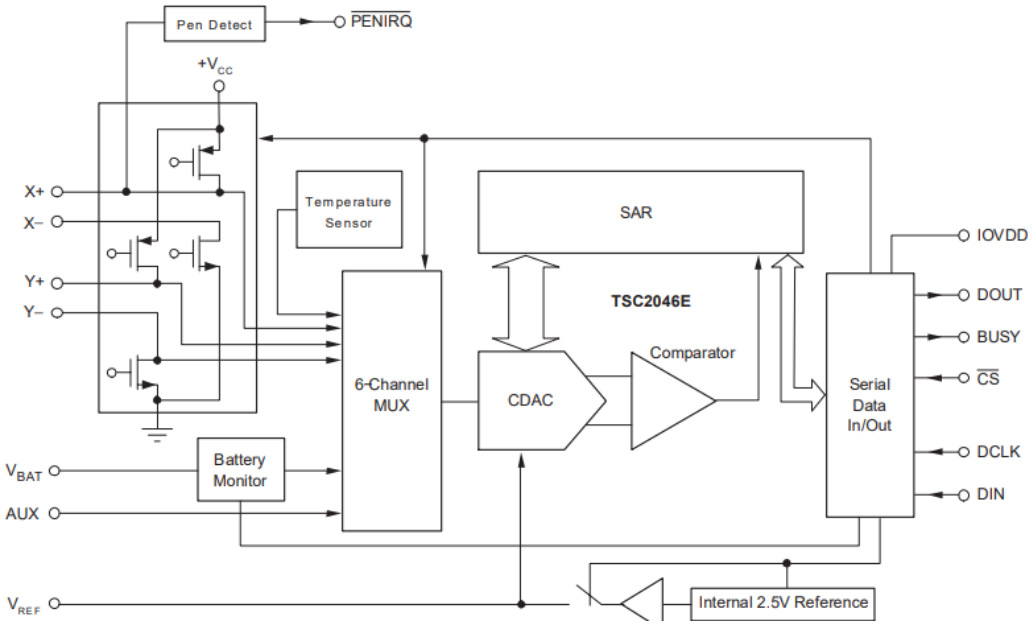

Abbildung 6.2: Aufbau des Touch-Controllers *TSC2046E* [49]

Pin-Nummer	Pin-Nummer	Funktion
1	$+V_{cc}$	Versorgungsspannung (2.7 V . . . 3.6 V)
2	X+	X+ Anschluss des Touchscreens
3	Y+	Y+ Anschluss des Touchscreens
4	X-	X- Anschluss des Touchscreens
5	Y-	Y- Anschluss des Touchscreens
6	GND	Bezugspotential/Masse
7	V_{BAT}	Messeingang für Batteriespannung (nicht verwendet)
8	AUX	Hilfseingang auf einen Analog to Digital Converter (ADC) (nicht verwendet)
9	V_{REF}	Referenzspannung (nicht verwendet)
10	IOVDD	Digitale Versorgungsspannung
11	$\overline{\text{PENIRQ}}$	„Pen Interrupt“, Berührungserkennung. Wechselt bei Berührung des Touchscreens auf Low-Pegel
12	DOUT	Serieller Datenausgang. Im folgenden auch Master in - Slave out (MISO) genannt. Der Datenwechsel erfolgt bei fallender Flanke des DCLK-Signals. Hochohmig, wenn an $\overline{\text{CS}}$ ein High-Pegel anliegt.
13	BUSY	Zeigt an, ob der IC gerade beschäftigt ist. Hochohmig, wenn an $\overline{\text{CS}}$ ein High-Pegel anliegt.
14	DIN	Serieller Dateneingang. Im folgenden auch Master out - Slave in (MOSI) genannt. Die Daten werden bei fallender Flanke von DCLK eingelesen, wenn an $\overline{\text{CS}}$ ein Low-Pegel anliegt.
15	$\overline{\text{CS}}$	Chip-Select Signal (s. Kapitel 3.3.1). Bei High-Pegel schaltet sich zusätzlich der ADC ab.
16	DCLK	Externes Takt-Signal. Der Takt wird zur Datenübertragung und für das Successive Approximation Register (SAR) verwendet.

Tabelle 6.1: Funktionsbeschreibung der Pins des Touch-Controllers [49]

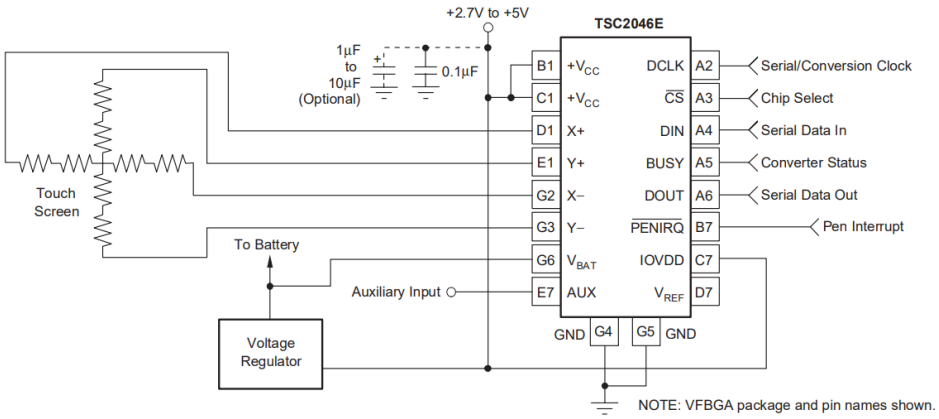

Abbildung 6.3: Grundschaltung des Touch-Controllers *TSC2046E* [49]

diesem Fall die PL. Um nicht durchgehend Messungen durchführen zu müssen, obwohl keine Berührung stattfindet, kann der $\overline{\text{PENIRQ}}$-Anschluss verwendet werden. Hierfür wird im entsprechenden Modus die Y-Lage des Touchscreens mit GND verbunden. Der $\overline{\text{PENIRQ}}$ ist mit dem X+ Eingang verbunden und wird über einen Pullup-Widerstand auf $+V_{cc}$-Pegel gezogen. Wird der Touchscreen nun berührt, entsteht eine leitende Verbindung zwischen den zwei Lagen (s. Kapitel 3.2.1). Dadurch wird der $\overline{\text{PENIRQ}}$-Pin auf Low gezogen und die Berührung kann von der PL erkannt werden. Im Anschluss kann das Messen der Koordinaten vom Master veranlasst werden.

Des Weiteren verfügt der IC über einen Temperatursensor und die Möglichkeit eine Batterie zu überwachen. Beide Funktionen werden nicht verwendet, da die Spannungsversorgung direkt über das Zedboard erfolgt.

6.1.2 Schaltungsentwurf und Platinenlayout

Auf Grundlage der im Handbuch [49] enthaltenden Grundschaltung (s. Abb. 6.3) wurde der Schaltplan (s. Abb. 6.4) und anschließend das Platinenlayout (s. Abb. 6.5) entworfen. Die Platine ist zweilagig. Die obere Lage ist in Rot, die untere in Blau dargestellt. Grün entspricht einer Verbindung der beiden Lagen.

Die Pins V_{BAT}, V_{REF} und AUX wurden auf Masse gelegt, da sie im weiteren nicht verwendet werden. Die Verwendung einer externen Referenzspannung ist beim Messen mit differentieller Referenz nicht notwendig. In diesem Fall wird die Spannung, welche über

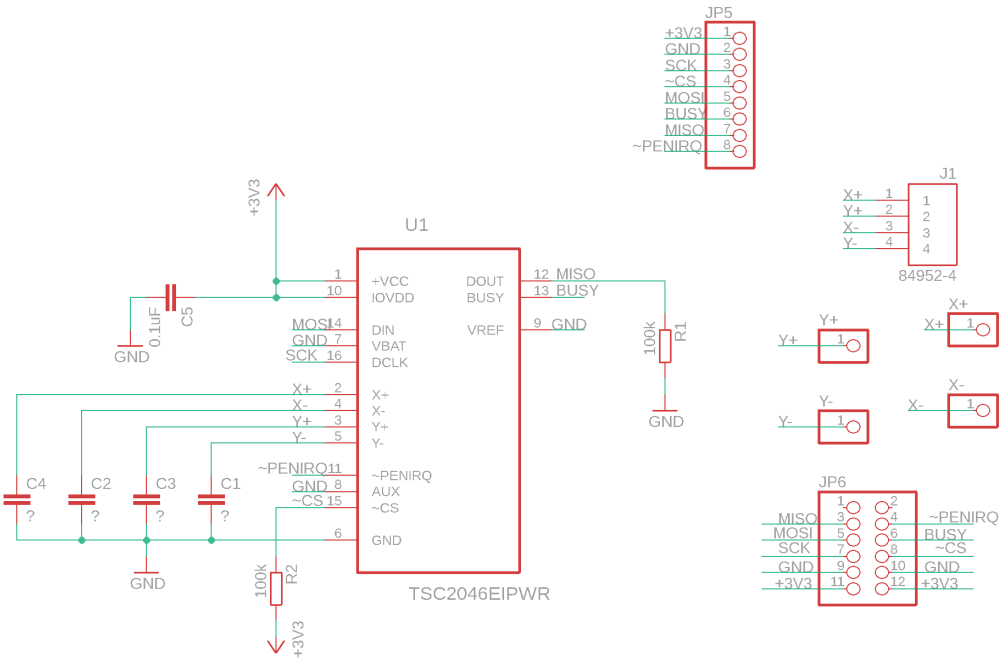

Abbildung 6.4: Schaltplan für Touchscreen-Anschlussboard

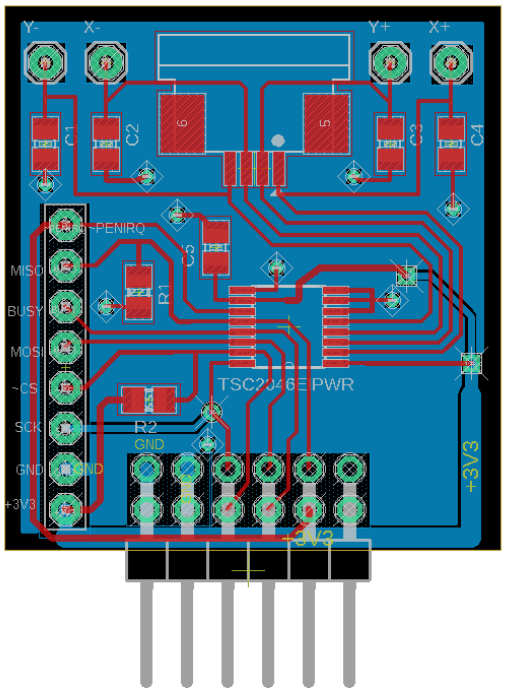

Abbildung 6.5: Platinenlayout für das Touchscreen-Anschlussboard. Obere Lage in Rot, untere in Blau. Verbindungen der Lagen in Grün.

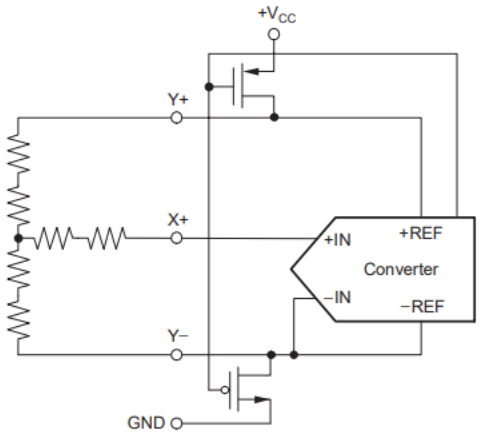

Abbildung 6.6: Vereinfachte Darstellung der Messschaltung mit differentieller Referenz [49]

den Touchscreen abfällt als Referenz verwendet (s. Abb. 6.6). Die Messung mit differentieller Referenz wird für Messungen der Touch-Koordinaten vom Hersteller empfohlen An dieser Stelle sei auch erwähnt, dass der IC über die Möglichkeit der Messung mit Single-Ended Referenz verfügt. Da diese nur zum Messen der Batteriespannung, Temperatur und der Hilfsspannung empfohlen wird, findet sie im weiteren keine Verwendung.

Die Anschlüsse des Touchscreens (+X, -X, +Y und -Y) werden über den Flexible Printed Circuit (FPC)-Anschluss *J1* (s. Abb. 6.4 und Abb. 6.5 oben, mittig) mit dem IC verbunden. Zusätzlich werden die Anschlüsse mit Steckverbindern (s. Abb. 6.5 oben, links und rechts) verbunden um eine mögliche spätere Fehlersuche zu erleichtern. Im Handbuch wird erwähnt, dass es notwendig sein kann zusätzliche Kondensatoren über die Anschlüsse des Touchscreen anzuschließen. Damit ließen sich etwaige Störungen durch dahinter liegende Displays herausfiltern. Gleichzeitig würde dadurch aber die Reaktionszeit der Schaltung verschlechtert. Da im Handbuch keine genauen Werte für diese Kondensatoren angegeben sind wurden lediglich vier Plätze für diese Kondensatoren eingeplant. Im Schaltplan (s. Abb. 6.4) sind dies C1 bis C4. Da es sich als nicht notwendig herausstelle wurden diese im weiteren Verlauf nicht bestückt.

Die Versorgungsspannung, Masse und die digitalen Signale wurden an den PMOD-Header JP6 und die Stiftleiste JP5 (s. Abb. 6.4 und Abb. 6.5 unten beziehungsweise links) angeschlossen. Der PMOD-Header wird für den Anschluss an das Zedboard verwendet, die Stiftleiste zur Fehlersuche und Funktionskontrolle.

Der Pin MISO ist zusätzlich über den Pulldown-Widerstand $R1 = 100\,\mathrm{k\Omega}$ auf Masse

gezogenen, der Pin $\overline{\text{CS}}$ über den Pullup-Widerstand $R2 = 100\,\text{k}\Omega$ auf $+V_{cc} = 3.3\,\text{V}$.

Die Versorgungsspannung von 3.3 V wird, wie bereits erwähnt, vom Zedboard zur Verfügung gestellt und über den Kondensator $C5 = 0.1\,\mu\text{F}$ geglättet. Die Bezugsmasse stammt ebenfalls vom Zedboard und wird über die untere Lage der Platine verteilt.

6.1.3 Digitales Interface des Touchscreen-Contollers TSC2046E

Die Kommunikation zwischen dem Touchscreen-Controller-IC und dem Zedboard erfolgt über ein SPI. Dabei ist das Zedboard der Master und der TSC2046E der Slave. Der Ablauf dieser Kommunikation ist in Abbildung 6.7 dargestellt.

Die Kommunikation wird immer vom Master initiiert. Hierfür wird $\overline{CS}$ auf Low gesetzt und das Taktsignal DCLK erzeugt. Anschließend wird während der ersten 8 Taktzyklen vom Master ein Kontroll-Byte über DIN (MOSI) an den IC geschickt. Nach den ersten 5 Bit hat der IC bereits genug Informationen um die Anschlüsse des Touchscreens entsprechend zu beschalten und den Multiplexer einzustellen. Nachdem die letzten 3 Bit des Kontroll-Bytes übertragen sind führt der IC die Messung durch und schickt das Ergebnis über DOUT (MISO) an das Zedboard. Da es sich um einen 12 Bit ADC handelt werden hierfür 12 Taktzyklen benötigt. Um das Byte aufzufüllen werden anschließend vier Nullen gesendet.

Ein Kommunikationszyklus dauert also 24 Taktzyklen. Es ist möglich den Kommunikationszyklus auf 16 Taktzyklen zu verkürzen, indem bereits nach 16 Taktzyklen das nächste Kontroll-Byte vom Master gesendet wird. Hierauf wurde jedoch verzichtet.

Das Kontroll-Byte besteht aus 8 Bit. Dies sind vom Most Significant Bit (MSB) zum Least Significant Bit (LSB):

- Das **Startbit** S:
  - Dieses ist immer High und signalisiert den Beginn des Kontroll-Bytes.
- Die **Adressbits** A2, A1 und A0:
  - Diese Bestimmen zusammen mit dem $SER/\overline{DFR}$ Bit die Einstellungen für den Multiplexer und die Treiber der Touchscreen-Anschlüsse. Beim Messen mit differentieller Referenz sind folgende Kombinationen Relevant:
    - **101**: Messen der X-Position,

    - **001**: Messen der Y-Position,
    - **011** oder **100**: Messen der Z-Position. Diese sind proportional zur Kontaktfläche der Touchscreen-Lagen und damit zum Druck der Berührung[1].

- Das **Modebit** M:
  - Ist dies Low wird die Messung mit einer Auflösung von 12 Bit durchgeführt, andernfalls mit einer Auflösung von 8 Bit. Im folgenden werden lediglich 12-Bit-Messungen durchgeführt.
- Das $SER/\overline{DFR}$ Bit:
  - Dieses bestimmt ob die Referenzspannung des ADCs Single-Ended (High) oder differentiell (Low) ist. Im folgenden wird nur die Messung mit differentieller Referenz verwendet. Dieses Bit ist also immer Low.
- Die **Power-Down-Bits** PD1 und PD0:
  - Sie bestimmen über das Verhalten des ICs zwischen Messungen. Im folgenden werden sie beide immer Low gesetzt. In diesem Modus ist die Pen-Interrupt-Funktion aktiviert. Zwischen den Messungen wird der IC in einen Energiesparmodus versetzt.

Um, wie in Kapitel 5.1 beschrieben, die Werte zu messen, werden folgende Kontroll-Bytes verwendet:

- X-Messung: 1**10**10000,
- Y-Messung: 1**00**10000,
- Z-Messung: 1**01**10000.

Wie zu erkennen unterscheiden sich die Kontroll-Bytes nur in den hervorgehobenen Adressbits.

[1] Der genaue Zusammenhang zwischen dem Druck der Berührung und den Z-Messungen kann dem Handbuch [49] entnommen werden. Während der Durchführung dieser Arbeit wurden die Werte der Z-Messung direkt zur Nutzererkennung verwendet.

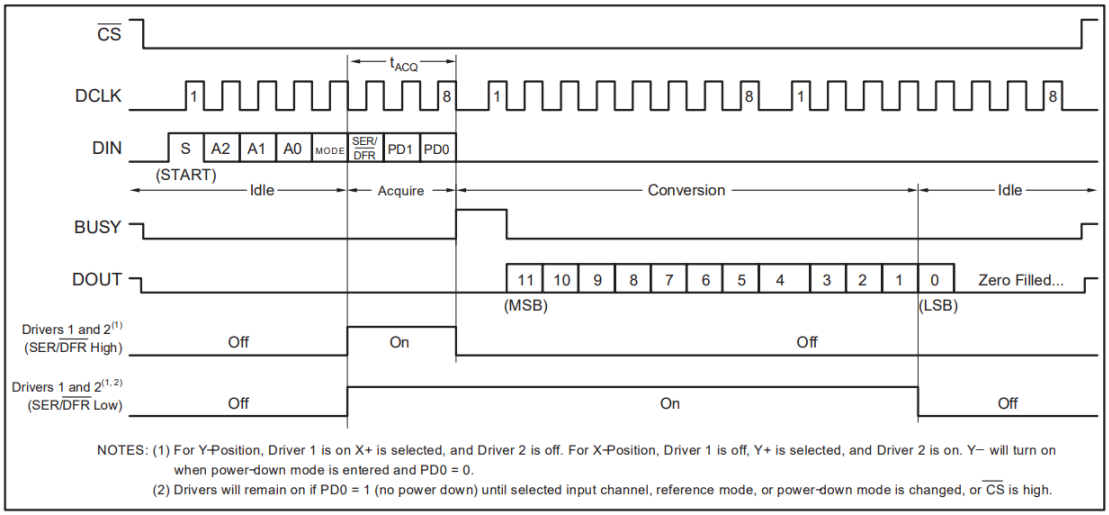

Abbildung 6.7: Signalablauf für den Touch-Controller *TSC2046E* [49]

6.2 Programmable Logic

Im Folgenden wird das Design der Programmable Logic (PL) beschrieben. Dazu wird zunächst ein Überblick über die verwendeten Komponenten gewährt. Im Anschluss wird auf die Funktionsblöcke im Einzelnen eingegangen.

6.2.1 PL Übersicht

In Abbildung 6.8 ist das Block-Design des Projekts zu sehen. Als Entwicklungsumgebung wurde Vivado in der Version 2018.2 verwendet. Bei dem Block-Design handelt es sich um eine Darstellung der PL Seite des Systems. Zu unterscheiden sind grundsätzlich die IP-Cores, welche von Vivado zur Verfügung gestellt werden und denjenigen, welche für diese Anwendung selbst entworfen wurden.

Zu den vorgefertigten IP-Cores zählen:

- Das *ZYNQ7 Processing System* in der Mitte der Abbildung. Dieses ist für die Anbindung der PL an das PS verantwortlich. Es lassen sich einige Parameter, wie die verwendete Taktfrequenz und die verwendeten Schnittstellen (USB, SD-Karte, etc.) einstellen. Die Taktfrequenz des PS wurde dabei nicht verändert, lediglich dem in der PL verwendetem Taktsignal *FCLK_CLK0* wurde eine Frequenz von 100MHz zugewiesen.
- Der Block *Processor System Reset* welcher keiner Konfiguration bedarf und sich um das Zurücksetzen der AXI-Komponenten kümmert.
- Der *AXI Interconnect* Block welcher die verschiedenen AXI-Slaves mit dem AXI-Master, dem PS in diesem Fall, verbindet.
- Zwei *AXI GPIO* Blöcke (*btn_5bits* und *led_8bits*) welche die Verwendung der, auf dem Zedboard vorhandenen, Druckknöpfe und LEDs über das PS ermöglichen. Dies ist notwendig, da diese mit der PL Seite des Zynq SoCs verbunden sind (s. Abb. 3.2). Für die Druckknöpfe wurde eine Interrupt Funktionalität konfiguriert.
- Der *AXI TIMER* Block welcher dem PS zwei 32Bit Timer zur Verfügung stellt. Ein Interrupt ist konfiguriert.
- Der *Concat* Block welcher die Interrupt Signale des Timers und der Druckknöpfe zu einem Bus zusammenführt und an das PS weiterleitet.

Zu den eigens entwickelten IP-Cores zählen:

- Der *Touch_IP* Block welcher für die Kommunikation mit dem Touchscreen-Anschlussboard zuständig. Er stellt die erhobenen Messdaten dem PS zur Verfügung.
- Der *HDMI_IP* Block welcher die Generierung der Bild- und Timingdaten für das HDMI-Display übernimmt. Über die Kommunikation mit dem PS erhält er Anweisungen über den anzuzeigenden Inhalt.

Bei dem *ZedboardOLED_v1.0* Block handelt es sich um einen Opensource IP-Core des Anbieters *Embedded Centric* [19]. Über diesen wird das Organic Light-Emitting Diode (OLED)-Display, welches sich direkt auf dem Zedboard befindet (s. Abb. 3.1 unten, mittig), angesteuert. Über das PS wird gesteuert welcher Text auf dem Organic Light-Emitting Diode (OLED)-Display dargestellt wird. Diese Funktion wird zur Menüführung verwendet.

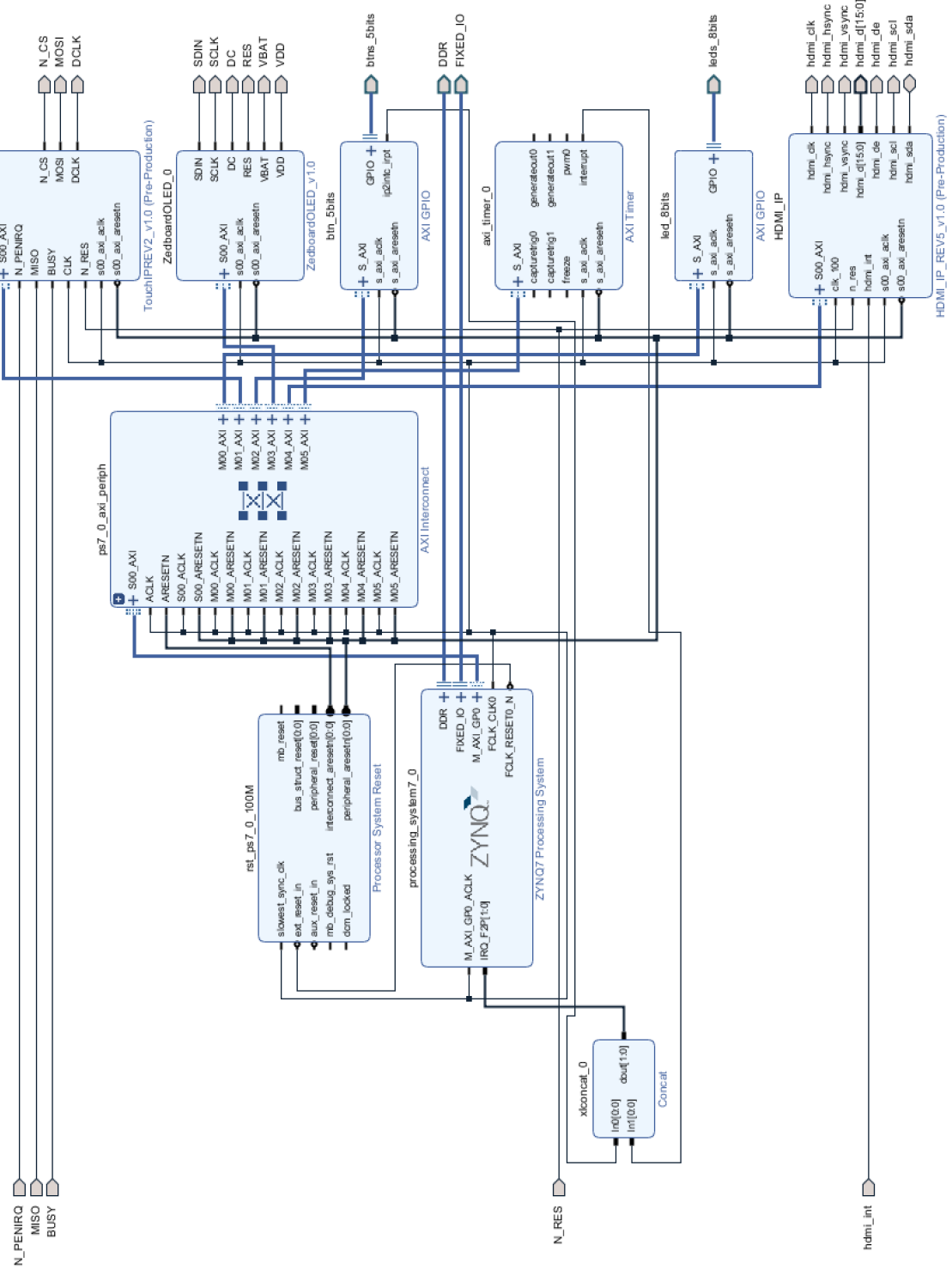

Abbildung 6.8: Block Design des Systems in Vivado [64]

6.2.2 Touchscreen IP-Core

In diesem Abschnitt wird die Implementierung der PL-seitigen Touchscreen-Funktionen beschrieben. Der IP-Core *Touch_IP* (s.Abb. 6.8) ist für die Kommunikation mit dem Touchscreen-Anschlussboard und die Weiterleitung der Informationen an das PS verantwortlich.

Insgesamt besteht das Touchscreen-System aus drei Komponenten (s. Abb. 6.9). Dem Touchscreen-Overlay, dem Touchscreen-Anschlussboard und dem Zedboard. Auf dem Zedboard ist ein Teil der Funktionalität in der PL realisiert und ein Teil in dem PS.

Ein grobes Blockschaltbild des Touch_IP-Cores ist in Abbildung 6.10 zu sehen. Er besteht aus dem Data_Handler und dem SPI_Treiber. Die Verknüpfung dieser beiden Komponenten erfolgt im Touch_Controller. Der Touch_Controller ist in einem, hier vereinfacht dargestelltem, AXI4-Lite-Wrapper eingebettet. Die Signale vom und zum Touchscreen-Anschlussboard sind bereits in Tabelle 6.1 beschrieben. In Tabelle 6.2 kann eine Beschreibung der übrigen Signale gefunden werden. Die internen Signale welche den Data_Handler und den SPI_Treiber verbinden und den Anschluss an das PS ermöglichen sind in Tabelle 6.3 aufgeführt.

6.2.2.1 Axi4-Interface des Touchscreen IP-Cores

Der AXI_Inst Block ist verantwortlich für die Kommunikation zwischen dem Touchscreen IP-Core und dem PS. Auf die genaue Implementierung soll hier nicht eingegangen werden, da diese von Vivado zur Verfügung gestellt wird. Mit dem *Create and Package New IP* Command wurde ein IP-Core mit AXI4-Lite Interface mit 7 32-Bit Registern erstellt. Anschließend wurden die notwendigen Ein- und Ausgangssignale hinzugefügt und die *Data_Handler*- und *SPI_Treiber*-Blöcke als Komponenten im VHDL-Code eingebettet. Die Belegung der 7 Register über welche die Kommunikation erfolgt ist in Tabelle 6.4 aufgeführt[2]. Die meisten Register sind nicht vollständig belegt. Alle nicht belegten Bits sind dauerhaft 0. Bei dem Register 6 handelt es sich um einen Identifikator für den IP-Core. Durch Auslesen dieses Registers kann das PS sicherstellen, dass es mit dem richtigen IP-Core kommuniziert.

[2]Der Operator „&“, welcher bei der Belegung von Register 4 verwendet wird, ist hier zu lesen wie in VHDL. Er entspricht der bitweisen Verknüpfung der Signale.

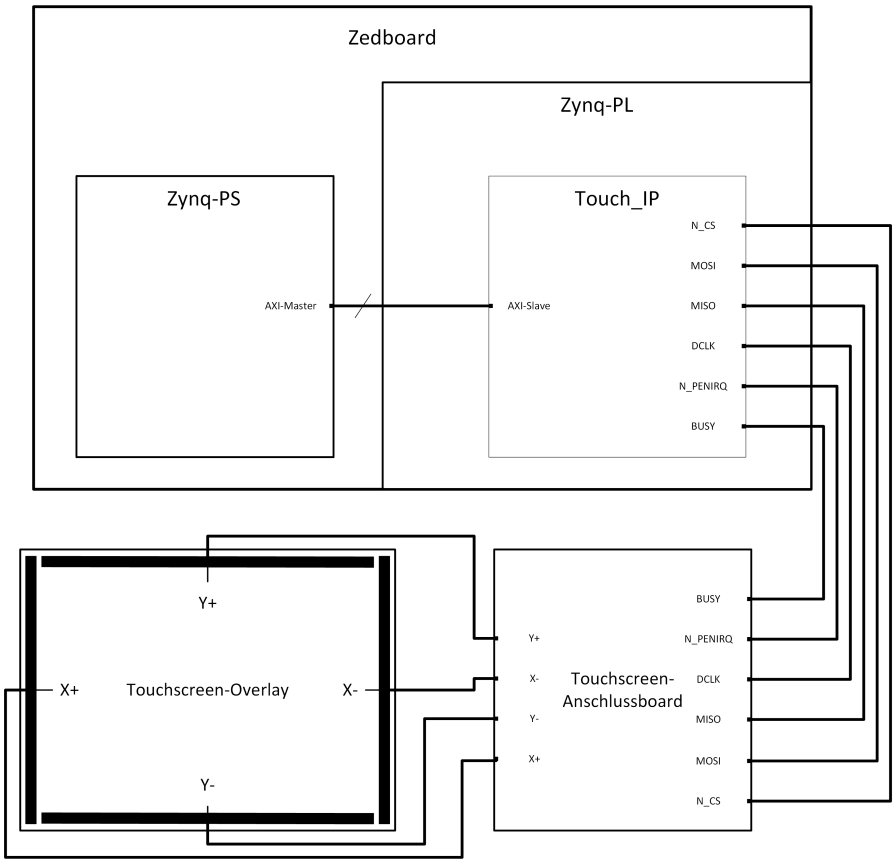

Abbildung 6.9: Grundsätzlicher Aufbau des Touchsystems

Signalname	Quelle	Beschreibung
CLK	PS	Taktsignal, 100 MHz
N_RES	Schalter auf Zedboard	Low-Aktives Resetsignal
S00_AXI	Teilweise Touch-IP-Core, teilweise AXI-Master (PS)	AXI-Slave-Port. Hier vereinfacht dargestellt. Eine Aufführung aller Signale des Ports kann der Tabelle 3.2 entnommen werden.
S00_axi_aclk	PS	AXI4-Taktsignal
S00_axiaresetn	PS-Reset	AXI4-Resetsignal

Tabelle 6.2: Signalbeschreibung des Touch_IP-Cores

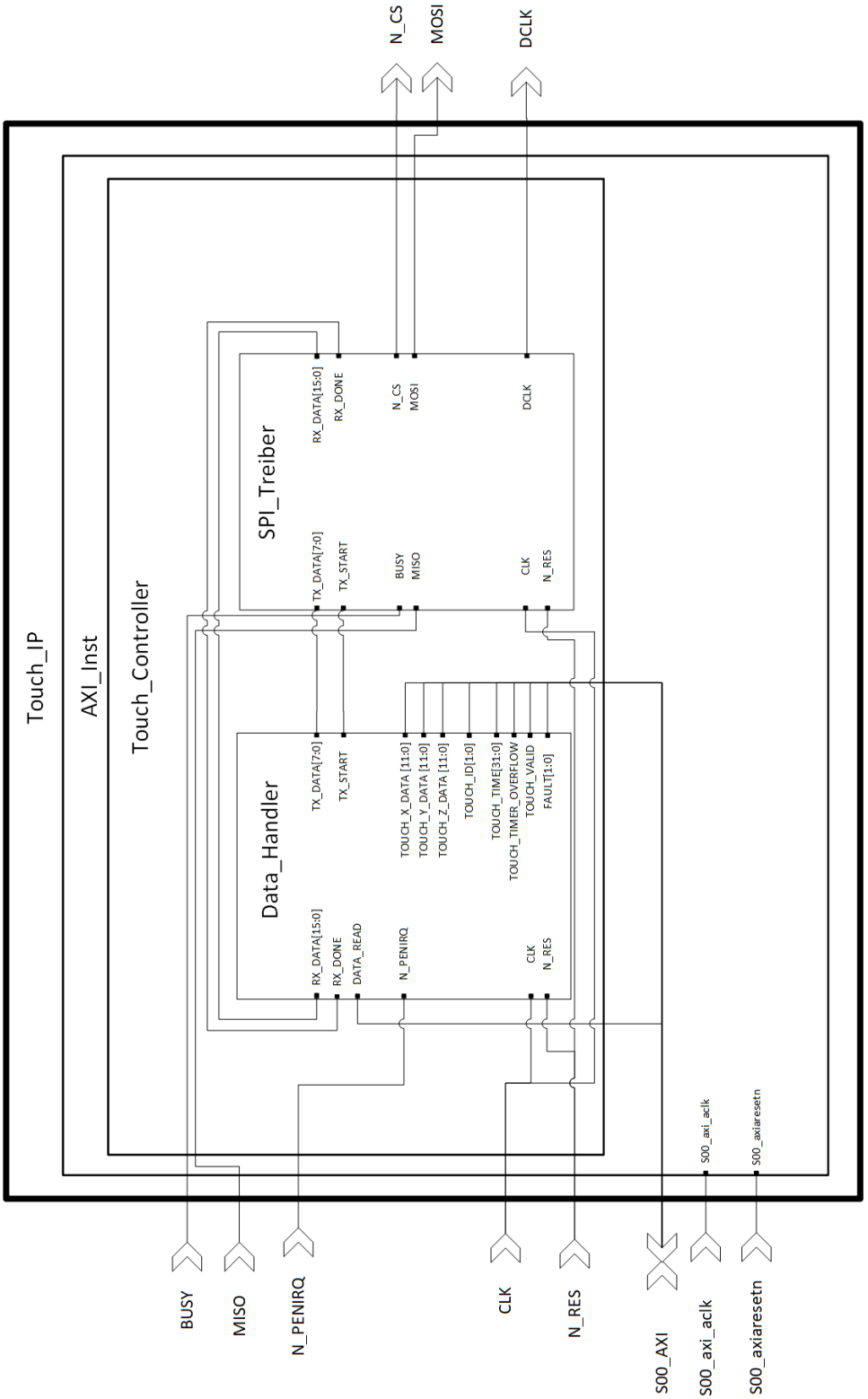

Abbildung 6.10: Blockdiagramm Touch-IP. AXI Umsetzung vereinfacht dargestellt.

Signalname	Quelle	Beschreibung
RX_DATA[15:0]	SPI_Treiber	Die vom Touchscreen-Anschlussboard erhaltenen Messdaten, welche vom Data_Handler gespeichert und dem PS zur Verfügung gestellt werden.
RX_DONE	SPI_Treiber	Teilt dem Data_Handler mit, dass die an RX_DATA anliegenden Daten gültig sind.
DATA_READ	PS	Handshake-Signal. Teilt dem Data_Handler mit, dass das PS die gültigen, an TOUCH_X_DATA, TOUCH_Y_DATA, TOUCH_Z_DATA, TOUCH_ID und TOUCH_TIME anliegenden Daten gelesen hat und bereit ist eine neue Messung zu erfassen.
TX_DATA[7:0]	Data_Handler	Beinhaltet die Daten, welche der SPI_Treiber an das Anschlussboard senden soll. Diese Bestimmen, welche Messung das Anschlussboard ausführt.
TX_START	Data_Handler	Teilt dem SPI_Treiber mit, dass die an TX_DATA anliegenden Signale gültig sind und fordert ihn auf die Übertragung zu starten.
TOUCH_X_DATA [11:0]	Data_Handler	Beinhaltet die Messung des X-Wertes der Berührung.
TOUCH_Y_DATA [11:0]	Data_Handler	Beinhaltet die Messung des Y-Wertes der Berührung.
TOUCH_Z_DATA [11:0]	Data_Handler	Beinhaltet die Messung des Z-Wertes der Berührung.
TOUCH_ID[1:0]	Data_Handler	Die Information, ob die Messung vom Anfang, der Mitte oder dem Ende einer Berührung stammt.
TOUCH_TIME [31:0]	Data_Handler	Der Timer-Wert bei Abschluss der Messung.
TOUCH_TIMER_ OVERFLOW	Data_Handler	Gibt an, ob der Timer übergelaufen ist. Dies kann bei langen Pausen zwischen zwei Berührungen auftreten.
FAULT[1:0]	Data_Handler	Informationen über den Fehlerzustand des Data_Handlers

Tabelle 6.3: Signalbeschreibung des Data_Handlers und SPI_Treibers

Adresse	Belegte Bits	Belegung
0	11...0	TOUCH_X_DATA
1	11...0	TOUCH_Y_DATA
2	11...0	TOUCH_Z_DATA
3	31...0	TOUCH_TIME
4	5...0	TOUCH_VALID & TOUCH_ID & TOUCH_TIMER_OVERFLOW & FAULT
5	0	DATA_READ
6	31...0	`0x1234aaFF`

Tabelle 6.4: Touchscreen AXI4-Lite-Interface Registerbelegung

6.2.2.2 Data_Handler

Der Data_Handler ist für das Erkennen einer Berührung, die Steuerung des Anschlussboards und die Zusammenstellung der Messdaten verantwortlich.

Er ist als endlicher Automat realisiert. Das entsprechende Zustandsdiagramm ist in Abbildung 6.11 zu finden.

Nach der Initialisierung des Systems, dem Auslösen des Resets oder dem Ende einer Berührung befindet sich der Automat im *Idle*-Zustand. Bei Erkennung einer Berührung veranlasst der Data_Handler nacheinander das Messen und Auslesen der X-, Y- und Z-Werte der Berührung. Eine Berührung wird durch einen Lowpegel an *N_PENIRQ* erkannt. Das Messen und Auslesen der Werte wird durch Anlegen der Steuersignale an *TX_DATA* und *TX_START* veranlasst. Dies bringt den SPI_Treiber dazu, die Steuersignale an das Anschlussboard zu senden. Nachdem der SPI_Treiber die Messdaten empfangen hat, stellt er sie dem Data_Handler über *RX_DATA* zur Verfügung. Die Messdaten werden zusammen mit einer Touch-ID und einem Zeitstempel dem PS über das AXI4-Lite-Interface zur Verfügung gestellt. Über das Signal *TOUCH_VALID* wird dem PS mitgeteilt, dass eine neue, gültige Messung in den Registern gespeichert ist. Das PS hat anschließend 20ms Zeit die Messwerte zu sichern und dies dem Data_Handler über das Signal *DATA_READ* mitzuteilen.

Solange das Touchscreen-Overlay betätigt und damit das Anschlussboard einen Lowpegel an *N_PENIRQ* ausgibt, erfasst der Data_Handler alle 20ms die X-, Y- und Z-Werte der Berührung.

Es sind außerdem zwei Fehlerzustände implementiert. Diese signalisieren ein Problem bei der Kommunikation mit dem Anschlussboard (*FAULT*=‚01‘) oder mit dem PS (*FAULT*=‚10‘). Die Fehlerzustände können nur über einen Lowpegel am Resetsignals *N_RES* verlassen werden. Erhält der Data_Handler nach dem Bereitstellen der Werte und dem Signalisieren der Gültigkeit innerhalb einer Wartezeit von 20ms keine Bestätigung vom PS, geht er in den Fehlerzustand ‚10‘ über. Eine Möglichkeit dem PS mehr Zeit zu geben die Werte auszulesen und zwischenzuspeichern, wäre die Implementierung eines Puffers zwischen dem Data_Handler und dem PS. Dies könnte beispielsweise ein FIFO sein. Der Fehlerzustand ‚01‘ wird erreicht, wenn keine plausiblen Messwerte vom Anschlussboard empfangen werden.

6.2.2.3 SPI_Treiber

Der SPI_Treiber ist für die Kommunikation mit dem Touchscreen-Anschlussboard verantwortlich. Er ist als endlicher Automat realisiert. Ein vereinfachtes Zustandsdiagramm ist in Abbildung 6.12 zu finden.

Nach Aufforderung durch den Data_Handler (Highpegel an *TX_START*) werden die 8 Bit, welche an *TX_DATA* anliegen, an das Anschlussboard übertragen. Sobald das Anschlussboard bereit ist (Lowpegel an *BUSY*) werden 16 Bit eingelesen und an dem Port *RX_DATA* bereitgestellt. Gleichzeitig wird das Signal *RX_DONE* gesetzt, welches dem Data_Handler die Gültigkeit der an *RX_DATA* anliegenden Signale mitteilt. Durch das Rücksetzen von *TX_START* kann der Data_Handler dem SPI_Treiber nun mitteilen, dass die Daten gesichert wurden. Der SPI_Treiber befindet sich wieder im *IDLE*-Zustand.

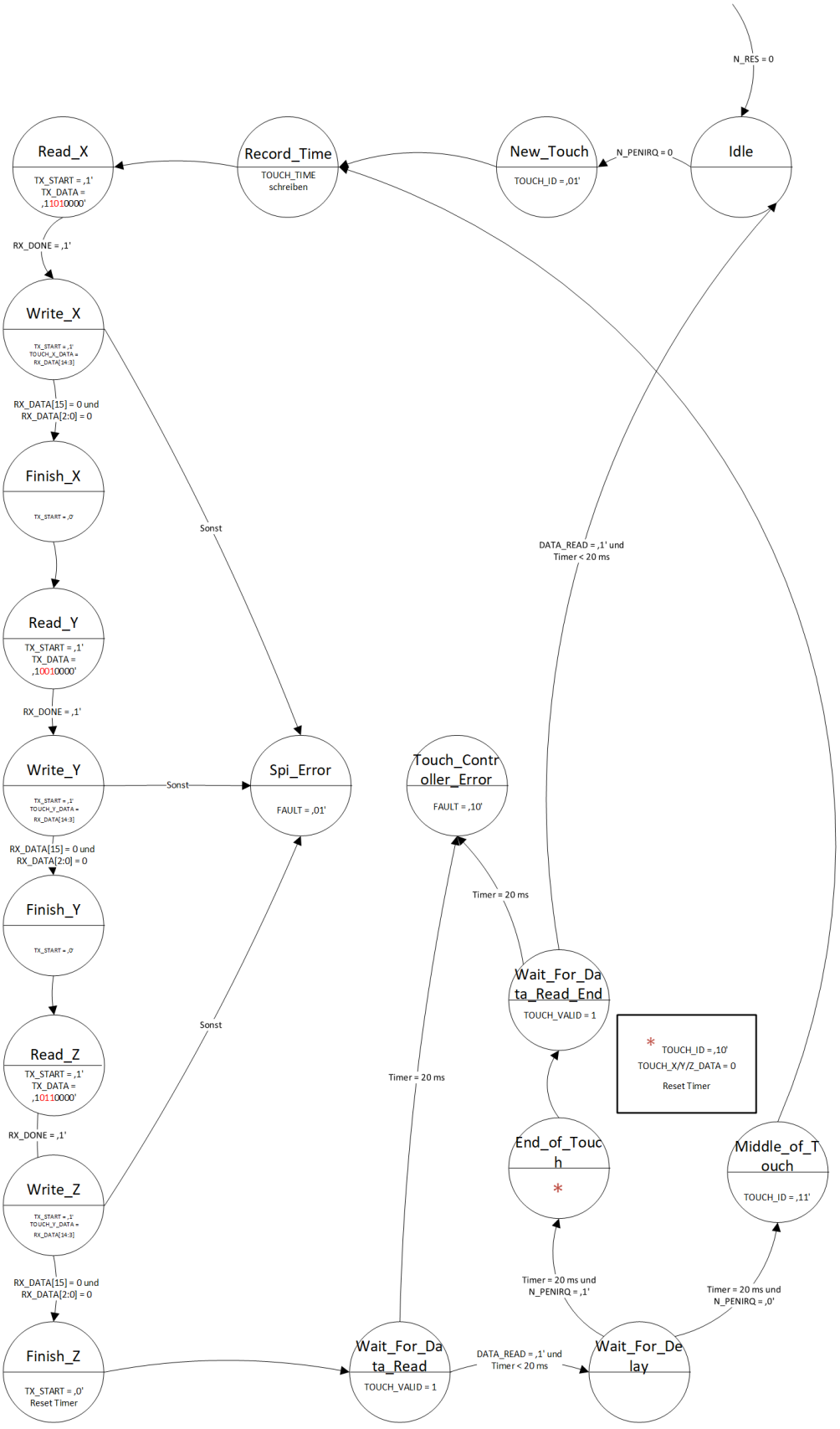

Abbildung 6.11: Zustandsdiagramm Data_Handler

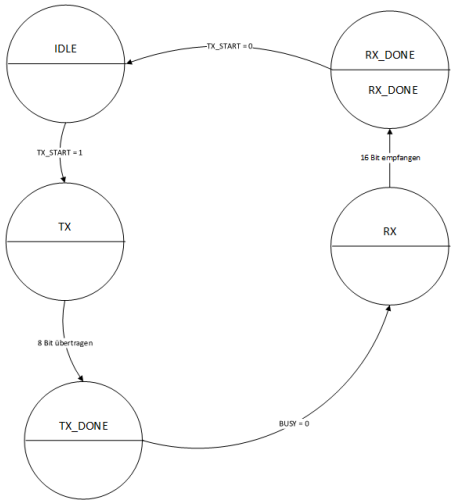

Abbildung 6.12: Zustandsdiagramm SPI Treiber

6.2.3 HDMI IP-Core

In diesem Abschnitt wird die Implementierung der PL-seitigen Display-Funktionen beschrieben. Der IP-Core *HDMI_IP* (s. Abb. 6.8) ist für die Erzeugung der Timing- und Bilddaten des HDMI-Displays verantwortlich.

Insgesamt besteht das Display-System aus zwei Komponenten (s. Abb. 6.13). Dem HDMI-Display (s. Abb. 5.3) und dem Zedboard. Auf dem Zedboard ist ein Teil der Funktionalität in der PL realisiert und ein Teil in dem PS. Die Kommunikation zwischen dem HDMI_IP-Core und dem Display läuft über den HDMI-Transmitter *ADV7511* [3]. Auf die Funktionsweise des Transmitters wird im weiteren nicht genauer eingegangen. Daher ist auch die Verbindung zwischen dem Transmitter und dem HDMI-Display vereinfacht dargestellt.

Ein grobes Blockschaltbild des HDMI_IP-Cores ist in Abbildung 6.14 zu sehen. Er besteht im wesentlichen aus dem Colour_Controller und dem HDMI_Treiber. Die Verknüpfung dieser zwei Komponenten erfolgt im Modul HDMI_Toplevel. Dieses ist, ebenso wie der Touch_Controller, in einem AXI4-Lite-Wrapper eingebettet. Die äußeren Signale sind in Tabelle 6.5 beschrieben. Eine Beschreibung der internen Signale, welche den Colour_Controller und den HDMI_Treiber verbinden kann in Tabelle 6.6 gefunden werden.

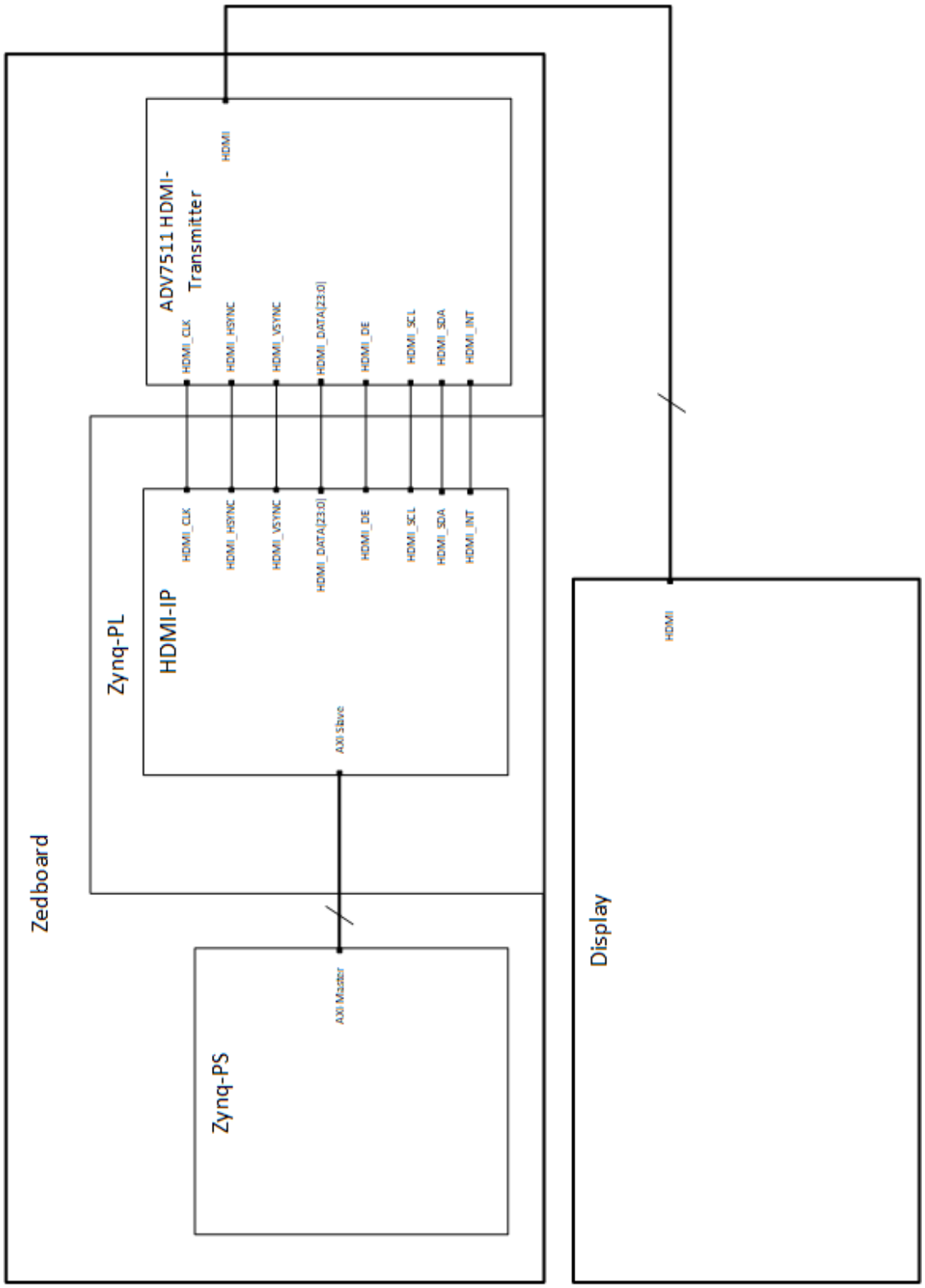

Abbildung 6.13: Grundsätzlicher Aufbau des Display Systems

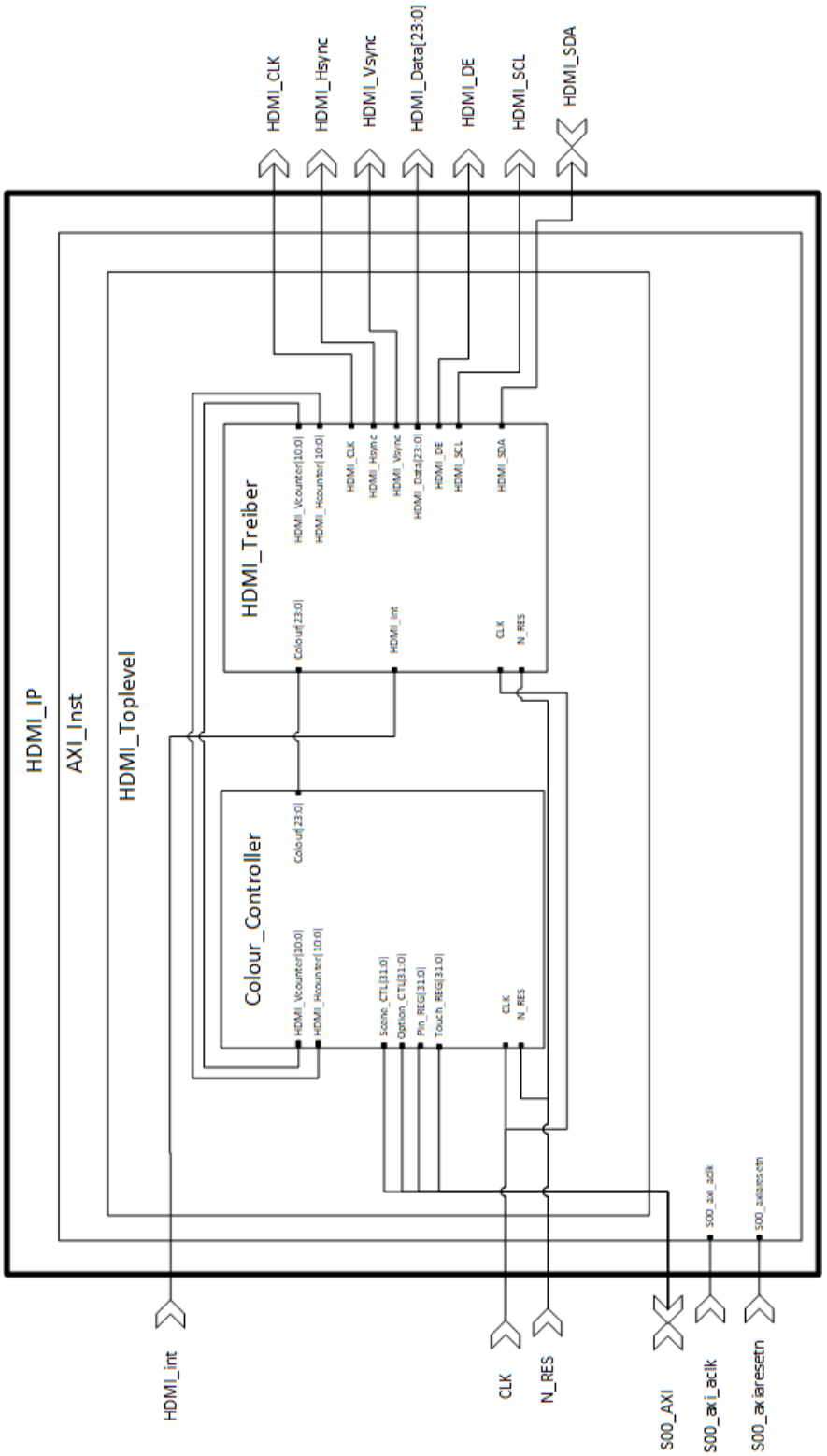

Abbildung 6.14: Blockdiagramm HDMI-IP. AXI Umsetzung vereinfacht dargestellt.

Signalname	Quelle	Beschreibung
HDMI_int	ADV7511	Interrupt Signal vom Transmitter zum HDMI_Treiber
HDMI_CLK	HDMI_Treiber	Video-Taktsignal
HDMI_Hsync	HDMI_Treiber	Horizontales Synchronisierungssignal
HDMI_Vsync	HDMI_Treiber	Vertikales Synchronisierungssignal
HDMI_Data[24:0]	HDMI_Treiber	Bilddaten im YCbCr-Farbformat
HDMI_DE	HDMI_Treiber	Data-Enable-Signal
HDMI_SCL	HDMI_Treiber	„Serial Port Data Clock“ - Taktsignal für die Kommunikation zwischen HDMI_Treiber und Transmitter
HDMI_SDA	ADV7511 und HDMI_Treiber	„Serial Port Data I/O“ - Datensignal für die Kommunikation zwischen HDMI_Treiber und Transmitter
CLK	PS	Taktsignal, 100 MHz
N_RES	Schalter auf Zedboard	Low-Aktives Resetsignal
S00_AXI	Teilweise HDMI-IP-Core, teilweise AXI-Master (PS)	AXI-Slave-Port. Hier vereinfacht dargestellt. Eine Aufführung aller Signale des Ports kann der Tabelle 3.2 entnommen werden.
S00_axi_aclk	PS	AXI4-Taktsignal
S00_axiaresetn	PS-Reset	AXI4-Resetsignal

Tabelle 6.5: Signalbeschreibung des HDMI_IP-Cores

Signalname	Quelle	Beschreibung
HDMI_Vcounter[10:0]	HDMI_Treiber	Momentane vertikale Position im Bild.
HDMI_Hcounter[10:0]	HDMI_Treiber	Momentane horizontale Position im Bild.
Scene_CTL[31:0]	PS	Bestimmt welches Bild dargestellt wird.
Option_CTL[31:0]	PS	Kontrolliert Optionen in den einzelnen Bildern.
Pin_REG[31:0]	PS	Bestimmt welche Pin angezeigt wird (nur im Hauptbild S_Main).
Touch_REG[31:0]	PS	Bestimmt die Position, an welcher die Berührung angezeigt wird.
Colour	Colour_Controller	Die vom Colour_Controller errechnete Farbe für das aktuelle Pixel.

Tabelle 6.6: Signalbeschreibung des Colour_Controllers und HDMI_Treibers

6.2.3.1 Axi4-Interface des HDMI IP-Cores

Der AXI_Inst Block ermöglichst die Kommunikation zwischen dem HDMI_IP-Core und des PS. Auch in diesem Fall wurde mit dem *Create and Package New IP* Command ein AXI4-Lite Template erzeugt, allerdings mit 5 32-Bit Registern. Analog zu Kapitel 6.2.2.1 wurden die notwendigen Signale und Komponenten hinzugefügt und die Register mit dem HDMI_Controller verknüpft. Die Belegung der 5 Register ist in Tabelle 6.7 aufgeführt. In diesem Fall befindet sich der Identifikator des IP-Cores in Register 4. Die Register 0 bis 3 werden von dem PS geschrieben und vom Colour_Controller gelesen. Hierüber erfolgt die Auswahl des Bildes und die Steuerung der Inhalte.

Adresse	Belegte Bits	Belegung
0	31...0	Scene_CTL[31:0]
1	31...0	Option_CTL[31:0]
2	31...0	Pin_REG[31:0]
3	31...0	Touch_REG[31:0]
4	31...0	`0xFFAB8301`

Tabelle 6.7: HDMI-IP AXI4-Lite-Interface Registerbelegung

6.2.3.2 Colour_Controller

Der Colour_Controller errechnet aus der aktuellen Position im Bild und den Steuerdaten die Farbe des aktuellen Pixels. Die Information über die aktuelle Position erhält er über *HDMI_Vcounter* und *HDMI_Hcounter* vom HDMI_Treiber. Die Steuerdaten erhält er über das AXI-Interface vom PS.

Die Berechnung der Bilddaten erfolgt rein kombinatorisch. Der Einfachheit halber sind alle Bildelemente über Rechtecke realisiert. Es wurde das VHDL Paket *pkg_colour_func* geschrieben. In diesem befinden sich die Funktionen, welche zur Generierung der Bildobjekte verwendet werden. Im einzelnen sind dies:

- cf_rect(x_start, y_start, x_stop, y_stop, hcounter, vcounter: unsigned(10 downto 0)):
  - Zeichnet ein Rechteck vom Punkt (x_{start}, y_{start}) bis zum Punkt (x_{stop}, y_{stop})
  - Gibt eine 1 zurück wenn die aktuelle Position $(h_{counter}, v_{counter})$ innerhalb dieses Rechtecks ist. Sonst wird eine 0 zurückgegeben.
- cf_number(x, y, hcounter, vcounter: unsigned(10 downto 0); number: STD_LOGIC_VECTOR(4 downto 0); big_not_small: std_logic):
  - Zeichnet ein Symbol je nach Inhalt von *number* am Punkt (x, y).
  - Gibt eine 1 zurück wenn die aktuelle Position $(h_{counter}, v_{counter})$ innerhalb des Symbols ist. Sonst wird eine 0 zurückgegeben.
  - Der Punkt (x, y) ist die oberee linke Ecke des Symbols.
  - Die einzelnen Symbole sind im Format einer Siebensegmentanzeige realisiert. Zum Darstellen der einzelnen Segmente wurde die Funktion *cf_rect()* verwendet.
  - Die mit *number* auswählbaren Symbole sind für die Werte `0x0...0X9` die Zahlen 0 bis 9. Diese werden für die Darstellung der Pin verwendet. Mit den Werten `0xA...0XD` lassen sich die Buchstaben $\{A, U, c, r\}$ auswählen. Diese werden für die Darstellung der Schriftzüge **Acc** und **USR** verwendet (s. Abb. 5.4). Für die Darstellung des Buchstaben **S** wird die Zahl **5** verwendet.

  - Über das Bit *big_not_small* lässt sich einstellen, ob das Symbol 20x40 oder 40x80 Pixel groß ist. Die Großen Zahlen werden im Pinfeld verwendet. Die kleinen zum Anzeigen der vorgegebenen Pin im oberen linken Bereich des Bildes.

- cf_pinfield(x, y, hcounter, vcounter: unsigned(10 downto 0)):
  - Zeichnet das Pinfeld mit der oberen linken Ecke am Punkt (x, y).
  - Das Pinfeld besteht aus dem Gitter und den Zahlen 0 bis 9 (s. Abb. 5.4). Das Gitter wird über Abrufe der Funktion *cf_rect()* realisiert. Die Zahlen entsprechend über *cf_number()*.
  - Gibt eine 1 zurück, wenn die aktuelle Position $(h_{counter}, v_{counter})$ innerhalb eines Elements des Pinfeldes liegt. Sonst wird eine 0 zurückgegeben.

Um die Funktionsweise zu verdeutlichen folgt nun ein kurzes Beispiel. Nehmen wir an wir hätten ein Display mit einer Auflösung von 4x4 Pixel. In diesem Fall würde der $h_{counter}$ immer von 0 bis 3 hochzählen und beim Überlauf auf 0 zurückspringen. Der $v_{counter}$ verhält sich ebenso, nur, dass das Hochzählen hier nicht bei jedem Pixel geschieht, sondern beim Überlaufen von $h_{counter}$. Die Pixel (x, y) würden also in der Reihenfolge $(0,0), (1,0), (2,0), (3,0), (0,1), (1,1), \ldots, (3,3), (0,0), \ldots$ durchlaufen werden. Durch einen Funktionsaufruf von *cf_rect()* mit den Werten x_start=1, y_start=0, x_stop=1 und y_stop=2 würde also ein Wert von 1 für die Pixel $(1,0)$, $(1,1)$ und $(1,2)$ zurückgeben. In diesen Fällen setzt der Colour_Controller die Farbe (*Colour[23:0]*) auf Blau. In allen anderen Fällen auf Weiß. Der HDMI_Treiber überträgt diese Farbinformationen zusammen mit den Timingdaten an das Display und wir erhalten das in Abbildung 6.15 gezeigte Bild.

Über das Register *Scene_CTL* können folgende Bilder ausgewählt werden[3]:

- 0x4 $\hat{=}$ S_CALIBRATE:
  - Wird zum Kalibrieren des Displays verwendet.
  - Es wird ein 41x41 Pixel großes Kreuz mit einer Strichstärke von 5 Pixel dargestellt. Der Mittelpunkt des Kreuzes wird über das Register *Option_CTL*

[3] Während der Entwicklung wurden noch weitere Testbilder verwendet. Diese sind aus Gründen der Optimierung in der finalen Version allerdings auskommentiert und werden daher hier nicht genauer beschrieben.

gesteuert. Dabei entspricht der X-Wert entspricht *Option_ CTL[10...0]* und der Y-Wert *Option_ CTL[22...12]*.

- 0x5 $\hat{=}$ S_MAIN:
  - Wird bei der Pineingabe und Authentifizierung verwendet.
  - Es wird ein Bild entsprechend Abbildung 5.4 ausgegeben.
  - Über *Option_ CTL[15...0]* wird die Farbe der Acht Kontrollboxen gesteuert. Dabei werden für jede der Acht Boxen zwei Bit verwendet. *Option_ CTL[1...0]* für die erste (linke) Box und so weiter. Die Werte werden wie folgt behandelt:
    - 0x0 $\hat{=}$ Box wird nicht angezeigt.
    - 0x1 $\hat{=}$ Box wird schwarz angezeigt.
    - 0x2 $\hat{=}$ Box wird grün angezeigt.
    - 0x3 $\hat{=}$ Box wird rot angezeigt.
  - Die User-Nummer wird über *Option_ CTL[23...16]* gesteuert. Wobei die Zahl im Binary-Coded Decimal (BCD)-Format vorliegen muss.
  - Die Account-Nummer wird über *Option_ CTL[31...24]* gesteuert. Wobei die Zahl im BCD-Format vorliegen muss.
  - Die Pin welche angezeigt wird ist über das Register *PIN_ REG* bestimmt. Wobei die Pin im BCD-Format vorliegen muss.

Ist das Bit *Touch_ REG[31]* auf 1 gesetzt wird an der Stelle X=*Touch_ REG[10...0]*, Y=*Touch_ REG[22...12]* ein blaues Quadrat angezeigt. Dies wird für die Visualisierung der letzten Berührung verwendet. Die Darstellung erfolgt unabhängig von den anderen Bildeinstellungen. Die Breite und Höhe des Quadrates entsprechen dem doppelten Wert von *Touch_ REG[30...24]* plus Eins[4].

[4] Die Erhöhung des Wertes um Eins erfolgt damit das Quadrat einen definierten Mittelpunkt hat. Aus diesem Grund sind beispielsweise auch die Zahlen 21x21 bzw. 41x41 Pixel groß und nicht 20x20 bzw. 40x40.

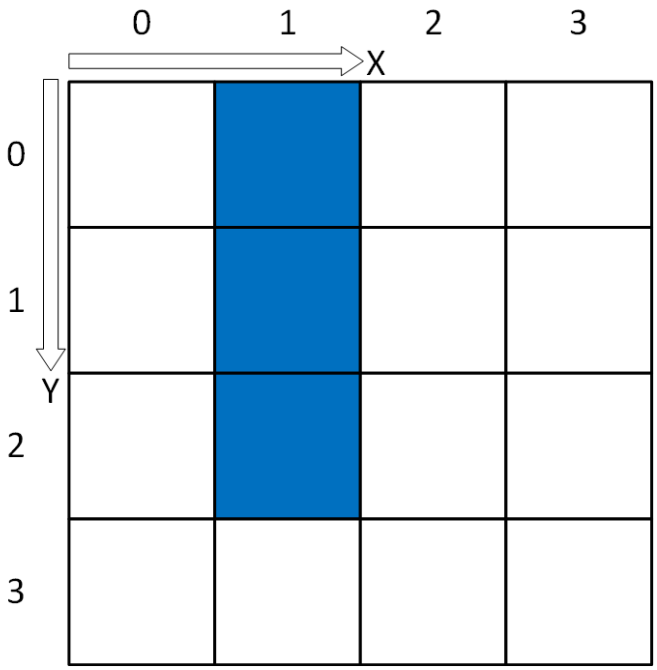

Abbildung 6.15: Funktionsbeispiel für *cf_rect* anhand eines 4x4 Pixel Displays.

6.2.3.3 HDMI_Treiber

Der HDMI_Treiber ist für die Generierung der Timingdaten und die Kommunikation mit dem HDMI-Transmitter ADV7511 verantwortlich. Der HDMI_Treiber wurde *nicht* selbständig entwickelt. Stattdessen wurde eine Opensource Anwendung [27] an die Auflösung des verwendeten Displays angepasst. Zusätzlich wurden die ursprünglich internen Signale *HDMI_VCounter*, *HDMI_HCounter* und *Colour* nach außen geführt. Dies ermöglicht die Generierung der Bilddaten über den Colour_Controller.

6.3 Processing System

Im Folgenden wird die Programmierung des PS beschrieben. Hierfür wird zunächst eine Übersicht des Programmablaufs gewährt bevor die wesentlichen Abschnitte genauer beschrieben werden.

6.3.1 Übersicht Programmablauf

Die wesentlichen Aufgaben des PS umfassen:

- Die Steuerung der Bildinhalte und damit die Kommunikation mit dem HDMI_IP-Core.
- Das Auslesen der Messdaten vom Touch_IP-Core.
- Das Filtern und Transformieren der Messdaten.
- Das Speichern der Messdaten auf einer SD-Karte. Dies ist für die Auswertung und das Training der Klassifikatoren auf dem PC notwendig.
- Das Ausführen des Authentifikationsalgorithmus.
- Eine Menüführung über das OLED-Display und die Taster des Zedboards.

In der Abbildung 6.16 ist der grundsätzliche Ablauf des Programms dargestellt. Nach dem anfänglichen Setup geht das Programm in eine Endlos-Schleife der Hauptschleife (engl. Main Loop) über. In dieser werden die oben beschriebenen Aufgaben abgearbeitet. Neben dem Hauptprogramm existieren noch zwei Interrupt Service Routines (ISRs). Diese sind erst nach der Setup-Phase aktiv. Wird eine der Interrupt-Bedingungen erfüllt wird die entsprechende ISR ausgeführt. Nach Abarbeitung der ISR springt das Programm zurück an die Stelle des Hauptprogramms, von der aus er dieses verlassen hat.

Die Interrupts können nicht verschachtelt werden. Wird beispielsweise gerade die Taster-ISR abgearbeitet führt ein Ablaufen des Timers nicht zum sofortigen auslösen der Timer-ISR. Stattdessen wird eine Warteschlange gebildet. Erst nach beenden der Taster-ISR würde die Timer-ISR ausgeführt werden. Da über die Timer-ISR mit dem Touch-IP-Core kommuniziert wird, kann dies zu Problemen führen. Wie in Kapitel 6.2.2.2 beschrieben, erwartet der Data_Handler innerhalb von 20ms eine Rückmeldung vom PS. Dauert die Taster-ISR nun zu lange könnte der Data_Handler in einen Fehlerzustand übergehen.

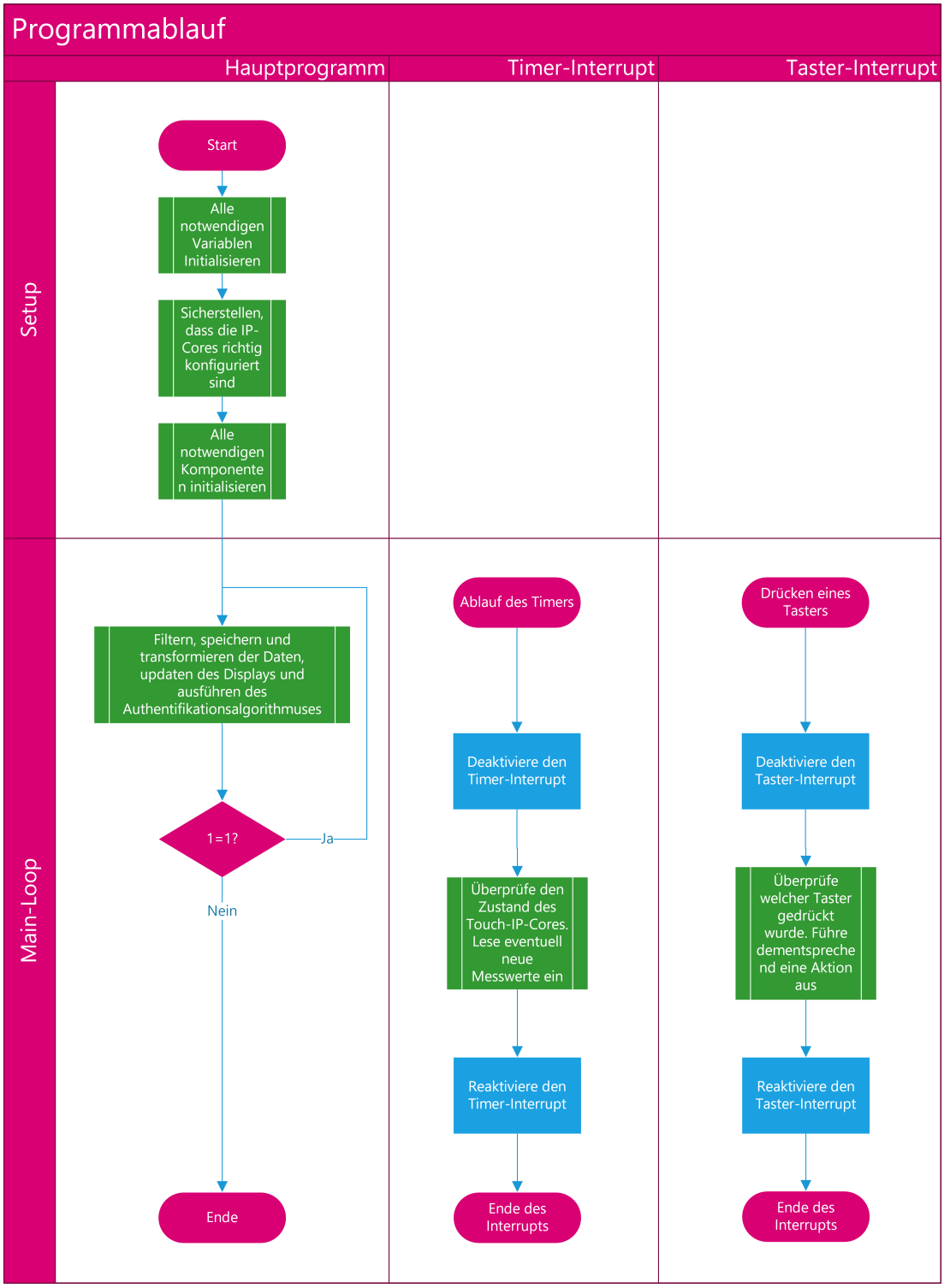

Abbildung 6.16: Programmablaufplan des PS, grobe Darstellung.

Dies könnte beispielsweise vermieden werden, indem man beide ARM-Prozessoren verwendet und die Interrupts parallel abarbeiten lässt. Da dieses Problem in der Praxis nur selten aufgetreten ist, wurde aus Zeitgründen auf eine Verfolgung dieses Ansatzes verzichtet.

Beim Aufruf einer ISR wird der entsprechende Interrupt für die Zeit der Ausführung deaktiviert. Es ist also sehr unwahrscheinlich, dass sich beide Interrupts gleichzeitig in der Warteschlange befinden. Sollten die beiden Interrupts im gleichen Taktzyklus ausgelöst werden, wird die Timer-ISR vor der Taster-ISR ausgeführt. Ihr wurde eine höhere Priorität zugewiesen.

Die Setup-Phase wird nicht im Detail beschrieben. In ihr werden:

- Variablen deklariert und initialisiert,
- die Konfiguration der IP-Cores überprüft,
- Timer und Interrupts konfiguriert.

Nach der Setup-Phase wird auf dem OLED-Display ein Menü angezeigt. Dieses kann mit Hilfe der Taster navigiert werden. Es ist folgende Bedienung vorgesehen:

1. Stelle über die Taster den Nutzer (engl. User) ein (s. Abb. 5.4 „USR“). Je nach Phase (s. Kap. 5.2) werden:

   a) die Daten der Pineingabe auf der SD-Karte gespeichert. Es wird vermerkt, dass die Daten zu dem ausgewählten Nutzer gehören. Es muss also der Nutzer ausgewählt werden, welcher auch wirklich die Pineingabe durchführt.

   b) die Daten der Pineingabe mit dem für diesen Nutzer hinterlegten Daten verglichen. Es muss also der Nutzer ausgewählt werden dessen Account „angegriffen“ werden soll.

2. Stelle über die Taster den Account (s. Abb. 5.4 „ACC“) ein. Dieser bestimmt welche der drei Pins (s. Kap. 5.2) eingegeben werden soll. Damit bestimmt er auch nach wie vielen Berührungen des Touchscreens eine Eingabe als abgeschlossen gilt.

3. Starte die Pineingabe mit Hilfe der Taster.

4. Nun muss je nach Phase:

a) der ausgewählte Nutzer die eingestellte Pin auf dem Touchscreen eingeben. Diese Daten werden später für das Training der Klassifikatoren verwendet.

b) der Angreifer die eingestellte Pin auf dem Touchscreen eingeben.

5. Während der Pineingabe besteht die Möglichkeit diese zu verwerfen. Dies hat sich als notwendig herausgestellt, da es gelegentlich vorkommt, dass sich Nutzer bei der Eingabe vertippen. Des Weiteren funktioniert der Touchscreen nicht perfekt. Es kann vorkommen, dass Eingaben doppelt oder gar nicht erkannt werden. In diesem Fall bietet es sich an die Daten der Pineingabe zu verwerfen.

6. Die Pineingabe gilt als beendet, sobald die 4, 6 oder 8 Stellen der Pin eingegeben wurden und eine Bestätigung durch das Berühren des grünen Feldes (s. Abb. 5.4) stattgefunden hat. Je nach Phase:

   a) kann die Pineingabe nun gespeichert oder verworfen werden. Die Auswahl erfolgt über die Taster.

   b) wird die Eingabe nun mit den hinterlegten Daten verglichen. Der Authentifikationsalgorithmus wird ausgeführt. War die Authentifikation erfolgreich werden die Kontrollkästchen grün. Wurde die Eingabe vom System abgelehnt werden die Kontrollkästchen rot. Anschließend können die Daten dieser Pineingabe ebenfalls gespeichert oder verworfen werden. Ein Speichern der Daten macht in diesem Fall jedoch nur Sinn, wenn es sich bei dem „Angreifer“ um die Person handelt, auf welche der Klassifikator trainiert wurde. Sprich für den Fall, dass sich jemand in seinen eigenen Account einloggen möchte.

Nach dem Speichern oder dem Verwerfen der Daten kann wieder von vorne begonnen werden. Haben sich der Nutzer und Account nicht geändert kann bei Punkt Drei gestartet werden.

6.3.2 Programmabschnitte im Detail

Im folgendem wird der Ablauf der Main-Loop sowie der zwei Interrupt Service Routines genauer beschrieben.

6.3.2.1 Main-Loop

Nach dem Setup wird die Main-Loop beziehungsweise Hauptschleife ausgeführt. Im normalen Programmablauf gibt es keine Abbruchbedingung für diese Schleife. Solange das Programm läuft, wird sie also immer wieder durchlaufen. Die Main-Loop ist für die folgenden Funktionen verantwortlich:

- Wenn der Nutzer die Speicherung der Pineingabe veranlasst hat, schreibe die Daten auf die SD-Karte und lösche sie anschließend aus dem Programmspeicher.
- Wenn der Nutzer das Verwerfen der aktuellen Pineingabe veranlasst hat, lösche die Daten aus dem Programmspeicher.
- Wenn eine Berührung beendet ist, filtere die gemessenen Daten und speichere diese im Programmspeicher. Teile der Timer-ISR mit, dass die Daten gesichert wurden und überschrieben werden können. Errechne zu welchem Pixel auf dem Display die Messwerte passen und zeige die Berührung auf dem Display an.
- Wenn die Pineingabe gestartet wurde, zähle mit wie viele Stellen bereits eingegeben wurden. Zeige dies über die Kontrollkästchen an.
- Sobald alle Stellen der Pin eingegeben vermerke, dass der Nutzer entscheiden muss, ob die Eingabe gespeichert oder verworfen werden soll. Führe je nach Phase zusätzlich den Authentifikationsalgorithmus aus und teile das Ergebnis dem Nutzer über die Farbe der Kontrollkästchen mit.

Der entsprechende Programmablaufplan kann in Abbildung 6.17 gefunden werden.

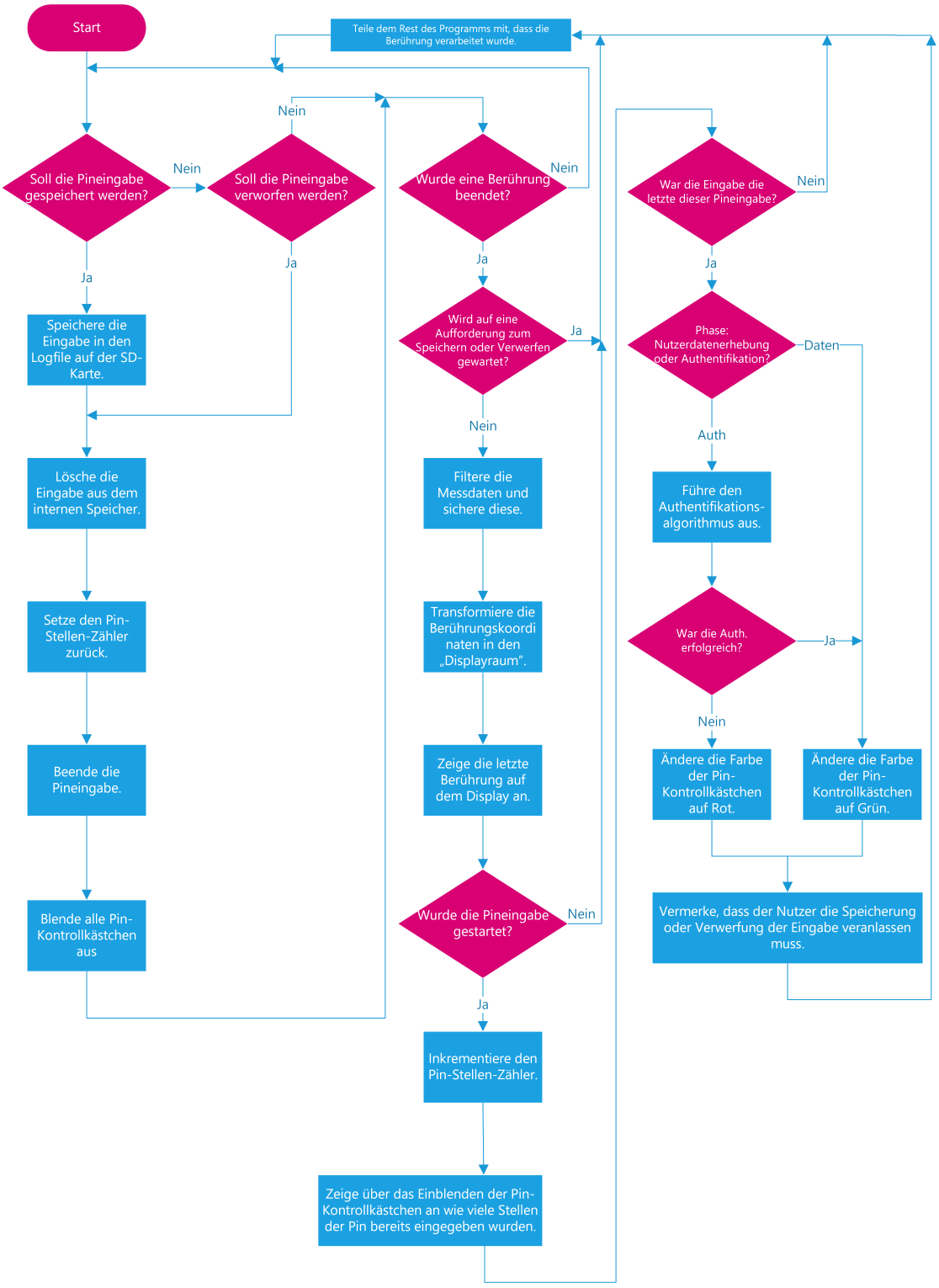

Abbildung 6.17: Programmablaufplan der Main-Loop

6.3.2.2 Timer Interrupt Service Routine

Die Timer-ISR wird bei Ablauf des Timers ausgeführt. In diesem Fall alle $7,5\mu s$. In ihr sind folgende Funktionalitäten realisiert:

- Falls der Touch_IP-Core sich in einem Fehlerzustand befindet, gebe eine Fehlermeldung über die Konsole aus.
- Falls der Touch_IP-Core neue Messwerte hat, lese diese ein und sichere diese. Bestätige den Empfang der Daten. Falls die Messung den Beginn einer neuen Berührung signalisiert ($Touch\_ID = 1$), fange an die Daten der letzten Berührung zu überschreiben.
- Falls eine Berührung beendet ($Touch\_ID = 2$) ist und sie lang genug war, teile dies der Main-Loop mit.
- Sollten bereits neue Messwerte zur Verfügung stehen, obwohl die Main-Loop die Verarbeitung der letzten Berührung noch nicht beendet hat, gebe eine Warnung auf der Konsole aus.

Der entsprechende Programmablaufplan kann in Abbildung 6.18 gefunden werden.

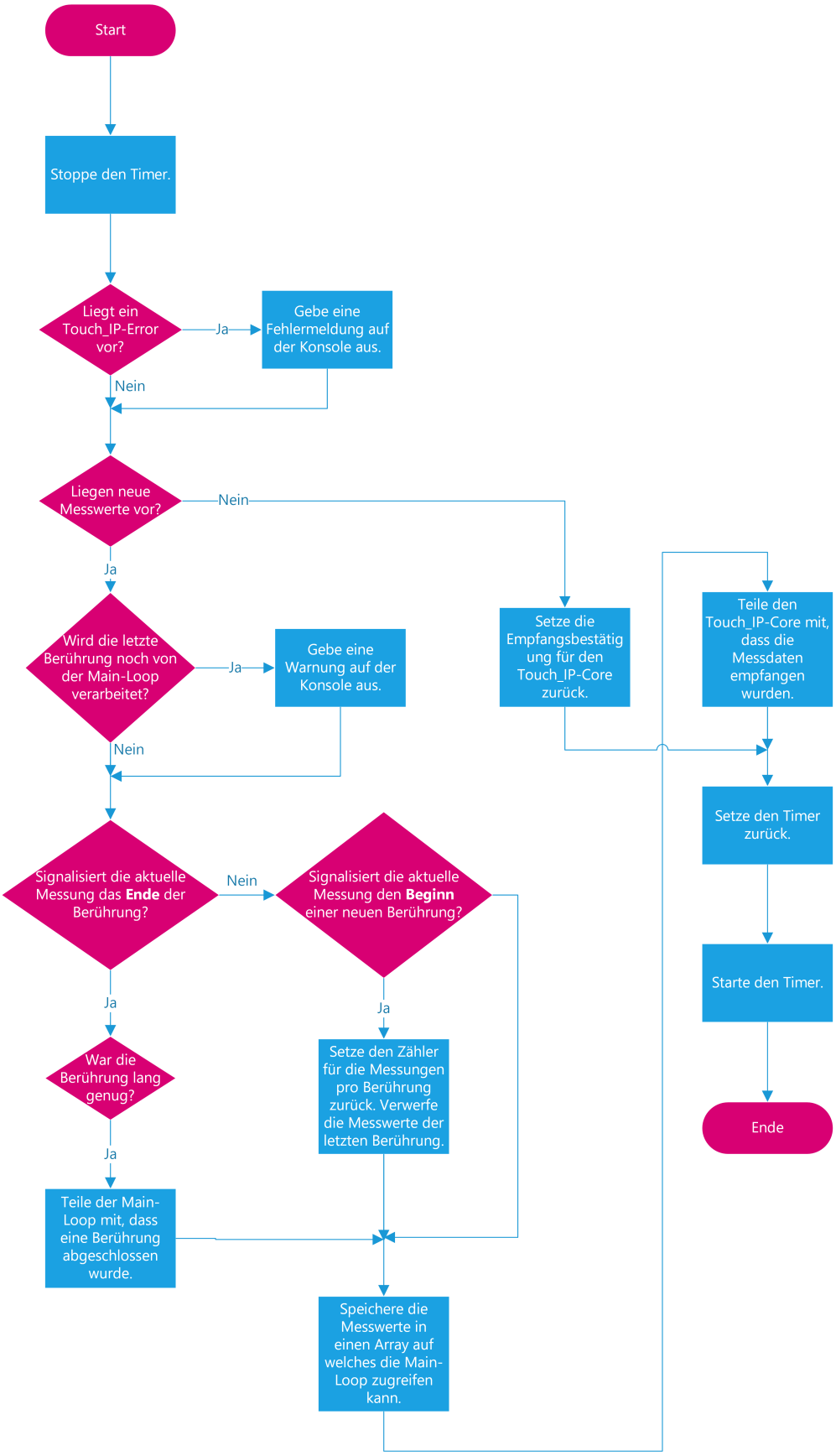

Abbildung 6.18: Programmablaufplan der Timer Interrupt Service Routine

6.3.2.3 Taster Interrupt Service Routine

Die Taster-ISR wird ausgeführt sobald einer der Taster gedrückt wird. Über diese ISR wird die Menüführung und Steuerung des Programms geregelt. Das Menü wird auf dem OLED-Display angezeigt. Dabei stehen Vier Zeilen zur Verfügung (Zeilen 0 bis 3). Ein Cursor zeigt an welche Zeile ausgewählt ist. Über die Taster *Oben* und *Unten* kann der Cursor bewegt und damit die Auswahl geändert werden. Da Vier Zeilen nicht reichen um alle Optionen darzustellen werden diese je nach Programmstatus verändert. Die Funktion der Taster *Links*, *Mitte* und *Rechts* hängen von der Position des Cursors und dem Programmstatus ab. Aufgeführt nach der ausgewählten Zeile können folgende Eingaben vom Nutzer betätigt werden:

- Zeile **0** ist unabhängig vom Programmstatus:
  - Es wird der Text „Select ACC“ angezeigt. Über die Taster *Links* und *Rechts* kann der Account ausgewählt werden.
- Zeile **1** ist unabhängig vom Programmstatus:
  - Es wird der Text „Select User“ angezeigt. Über die Taster *Links* und *Rechts* kann der Nutzer/User ausgewählt werden.
- Zeile **2** ist abhängig vom Programmstatus:
  - Wurde die Pineingabe noch nicht gestartet wird der Text „Start“ angezeigt. Durch betätigen des *mittleren* Tasters wird die Pineingabe gestartet.
  - Wurde die Pineingabe gestartet aber noch nicht als beendet erklärt wird der Text „Discard“ angezeigt. Durch betätigen des *mittleren* Tasters wird der Main-Loop mitgeteilt, dass die Pineingabe verworfen werden soll.
  - Wurde eine Pineingabe von der Main-Loop als beendet erklärt wird der Text „Save“ angezeigt. Durch betätigen des *mittleren* Tasters wird der Main-Loop mitgeteilt, dass die Pineingabe gespeichert werden soll.
- Zeile **3** ist abhängig vom Programmstatus:
  - Wurde eine Pineingabe von der Main-Loop als beendet erklärt wird der Text „Discard“ angezeigt. Durch betätigen des *mittleren* Tasters wird der Main-Loop mitgeteilt, dass die Pineingabe verworfen werden soll.

– Andernfalls wird kein Text dargestellt. Die Taster haben keine Funktion.

Nicht aufgeführte Taster sind in dem entsprechenden Menüpunkt ohne Funktion.

Der entsprechende Programmablaufplan kann in Abbildung 6.19 gefunden werden.

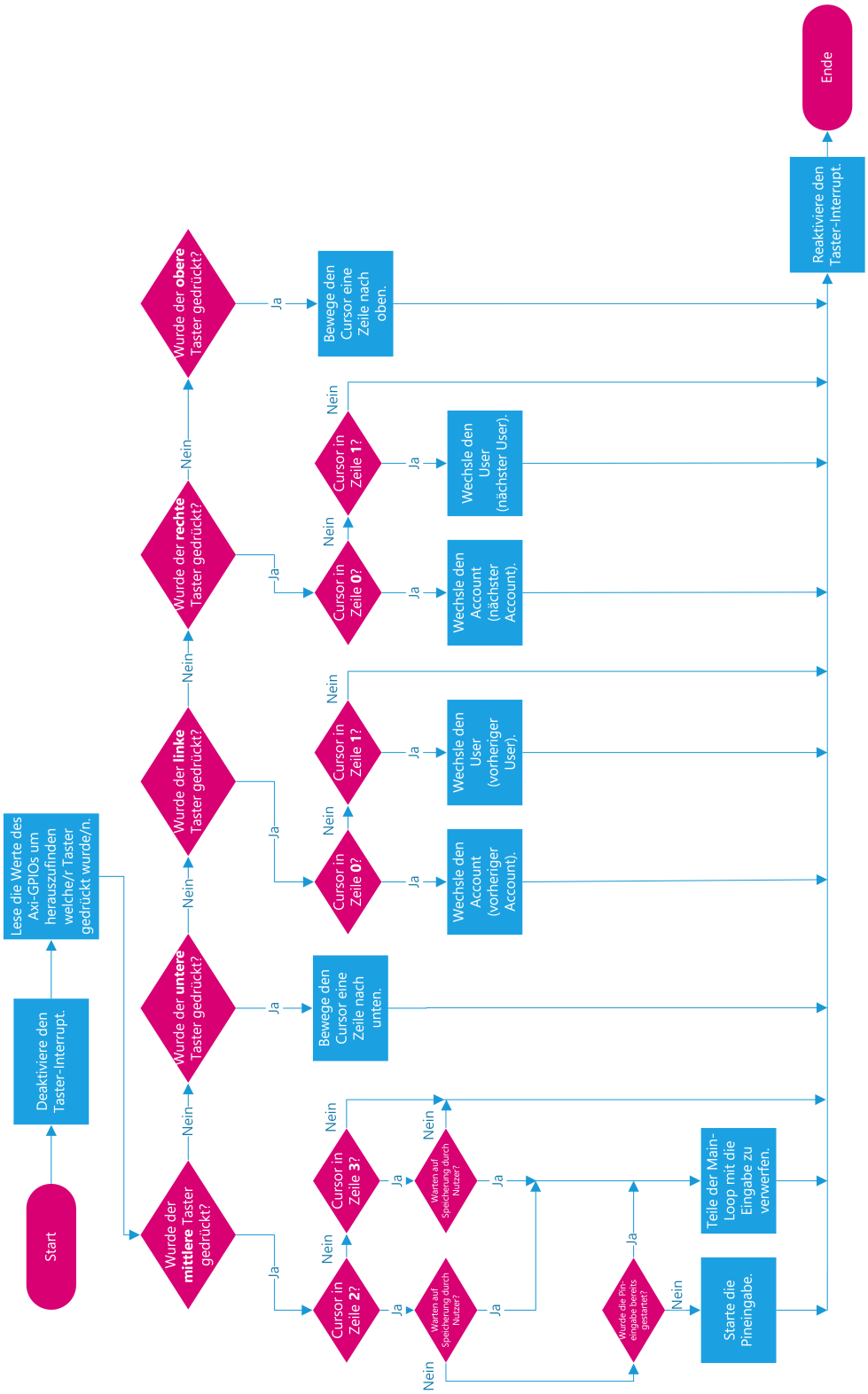

Abbildung 6.19: Programmablaufplan des Taster Interrupt Service Routine

6.4 Kalibrierung

Die Kalibrierung des Touchscreens umfasst zwei Unterthemen. Zunächst müssen die Messwerte, welche während einer Berührung alle 20ms erfasst werden, gefiltert werden. Danach ist eine Transformation der Werte auf die Dimensionen des Displays notwendig, um die Berührung auf diesem anzeigen zu können.

Das Finden eines geeigneten Filteralgorithmus und das Optimieren der Transformation wurde auf einem PC durchgeführt. Da während der Pineingabe nur Berührungen im Pinfeld erwartet werden, wurde die Kalibrierung auch nur für diesen Bereich durchgeführt. Hierfür wurden für 63 Kalibrierungspunkte jeweils 10 Berührungen aufgezeichnet. Die 63 Punkte wurden dabei in Form von Kreuzen auf dem Display angezeigt. Die Koordinaten sind dabei (x_k, y_k) wobei gilt $x_k \in \{308, 360, 412, 464, 516, 568, 620\}$ und $y_k \in \{30, 82, 134, 186, 238, 290, 342, 394, 446\}$. Die Berührungen wurden für eine erhöhte Genauigkeit mit einem Stift durchgeführt. Diese Messungen können in der Datei *MESPIN.txt* im Anhang A.5 gefunden werden. Die Auswertung dieser Daten erfolgt in dem Python-Skript *Calibration.py* welches sich ebenfalls im Anhang A.6 befindet.

6.4.1 Filterung

Das eine Filterung der Messewerte notwendig ist, wird durch die Abbildungen 6.20 und 6.21 deutlich. In diesen sind die ungefilterten Messungen der X- beziehungsweise Y-Werte für den Kalibrierungspunkte $(x_k, y_k) = (360, 82)$ dargestellt. Die verschiedenen Farben stehen dabei für jeweils eine der 10 Berührungen in diesem Punkt.

Zu erkennen ist, dass die Messwerte jeweils am Anfang und Ende einer Berührung stark vom Mittelwert abweichen. Es bietet sich also an, diese bei der Filterung einfach zu ignorieren. Auf Grundlage einer Anleitung zum Auswerten von Touchdisplays [13] wurde entschieden, über die restlichen Messwerte den Mittelwert eines lokalen Medians zu bilden.

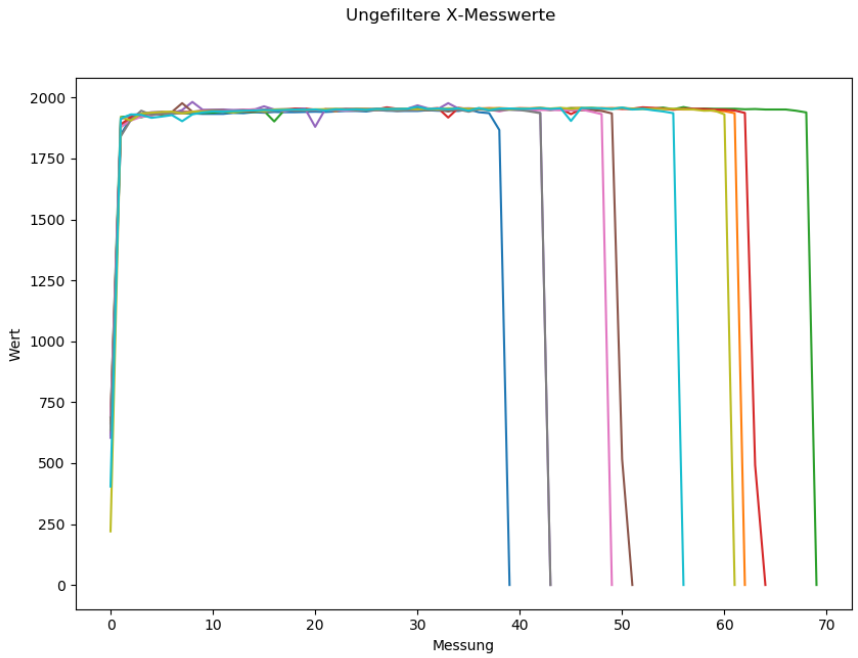

Abbildung 6.20: Ungefilterte X-Messwerte im Kalibrierungspunkt $(x_k, y_k) = (360, 82)$

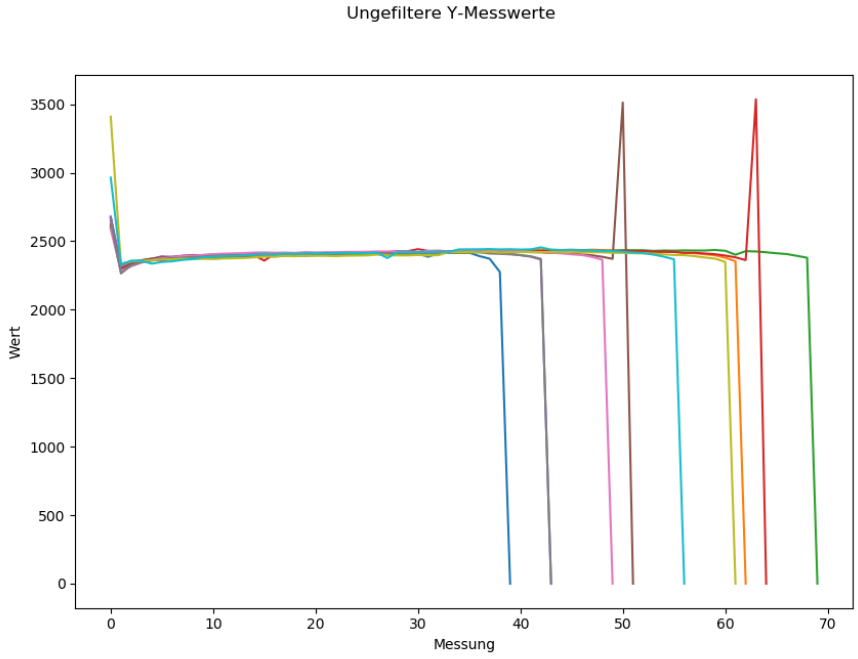

Abbildung 6.21: Ungefilterte Y-Messwerte im Punkt $(x_k, y_k) = (360, 82)$

Nach Untersuchungen der Messdaten wurde entschieden, jeweils die ersten und letzten vier Messwerte zu ignorieren und den lokalen Median über jeweils sechs Messwerte zu bilden.

Betrachten wir der Einfachheit halber nur die X-Werte würde dies für eine Berührung mit den X-Messwerten $x_{ungefiltert} = [0, 2, 5, 20, 40, 42, 42, 50, 43, 35, 41, 22, 3, 1, 0]$ folgendes bedeuten:

1. Ignoriere die ersten und letzten vier Werte.
   $x_{ignoriert} = [40, 42, 41, 50, 43, 35, 42]$
2. Bilde den Median über die sechs Elemente $x_{ingnoriert,0} \dots x_{ignoriert,5}$.
   $x_{m,0} = 41.5$
3. Bilde den Median über die sechs Elemente $x_{ingnoriert,1} \dots x_{ignoriert,6}$.
   $x_{m,1} = 42$
4. Bilde den Mittelwert über die in Punkt 2 und 3 errechneten Werte.
   $x_{gefiltert} = 41.75$

Dies bedeutet allerdings, dass eine Berührung mindestens 14 Messwerte enthalten beziehungsweise 280ms lang sein muss. Dies stellte sich während der Nutzerdatenerfassung in den meisten Fällen als gegeben heraus.

In Abbildung 6.22 sind die gefilterten Messwerte dargestellt. Bis auf wenige Ausreißer liegen die Werte für Berührungen an ein und demselben Kalibrierungspunkt relativ nah beisammen. Es wird jedoch auch deutlich, dass die Messwerte von den Koordinaten der eingestellten Kalibrierungspunkte abweichen. So gehört der untere rechte Punkt in Abbildung 6.22 beispielsweise zum Kalibrierungspunkt $(308, 30)$ und der obere linke zum Kalibrierungspunkt $(620, 446)$. Die Messwerte sind also zu den Kalibrierungspunkten sowohl gespiegelt als auch verzerrt. Möchte man nun aus den Messwerten die dazugehörige Position auf dem Display erfahren, müssen diese transformiert werden.

6.4.2 Transformation

Die Messwerte seien x_m und y_m und die dazugehörigen Werte auf dem Display als x_k und y_k. Über mehrere Iterationen wurde ein Algorithmus gesucht, welcher alle Punkte (x_m, y_m) mit möglichst geringen Fehler in die Kalibrierungspunkte (x_k, y_k) überführt.

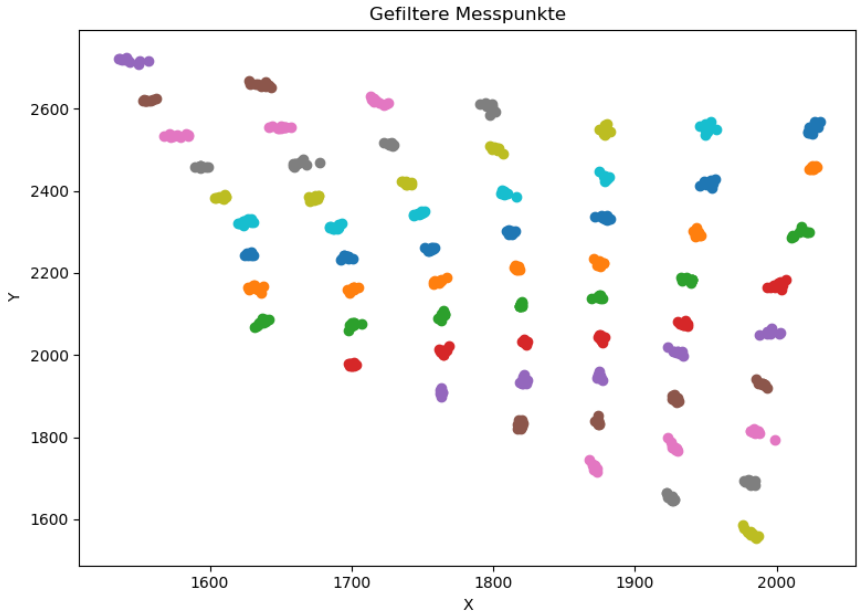

Abbildung 6.22: Messpunkte nach der Filterung.

Verwendet wurde ein Polynom, dessen genaue Form und Koeffizienten dem Python-Skript (s. Anh. A.5) oder der Datei *main.c* (s. Anh. A.4) entnommen werden können.

Ein Vergleich der transformierten Werte und der Sollwerte, also der Kalibrierungspunkte, kann in Abbildung 6.23 gefunden werden. Die durchschnittlichen Abweichungen von den Sollwerten betragen dabei 1.73 für die X- und 3.7 für die Y-Werte.

Nach der Implementierung auf dem Zedboard fällt auf, dass die Transformation bei der Bedienung des Touchscreens mit einem Stift oder Fingernagel gut funktioniert. Führt man die Berührung hingegen mit einem großflächigeren Objekt aus, weicht die errechnete Position teilweise stark von der tatsächlichen Position der Berührung ab. Eine mögliche Erklärung hierfür ist der mechanische Aufbau des Systems. Das Touchoverlay ist etwas größer als das Display und steht deswegen an einer Seite über. Erfolgt die Bedienung nun mit einer großen Fläche biegt sich das Touchoverlay in der Mitte durch. Dies könnte zu abweichenden Messwerten führen. Damit wären auch die transformierten Werte fehlerhaft. Für die Erfassung der Pineingabe wurden die gefilterten aber untransformierten Messwerte verwendet.

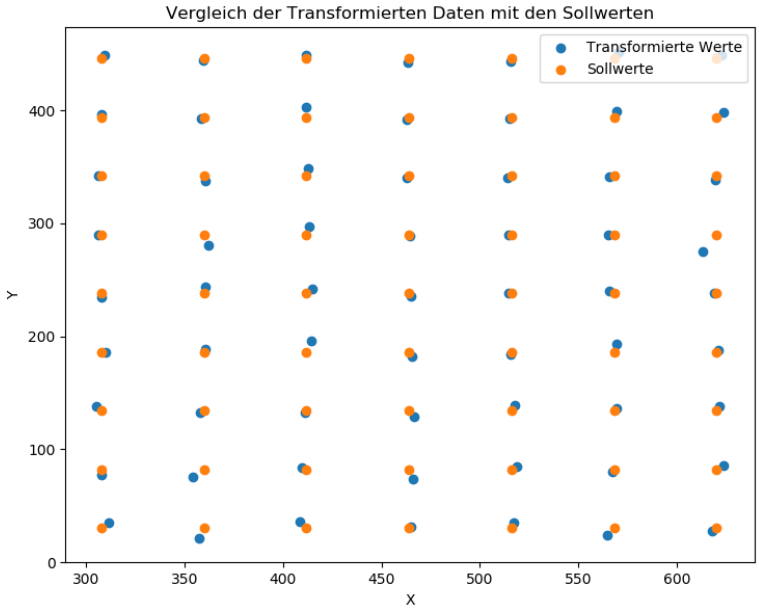

Abbildung 6.23: Vergleich der transformierten Messpunkte mit den Sollwerten

Die Einbeziehung des dritten Messwertes z_m könnte dabei helfen die Ergebnisse der Transformation zu verbessern, da dieser Rückschlüsse auf den Druck der Berührung ermöglicht. Aus zeitlichen Gründen wurde jedoch auf ein Verfolgen dieses Ansatzes verzichtet.

6.5 Klassifizierung

In diesem Abschnitt wird die Auswahl, das Training und die Validierung des Klassifikations- beziehungsweise Authentifikationsalgorithmus beschrieben. Hierzu wird zunächst der Prozess der Nutzerdatenerfassung beschrieben. Anschließend wird auf die Auswertung dieser Daten am PC eingegangen. Die Performance verschiedener Klassifikatoren wird verglichen. Abschließend wird einer dieser Klassifikatoren exportiert und die Implementierung des Verfahrens im PS des Zedboards wird beschrieben. Die Performance dieses Klassifikators wird in einem Feldtest überprüft und mit der auf dem PC erreichten verglichen.

6.5.1 Datenerfassung

Während der ersten Phase (s. Kap. 5.2.1) wurden von 16 Testpersonen die Daten der Pineingabe erfasst. Die Nutzer wurden vor der Eingabe darauf hingewiesen, welche Daten vor ihnen erfasst werden. Des Weiteren war ihnen bekannt, dass mit diesen Daten ein Klassifikations- beziehungsweise Authentifikationsalgorithmus trainiert werden soll. Es wurde ihnen nahe gelegt darauf zu achten ein spezifisches Eingabemuster zu verwenden und dies über die verschiedenen Eingaben hinweg beizubehalten.

Insgesamt wurden die Daten von 948 Pineingaben über einen Zeitraum von 11 Tagen erfasst. Dabei erfolgte die Erfassung der Daten zu unterschiedlichen Tageszeiten, jedoch immer zwischen 8 und 17 Uhr. Da die Testpersonen unterschiedlich viel Zeit für die Teilnahme hatten, wurden auch unterschiedlich viele Eingaben pro Person erfasst. Für die vierstellige Pin (**6649**) wurden insgesamt 388 Eingaben erfasst, für die sechsstellige (**283764**) 287 und für die achtstellige (**15948755**) 273. Eine Aufschlüsselung nach Testperson ist in Tabelle 6.8 zu sehen.

Pin	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
6649	51	18	22	26	5	23	14	4	5	25	44	38	60	18	17	18
283764	45	13	16	25	6	18	10	3	5	29	20	29	39	7	5	17
15948755	48	9	18	25	7	19	12	6	5	21	7	34	28	10	4	20

Tabelle 6.8: Anzahl der erfassten Pineingaben nach Nutzer und Pin. Die Spalten stehen dabei für die Nutzer, die Zeilen für die verwendete Pin.

Alle Nutzerdaten können in der Datei *USERD.tet* im Anhang A.6 gefunden werden.

6.5.2 Trainieren der Klassifikatoren in Python

Der Source-Code welcher für das Trainieren und Testen der Klassifikatoren verwendet wurde kann in der Datei *Classification.py* im Anhang A.6 gefunden werden. Für die Verarbeitung der Nutzerdaten und das Trainieren der Klassifikatoren wurde die *SciPy*-Bibliothek [32] verwendet. Für die Visualisierung der Ergebnisse wurde die *Seaborn*-Bibliothek [53] verwendet.

Das Training der Klassifikatoren erfolgt für die verschiedenen Pins separat. Sprich für das Trainieren eines Klassifikator auf den Account 0, also der Pin **6649**, und den User 1, werden nur die Eingaben aller Nutzer für diesen Pin verwendet. Das Eingabeverhalten der Nutzer bei den anderen Pins spielt keine Rolle.

Die einzelnen Merkmale einer Pineingabe haben teilweise sehr unterschiedliche Mittelwerte und Standartabweichungen. Um optimale Ergebnisse zu erzielen, ist es sinnvoll die einzelnen Merkmale zu normalisieren. Dabei werden die Merkmale auf einen Mittelwert von 0 und eine Standartabweichung von 1 skaliert. Es sei $\overline{M_n}$ das Arithmetische Mittel des n-ten Merkmals der Eingabe und $\sigma(M_n)$ die Standardabweichung des n-ten Merkmals. Dann ergibt sich das normalisierte n-te Merkmal als $M_{n,norm} = (M_n - \overline{M_n}) \cdot \frac{1}{\sigma(M_n)}$.

Um die trainierten Klassifikatoren testen zu können, müssen die Nutzerdaten in ein Trainings- und ein Testset aufgeteilt werden. In diesem Fall wurden 75% der Pineingaben zum Training und 25% zum Testen verwendet. Diese Aufteilung erfolgt zufällig und hat keinen unwesentlichen Einfluss auf das Ergebnis. Daher werden alle Klassifikatoren 200 Mal trainiert und getestet. Zur Bewertung der Klassifikatoren werden Recall und Precision verwendet. Damit dieser Prozess sinnvoll funktioniert ist eine gewisse Mindestanzahl an Eingaben erforderlich. Da es hierfür keine harte Grenze gibt, wurde sie auf 25 Eingaben pro Nutzer und Pin festgelegt. Für die Pin **6649** wurden die Klassifikatoren also nur für 6 Nutzer trainiert und getestet (vgl. Tab. 6.8).

Zunächst wurden die Klassifikatoren sowohl mit den Daten des jeweiligen Nutzers, als auch mit Daten der potentiellen Angreifer trainiert. Dabei wurde jedoch nur zwischen „Nutzer“ und „Nicht Nutzer“ unterschieden. Eine Unterscheidung der Angreifer findet nicht statt.

Zum Vergleich wurden die Fünf in Kapitel 3.4.2 vorgestellten Klassifizierungsalgorithmen trainiert. Die genauen Einstellungen der Klassifikatoren können dem Python-Skript entnommen werden.

In den Tabellen 6.9, 6.10 und 6.11 sind der durchschnittliche Recall und Precision pro User und Algorithmus für die drei verschiedenen Pins aufgeführt. Dabei fällt auf, dass es zwischen den verschiedenen Nutzern starke Unterschiede gibt. Betrachten wir beispielsweise die Performance des k-NN-Klassifikators welcher mit den Eingaben für den Pin **6649** trainiert wurde (s. Tab. 6.9). Während dieser Klassifikator den Nutzer 1 in 97% der Fälle richtig erkennen würde, ist dies für Nutzer 10 in nur 47% der Fall. Dieser Trend lässt sich auch bei den anderen Algorithmen und Pins erkennen. Es gibt Nutzer bei denen fast alle Algorithmen eine sehr gute Performance liefern (zum Beispiel Nutzer 1 und 11) und Nutzer bei denen alle Algorithmen eine schlechtere Performance liefern (zum Beispiel Nutzer 10 oder 12). Im allgemeinen hängt die Performance der Algorithmen von der Einzigartigkeit des Eingabeverhaltens ab. Aber auch die Konstanz des Eingabeverhaltens über verschiedene Eingaben spielt eine Rolle. Wird die Pin jedes Mal auf eine unterschiedliche Art und Weise eingeben, ist es für die Algorithmen schwerer gute Entscheidungsgrenzen zu ziehen.

Auch die Länge der Pineingabe hat einen Einfluss auf die Performance der Klassifikatoren. In Tabelle 6.12 wurde der Mittelwert über alle Nutzer bei den verschiedenen Pins gebildet. Bis auf wenige Ausnahmen steigt sowohl der Recall, als auch die Precision bei allen Algorithmen mit der Länge der verwendeten Pin. Der Einfluss der Länge ist jedoch geringer als erwartet. Auch mit einer vierstelligen Pin schneiden die Klassifikatoren k-NN und SVM, mit einem Recall und einer Precision von über 70% beziehungsweise 80%, gut ab. In der letzten Zeile dieser Tabelle sind die durchschnittlichen Werte über alle Pineingaben aller Nutzer aufgeführt. Auch in diesem Fall bilden der k-NN-Klassifikator und die SVM das Spitzenfeld. Der Random-Forest-Klassifikator bietet zwar eine gute Precision jedoch einen schlechten Recall. Ein rechtmäßiger Nutzer würde also nur in 57% der Fälle erkannt werden. Im Gegensatz dazu würde der Naive-Bayes-Klassifikator einen rechtmäßigen Nutzer in 90% der Fälle richtig erkennen. Die Eingaben, welche als berechtigter Nutzer erkannt wurden, stammten jedoch in Wirklichkeit nur in 52% der Fälle von dem jeweilig berechtigten Nutzer. Der Decision-Tree-Klassifikator ist ausgeglichener was Recall und Precision angeht, jedoch in beiden schlechter als der k-NN-Klassifikator und die SVM.

	Decision Tree		Random Forest		k-NN		Naive Bayes		SVM	
Nutzer	R	P	R	P	R	P	R	P	R	P
1	1.00	0.98	0.97	1.0	0.97	0.94	1.0	0.94	0.99	0.90
4	0.46	0.59	0.39	0.8	0.74	0.79	0.93	0.32	0.89	0.53
10	0.39	0.52	0.34	0.81	0.42	0.96	0.91	0.36	0.79	0.72
11	0.90	0.86	0.89	0.98	0.86	0.89	0.89	0.43	0.97	0.85
12	0.54	0.59	0.34	0.86	0.62	0.81	0.57	0.51	0.71	0.7
13	0.69	0.79	0.56	0.90	0.76	0.75	0.88	0.35	0.88	0.55

Tabelle 6.9: Durchschnittliche Recall (R) und Precision (P) pro Nutzer und Algorithmus für Account 0 beziehungsweise Pin 6649

	Decision Tree		Random Forest		k-NN		Naive Bayes		SVM	
Nutzer	R	P	R	P	R	P	R	P	R	P
1	0.97	0.96	0.97	1.00	0.98	0.95	1.00	0.88	1.00	0.92
4	0.69	0.71	0.41	0.93	0.82	0.81	1.00	0.41	0.95	0.69
10	0.68	0.75	0.66	0.95	0.66	0.90	0.95	0.47	0.86	0.86
12	0.51	0.54	0.33	0.88	0.64	0.85	0.90	0.19	0.81	0.57
13	0.62	0.70	0.50	0.95	0.72	0.78	0.89	0.30	0.92	0.65

Tabelle 6.10: Durchschnittliche Recall (R) und Precision (P) pro Nutzer und Algorithmus für Account 1 beziehungsweise Pin 283764

	Decision Tree		Random Forest		k-NN		Naive Bayes		SVM	
Nutzer	R	P	R	P	R	P	R	P	R	P
1	0.98	0.97	0.98	1.00	0.98	0.98	1.00	0.98	1.00	1.00
4	0.77	0.84	0.74	1.00	0.98	0.89	0.93	0.87	0.98	0.94
12	0.45	0.54	0.20	0.79	0.69	0.75	0.77	0.33	0.77	0.65
13	0.68	0.76	0.35	0.91	0.76	0.88	0.80	0.3	0.79	0.89

Tabelle 6.11: Durchschnittliche Recall (R) und Precision (P) pro Nutzer und Algorithmus für Account 2 beziehungsweise Pin 15948755

	Decision Tree		Random Forest		k-NN		Naive Bayes		SVM	
Account	R	P	R	P	R	P	R	P	R	P
0	0.66	0.71	0.58	0.89	0.73	0.86	0.86	0.49	0.87	0.71
1	0.69	0.73	0.57	0.94	0.76	0.86	0.95	0.45	0.91	0.74
2	0.72	0.77	0.57	0.93	0.85	0.88	0.88	0.62	0.89	0.87
	0.69	0.74	0.57	0.92	0.78	0.86	0.9	0.52	0.89	0.77

Tabelle 6.12: Durchschnittliche Recall (R) und Precision (P) über alle Nutzer pro Account und Algorithmus. Durchschnittswerte über alle Accounts in der letzten Zeile.

In den Abbildungen 6.24 und 6.25 ist die zuvor erwähnte Abhängigkeit der Performance von der Aufteilung der Daten in Test- und Trainingsdaten veranschaulicht[5]. Die entsprechenden Grafiken für alle weiteren Algorithmen und Eingabelängen können im Anhang A.7 gefunden werden.

Die in Kapitel 4.6 angesprochene Novelty und Outlier Erkennung konnte aufgrund der begrenzten Zeit nicht genauer verfolgt werden. Erste Untersuchungen zeigen, dass diese Verfahren auch funktionieren, jedoch schlechter als die vorgestellten Klassifikatoren. Eine Funktion welche drei Algorithmen dieser Problemstellung vergleicht ist ebenfalls im Python-Skript implementiert.

[5]Die Box-Plots wurden mit Hilfe von Seaborn [53] erstellt. Ein Boxplot oder auch Box-Whisker-Plot genannt stellt die Verteilung eines Datensatzes über fünf festgelegte Punkte (Minimum, Maximum, Median, unteres und oberes Quartil) dar. Der Kasten erstreckt sich vom unteren bis zum oberen Quartil und wird an der Stelle des Median von einer Linie geteilt. Die zwei Striche am oberen und unteren Ende stehen für das Maximum beziehungsweise das Minimum. Ausgeschlossen hiervon sind Datenpunkte welche von dem Algorithmus als Ausreißer erkannt wurden. Diese werden separat dargestellt. In diesem Fall als Raute. [55]

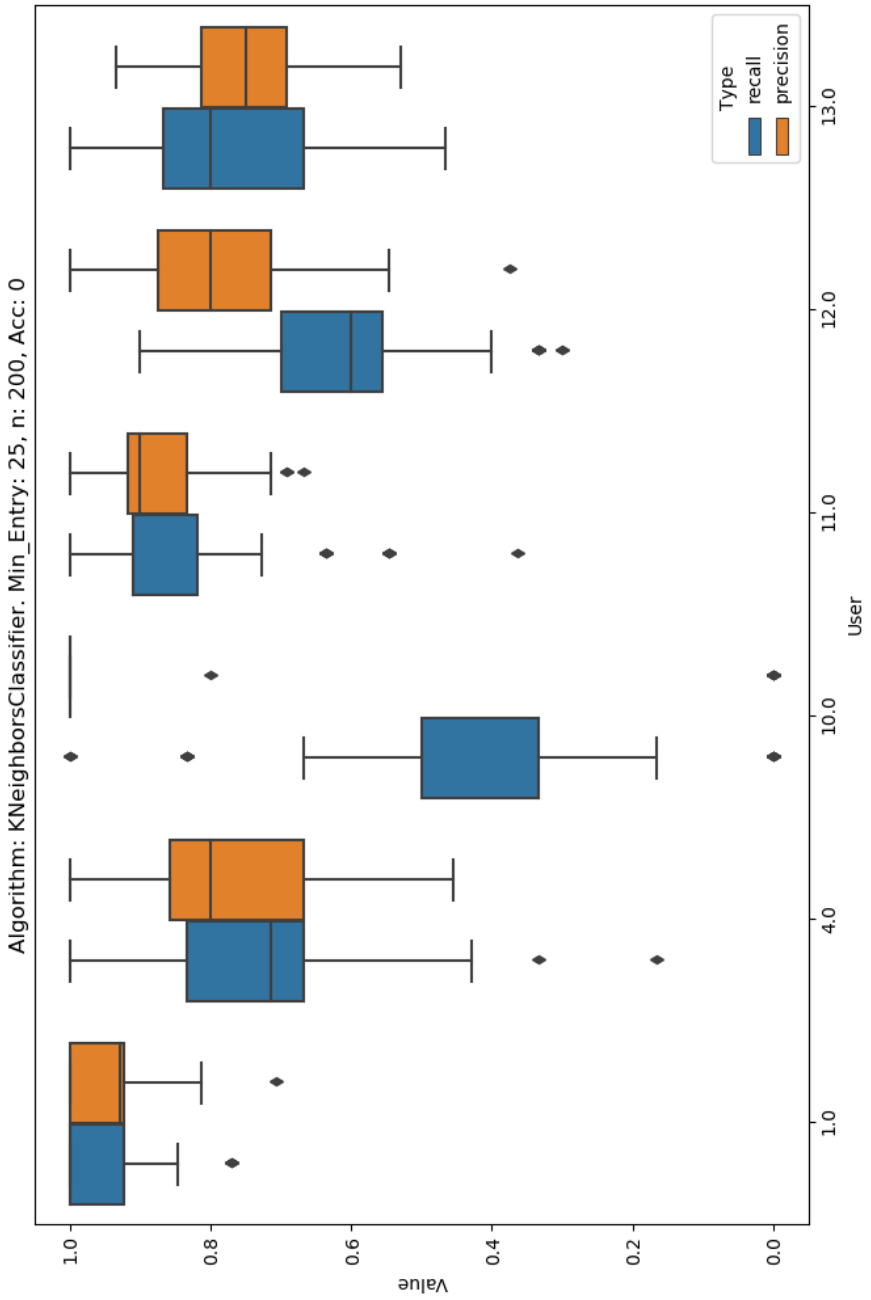

Abbildung 6.24: Box-Plot der Precision und Recall aller 200 Durchgänge für den k-NN-Klassifikator (verwendete Pin: **6649**)

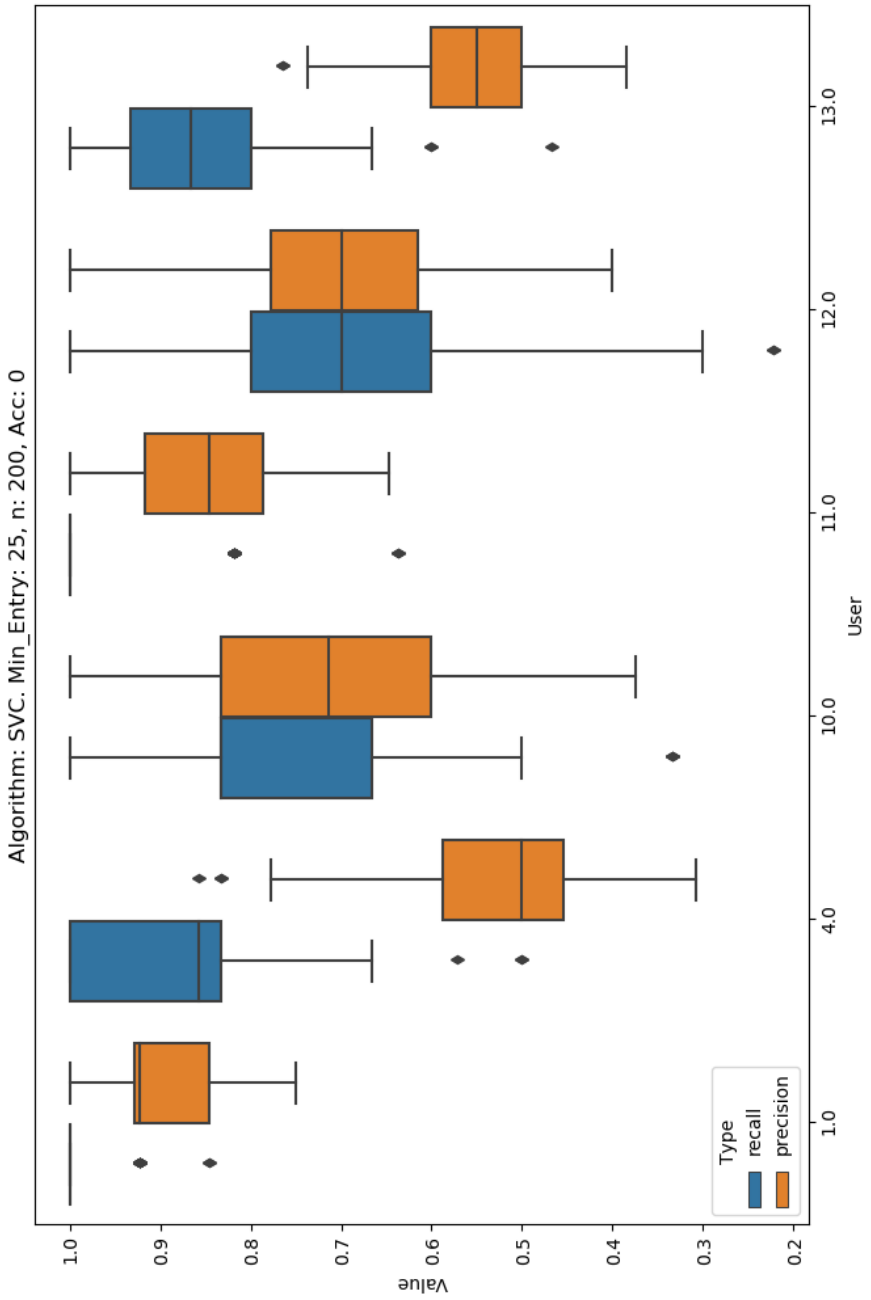

Abbildung 6.25: Box-Plot der Precision und Recall aller 200 Durchgänge für die SVM (verwendete Pin: **6649**)

6.5.3 Implementierung der Klassifikatoren auf dem Zedboard

Um die in Python trainierten Klassifikatoren auf dem Zedboard ausführen zu können sind zwei Schritte notwendig:

- Der vom Klassifikator ausgeführte Algorithmus und alle Einstellungen sowie die trainierten Parameter müssen in C-Code überführt werden.
- Der für die Normalisierung der Merkmale verwendete Algorithmus sowie die verwendeten Mittelwerte und Standardabweichungen müssen in C-Code überführt werden.

Zum Überführen der Klassifikatoren wurde die Bibliothek *Sklearn-Porter* [39] verwendet. Mit dieser ist es unter anderem möglich über einen Funktionsaufruf eine trainierte SVM in einen Lauffähigen C-Code überführen zu lassen. Der Quellcode des überführten Klassifikators kann in der Datei *Classification.c* im Anhang A.4 gefunden werden. Der Export der Koeffizienten und die Entwicklung des Algorithmus für die Normalisierung der Merkmale erfolgte manuell. Die entsprechende Funktion mit dem Namen *normalize()* ist ebenfalls in der Datei *Classification.c* implementiert und wird vor dem Anwenden des Authentifikationsalgorithmus auf die Daten der Pineingabe angewandt.

Für das Trainieren der zu exportierenden Klassifikatoren wurden die Pineingaben nicht in Trainings- und Testdaten aufgeteilt Es wurden alle Daten zum Trainieren verwendet.

6.5.4 Validierung der Klassifikatoren im Feldtest

Aufgrund der limitierten Zeit, welche zur Verfügung stand, wurde nur einer der Klassifikatoren im Feldtest untersucht. Es wurde entschieden die SVM welche auf den Nutzer 1 und den Account 0 (also der Pin **6649**) trainiert wurde zu testen.

In der Simulation erreicht dieser Klassifikator einen Recall von 99% und eine Precision von 90%.

Der Feldtest wurde mit 12 Angreifern durchgeführt und bestand aus 2 Teilphasen. Teilweise nahmen einige der Testpersonen auch schon an der Phase der Nutzerdatenerfassung teil. Die Nummerierung der Teilnehmer aus Tabelle 6.8 wurde beibehalten.

Den Teilnehmern wurde während des gesamten Versuches die einzugebende Pin angezeigt. Des Weiteren wurde ihnen mitgeteilt, wie der Algorithmus funktioniert und welche

Merkmale er verwendet um zu entscheiden, ob es sich bei dem Eingebenden um den berechtigten Nutzer handelt oder nicht. Sie wurden dazu angehalten bei abgelehnten Eingaben ihr Eingabeverhalten zu variieren. Jeder der Teilnehmer hatte 20 Versuche den Pin einzugeben. Nach den ersten 10 Versuchen wurde ihnen die Möglichkeit gegeben zu beobachten, wie der berechtigte Nutzer (Nutzer 1) die Pin eingibt. Die Eingabe durch Nutzer 1 konnten die Testpersonen beliebig oft beobachten. Es wurde also untersucht, ob die Testpersonen mit bestmöglichen Voraussetzungen in der Lage sind den Authentifikationsalgorithmus zu umgehen. Um zu überprüfen ob der Algorithmus den berechtigten Nutzer richtig erkennt, wurden auch von Nutzer 1 20 Eingaben vorgenommen.

Die Ergebnisse des Feldtests sind in Tabelle 6.13 aufgeführt. Damit ergibt sich ein Recall von 95% und eine Precision von 53%. Beziehungsweise eine False Rejection Rate (FRR) von 5% und eine False Acceptance Rate (FAR) von 7%.

Die schlechte Precision hängt mit der Tatsache zusammen, dass es wesentlich mehr Eingaben von Angreifern gab, als vom berechtigten Nutzer. Nehmen wir an, der Recall würde bei mehr Eingaben durch Nutzer 1 konstant bleiben, dann wäre bei 240 Eingaben von Nutzer 1 mit 228 erfolgreichen Versuchen zu rechnen. In diesem Fall würde es gleich viele Eingaben von Angreifern und dem berechtigten Nutzer geben. Die Precision wäre in diesem Fall 93%.

Damit schneidet dieser Klassifikator im Feldtest etwas schlechter ab als in der Simulation. Eine mögliche Ursache ist, dass im Feldtest weniger Versuche erfolgt sind als in der Simulation. Einen nicht unwesentlichen Einfluss dürfen auch die Testbedingungen haben. In der Simulation wurden die Eingaben des Nutzers mit denen von *potentiellen Angreifern* verglichen. Diese *potentiellen Angreifer* versuchten jedoch, im Gegensatz zum Feldtest, nicht das Eingabeverhalten des Nutzers nachzuahmen. Wie zuvor erwähnt, wurden die Testpersonen in der Phase der Nutzerdatenerfassung angehalten, ihr eigenes spezifisches Eingabeverhalten zu verwenden.

Es fällt auf, dass nur einer der Angreifer (Nutzer 22) es in den ersten 10 Versuchen geschafft hat, sich erfolgreich als berechtigter Nutzer auszugeben. Ohne Kenntnis über das Eingabeverhalten des berechtigten Nutzers, ist es also scheinbar sehr schwer, dieses gut genug nachzuahmen. Mit entsprechend vielen Versuchen wäre es vermutlich dennoch möglich, das erwartete Eingabeverhalten durch Ausprobieren zu finden. In der Praxis würde ein Account nach einer bestimmten Anzahl an Fehlversuchen jedoch mit Sicherheit gesperrt werden.

Nutzer	Erfolgreiche Versuche (absolut)	Erfolgreiche Versuche (prozentual)	Erster Erfolgreicher Versuch
1	19	0.95	1
2	2	0.10	16
3	0	0.00	-
10	1	0.05	12
14	1	0.05	14
17	4	0.20	12
18	0	0.00	-
19	0	0.00	-
20	0	0.00	-
21	1	0.05	14
22	5	0.25	5
23	2	0.10	19
24	1	0.05	13

Tabelle 6.13: Ergebnisse der Validierung der SVM (trainiert auf Nutzer 1, Account 1) im Feldtest

Mit konkreter Kenntnis des vom Klassifikator erwarteten Eingabeverhaltens konnten sich 66% der Angreifer erfolgreich als berechtigter Nutzer ausgeben. Nutzer 22, welcher auch die meisten erfolgreichen Angriffe durchführen konnte, gab an sich das Eingabeverhalten von Nutzer 1 aus der Phase der Nutzerdatenerfassung erinnert zu haben. Außerdem gab er an die Gespräche der anderen Angreifer gehört zu haben. So hatte er bereits Vorkenntnisse über das Eingabeverhalten, welches vom Klassifikator erwartet wird. Dies verdeutlicht, dass es mit entsprechender Kenntnis über das erwartete Eingabeverhalten durchaus möglich ist, dieses zu reproduzieren.

7 Fazit und Ausblick

Im Rahmen dieser Arbeit wurde gezeigt, dass sich das Eingabeverhalten einer Pin an einem Touchscreen durchaus für eine psychometrische Authentifikation eignet. Als Basis dieser Authentifikation wurden verschiedene Klassifizierungsalgorithmen getestet und verglichen. Von diesen Algorithmen wurde die Support Vector Machine ausgewählt und auf einem SoC-Entwicklungsboard implementiert. Ein Feldtest hat gezeigt, dass dieser Klassifizierungsalgorithmus in 95% der Fälle in der Lage ist, den Nutzer richtig zu erkennen. Des Weiteren konnten Angreifer, obwohl sie Kenntnis über die Funktionsweise des Algorithmus, die verwendete Pin und das Eingabeverhalten des Nutzers hatten, sich nur in 7% der Fälle erfolgreich authentifizieren. Das Kopieren des Eingabeverhaltens einer Person ist dementsprechend keine triviale Aufgabe. Ohne die Kenntnis über das Eingabeverhalten konnte im Feldtest kein erfolgreicher Angriff durchgeführt werden.

Auch wenn im Feldtest ein Angriff ohne Wissen über das Eingabeverhalten nicht möglich war, so zeigt die Simulation jedoch, dass es sehr wohl Fälle gibt in denen die Klassifizierungsalgorithmen Nutzer nicht eindeutig anhand des Eingabeverhaltens auseinander halten können. Die alleinige Verwendung des Eingabeverhaltens für eine Authentifikation ist demnach nicht empfehlenswert. Als zusätzliches Merkmal bietet es jedoch eine erhebliche Steigerung der Sicherheit ohne das sich der Nutzer mehr Informationen merken muss.

Es wurde ein nicht unwesentlicher Teil der zur Verfügung stehenden Zeit für die Entwicklung des Touchscreen-Systems verwendet. Dennoch kann die Genauigkeit und Verlässlichkeit des Touchscreens nicht mit kommerziellen Lösungen mithalten. Die Verwendung eines solchen Touchscreens könnte die Qualität der bei der Pineingabe erfassten Messdaten verbessern. Eine solche Verbesserung der Nutzerdaten könnte dabei helfen die Performance der Klassifizierungsalgorithmen zu steigern.

Bei Verzicht auf das eigenständige Entwicklen des Touch- und Display-Systems hätten die Klassifizierungsalgorithmen tiefgründiger behandelt werden können. Es wurden fünf

verschiedene Klassifikatoren getestet. Es ist jedoch fraglich, ob für jeden der Klassifikatoren die optimalen Einstellungen gefunden wurden. Zur Bewertung wurden der Recall und die Precision der jeweiligen Klassifikatoren erfasst. Für eine bessere Vergleichbarkeit mit ähnlichen Arbeiten wäre es jedoch sicherlich sinnvoll gewesen, auch die FAR und die FRR für jeden Algorithmus zu erfassen.

Eine weitere potenzielle Möglichkeit die Performance des Authentifikationsalgorithmus zu verbessern wäre das Zusammenschalten verschiedener Klassifikatoren. In diesem Fall könnte die Entscheidung über ein Mehrheitsvotum der einzelnen Klassifikatoren fallen. Eventuell könnten dadurch die Schwächen der einzelnen Klassifikatoren ausgeglichen werden um zu einem besseren Ergebnis zu gelangen.

Um aussagekräftigere Ergebnisse über die Performance der Algorithmen zu erlangen, wäre es sinnvoll einen ausführlicheren Feldtest durchzuführen. Die Erfassung von Nutzerdaten zu verschiedenen Tageszeiten und in verschiedenen Situationen, das Implementieren verschiedener Algorithmen für verschiedene Nutzer und das Erstellen von verschiedenen Testgruppen, welche unterschiedlich viele Informationen über den Algorithmus oder das erwartete Eingabeverhalten erhalten, sind Möglichkeiten die hier untersuchte Eignung genauer zu bestimmen.

Literaturverzeichnis

[1] Adafruit: *HDMI 7″ Display Backpack - Without Touch*. – URL https://www.adafruit.com/product/2406. – Zugriff: 14.01.2019

[2] Adafruit: *Resistive Touchscreen Overlay*. – URL https://www.adafruit.com/product/1676. – Zugriff: 14.01.2019

[3] Advantiv ; Devices, Analog: *ADV7511 Low-Power HDMI Transmitter with Audio Return Channel HARDWARE USERS GUIDE*. – URL https://www.analog.com/media/en/technical-documentation/user-guides/ADV7511_Hardware_Users_Guide.pdf. – Zugriff: 13.04.2019

[4] Anil, Jain ; Arun, Ross ; Salil, Prabhakar: An Introduction to Biometric Recognition. In: *IEEE Transactions on Circuits and Systems for Video Technology, Special Issue on Image- and Video-Based Biometrics* (2004), Janauar, Nr. 1. – URL https://researchweb.iiit.ac.in/~vandana/PAPERS/BASIC/intro.pdf. – Zugriff: 11.03.2019

[5] Antal, Margit ; Szabo, Laszlo Z.: *An evaluation of one-class and two-class classification algorithms for keystroke dynamics authentication on mobile devices*. – URL https://www.researchgate.net/profile/Margit_Antal/publication/279054322_An_Evaluation_of_One-Class_and_Two-Class_Classification_Algorithms_for_Keystroke_Dynamics_Authentication_on_Mobile_Devices. – Zugriff: 13.03.2019

[6] Arm: *AMBA AXI and ACE Protocol Specification*. – URL http://www.gstitt.ece.ufl.edu/courses/fall15/eel4720_5721/labs/refs/AXI4_specification.pdf. – Zugriff: 08.04.2019

[7] Arun, Ross ; Jain, Anil: Multimodal Biometrics: An Overview. In: *European Signal Processing Conference* 12 (2004), September. – URL https://ieeexplore.ieee.org/abstract/document/7080214/. – Zugriff: 11.03.2019

[8] Asiri, Sidath: *Machine Learning Classifiers*. – URL https://towardsdatascience.com/machine-learning-classifiers-a5cc4e1b0623?gi=2c3005c32c3c. – Zugriff: 17.03.2019

[9] Avnet: *ZedBoard*. – URL http://zedboard.org/product/zedboard. – Zugriff: 17.03.2019

[10] AVNET: *ZedBoard Getting Started Guide*. – URL http://zedboard.org/sites/default/files/documentations/GS-AES-Z7EV-7Z020-G-V7-1.pdf. – Zugriff: 12.01.2019

[11] AVNET: *ZedBoard Hardware User's Guide*. – URL http://zedboard.org/sites/default/files/documentations/ZedBoard_HW_UG_v2_2.pdf. – Zugriff: 05.03.2019

[12] Balagani, Kiran S. ; Phoha, Vir V. ; Ray, Asok ; Phoha, Shashi: On the discriminability of keystroke feature vectors used in fixed text keystroke authentication. In: *Elsevier* (2009). – URL https://www.mne.psu.edu/ray/journalAsokRay/2011/226Balagani_Kystroke11.pdf. – Zugriff: 13.03.2019

[13] Beer, Daniel: *Touch-screen filtering*. – URL https://dlbeer.co.nz/articles/tsf.html. – Zugriff: 15.01.2019

[14] Bhalla, M. ; Bhalla, A.: Comparative Study of Various Touchscreen Technologies. In: *International Journal of Computer Applications (0975 - 8887)* (2010), September, Nr. 6. – URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.206.5024&rep=rep1&type=pdf. – Zugriff: 12.01.2019

[15] Bronshrein, Adi: *A Quick Introduction to K-Nearest Neighbors Algorithm*. – URL https://medium.com/@adi.bronshtein/a-quick-introduction-to-k-nearest-neighbors-algorithm. – Zugriff: 18.03.2019

[16] Buschek, Daniel ; Luca, Alexander D. ; Alt, Florian: Improving Accuracy, Applicability and Usability of Keystroke Biometrics on Mobile Touchscreen Devices. (2015). – URL http://www.mmi.ifi.lmu.de/pubdb/publications/pub/buschek2015chi/buschek2015chi.pdf. – Zugriff: 13.03.2019

[17] Canvys: *Touch-Technologien.* – URL https://www.canvys.de/produkte/touch-technologien/. – Zugriff: 13.01.2019

[18] Cao, Kai ; Jain, Anil: Hacking Mobile Phones Using 2D Printed Fingerprints. In: *Department of Computer Science and Engineering Michigan State University* (2016). – URL https://pdfs.semanticscholar.org/0dc9/39b07c4deb8ab79a04ca3f2fe77b0b3737c6.pdf. – Zugriff: 12.03.2019

[19] Centric, Embedded: *Zynq SoC Training.* – URL https://web.archive.org/web/20170223123601/https://embeddedcentric.com/zynq-training-course/. – Zugriff: 12.01.2019

[20] Debnath, Bhattacharyya ; Rahul, Ranjan ; Farkhod, Alisherov ; Choi, Minkyu: Biometric Authentication: A Review. In: *International Journal of u- and e- Service, Science and Technology* 2 (2009), September, Nr. 3. – URL https://www.researchgate.net/profile/Debnath_Bhattacharyya/publication/46189709_Biometric_Authentication_A_Review/links/09e4150ff1c2ef3463000000/Biometric-Authentication-A-Review.pdf. – Zugriff: 11.03.2019

[21] Dhaker, P.: *Introduction to SPI Interface.* – URL https://www.analog.com/en/analog-dialogue/articles/introduction-to-spi-interface.html. – Zugriff: 14.01.2019

[22] Digilent: *Digilent Pmod Interface Specification.* 2011. test: . – URL https://www.digilentinc.com/Pmods/Digilent-Pmod_%20Interface_Specification.pdf. – Zugriff: 15.01.2019

[23] Donges, Niklas: *The Random Forest Algorithm.* – URL https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd. – Zugriff: 18.03.2019

[24] Duncan, Richard: An Overview of Different Authentification Methods and Protocols. (2001), Oktober. – URL https://www.moreilly.com/CISSP/Dom3-1-an_overview_of_different_authent.pdf. – Zugriff: 11.03.2019

[25] Eiband, Malin ; Khamis, Mohamed ; Zezschwitz, Emanuel von ; Hussmann, Heinrich ; Alt, Florian: Understanding Shoulder Surfing in the Wild: Stories from Users and Observers. In: *CHI Conference on Human Factors in Computing Systems,*

Denver (2018). – URL http://eprints.gla.ac.uk/170223/1/170223.pdf. – Zugriff: 12.03.2019

[26] Faruki, Parvez ; Bharmal, Ammar ; Laxmi, Vijay ; Ganmoor, Vijay ; Gaur, Manoj S. ; Conti, Mauro: Android Security: A Survey of Issues, Malware Penetration and Defenses. In: *IEEE COMMUNICATIONS SURVEYS AND TUTORIALS* (2014). – URL http://openaccess.city.ac.uk/12200/1/comsec-review(Raj).pdf. – Zugriff: 12.03.2019

[27] Field, Mike: *Zedbaord HDMI v2*. – URL http://hamsterworks.co.nz/mediawiki/index.php/Zedboard_HDMI_v2. – Zugriff: 15.01.2019

[28] Frank, Mario ; Biedert, Ralf ; Ma, Eugene ; Martinovic, Ivan ; Song, Dawn: Touchalytics: On the Applicability of Touchscreen Input as a Behavioral Biometric for Continuous Authentication. (2012). – URL https://arxiv.org/pdf/1207.6231.pdf. – Zugriff: 13.03.2019

[29] Hadid, Abdenour ; Evans, Nicholas ; Marcel, Sabastien ; Fierrez, Julian: Biometrics Systems Under Spoofing Attack. In: *IEEE SIGNAL PROCESSING MAGAZINE* (2015), September, Nr. 20. – URL http://atvs.ii.uam.es/fierrez/files/hadid15SPMspoofing.pdf. – Zugriff: 12.03.2019

[30] HantouchUSA: *How it works: 4-Wire Analog-Resistive Touch Screens*. – URL https://www.sparkfun.com/datasheets/LCD/HOWDOESITWORK.pdf. – Zugriff: 13.01.2019

[31] Jain, Lohit ; Monaco, John V. ; Coakley, Michael J. ; Tappert, Charles C.: Passcode Keystroke Biometric Performance on Smartphone Touchscreens is Superior to that on Hardware Keyboards. In: *International Journal of Research in Computer Applications & Information Technology* 2 (2014), August, Nr. 2014. – URL https://pdfs.semanticscholar.org/9765/9c5e7b1af3ec1abd1454bbf4e28607f12f58.pdf. – Zugriff: 13.03.2019

[32] Jones, Eric ; Oliphant, Travis ; Peterson, Pearu u. a.: *SciPy: Open source scientific tools for Python*. 2001–. – URL http://www.scipy.org/. – Zugriff: 19.04.2019

[33] Kambourakis, Georgios ; Damopoulos, Dimitrios ; Papamartzivanos, Dimotrios ; Pavlidakis, Emmanouil: Introducing touchstroke: keystroke-based authentication system for smartphones. In: *Wiley Online Library* (2014), Ju-

ly. – URL https://onlinelibrary.wiley.com/doi/pdf/10.1002/sec.1061. – Zugriff: 11.03.2019

[34] Kerckhoffs, Auguste: *LA CRYPTOGRAPHIE MILITAIRE.* 1883. – URL https://www.petitcolas.net/kerckhoffs/crypto_militaire_2.pdf. – Zugriff: 22.03.2019

[35] Khan, Rasib ; Hasan, Ragib ; Xu, Jinfang: *SEPIA: Secure-PIN-Authentication-as-a-Service for ATM using Mobile and Wearable Devices.* – URL http://secret.cs.uab.edu/media/Khan2015sepia.pdf. – Zugriff: 11.03.2019

[36] Krishnamoorthy, Sowndarya: Identification of User Behavioral Biometrics for Authentication using Keystroke Dynamics and Machine Learning. (2018). – URL https://scholar.uwindsor.ca/cgi/viewcontent.cgi?article=8442&context=etd. – Zugriff: 13.03.2019

[37] MatheGuru: *Satz von Bayes.* – URL https://matheguru.com/stochastik/satz-von-bayes.html. – Zugriff: 18.03.2019

[38] Monrose, Fabian ; Rubin, Aviel D.: Keystroke dynamics as a biometric for authentication. In: *Elsevier* (1999). – URL http://www1.cs.columbia.edu/~hgs/teaching/security/hw/keystroke.pdf. – Zugriff: 13.03.2019

[39] Morawiec, Darius: *sklearn-porter – Transpile trained scikit-learn estimators to C, Java, JavaScript and others.* – URL https://github.com/nok/sklearn-porter. – Zugriff: 20.04.2019

[40] Ngyuen, Toan V. ; Sae-Bae, Nappa ; Memon, Nasir: DRAW-A-PIN: Authentication using finger-drawn PIN on touch devices. In: *ScienceDirect* (2016), Januar. – URL https://toannguyen.me/Draw-A-PIN.pdf. – Zugriff: 11.03.2019

[41] Ogbanufe, Obi ; Kim, Dan: Comparing fingerprint-based biometrics authentication versus traditionalauthentication methods for e-payment. In: *Elsevier* (2017), Januar. – URL https://www.bedicon.org/wp-content/uploads/2018/01/laws_topic4_source1.pdf. – Zugriff: 11.03.2019

[42] Rashidi, Bahman ; Fung, Carol: A Survey of Android Security Threats and Defenses. In: *Virginia Commonwealth University, Richmond, Virginia* (2015). – URL https://www.researchgate.net/profile/Bahman_Rashidi2/publication/282365848_A_Survey_of_Android_Security_Threats_and_Defenses/links/560ec06908ae6b29b499a51f/

A-Survey-of-Android-Security-Threats-and-Defenses.pdf. – Zugriff: 12.03.2019

[43] Revett, Kenneth ; Gorunescu, Florin ; Gorunescu, Marina ; Ene, Marius ; Magalhaes, Sargio T. de ; Santos, Henrique M. D.: A machine learning approach to keystroke dynamics based user authentication. In: *Security and Digital Forensics* 1 (2007), Nr. 1. – URL https://repositorium.sdum.uminho.pt/bitstream/1822/6388/1/f191031146728125.pdf. – Zugriff: 13.03.2019

[44] Ricardo N. Rodrigues, Niranjan K. ; Govindaraju, Venu: Evaluation of Biometric Spoofing in a Multimodal System. In: *IEEE* (2010). – URL https://www.researchgate.net/profile/Venu_Govindaraju/publication/224194511_Evaluation_of_biometric_spoofing_in_a_multimodal_system/links/02bfe5117e0584c849000000/Evaluation-of-biometric-spoofing-in-a-multimodal-system.pdf. – Zugriff: 12.03.2019

[45] Scikit-Learn: *Model evaluation: quantifying the quality of predictions*. – URL https://scikit-learn.org/stable/modules/model_evaluation.html. – Zugriff: 17.03.2019

[46] Scikit-Learn: *Novelty and Outlier Detection*. – URL https://scikit-learn.org/stable/modules/outlier_detection.html. – Zugriff: 18.03.2019

[47] Seelam, Raj: *I/O Design Flexibility with the FPGA Mezzanine Card (FMC)*. August 2009. – URL https://www.xilinx.com/support/documentation/white_papers/wp315.pdf. – Zugriff: 18.03.2019

[48] Texas Instruments: *KeyStone Architecture Serial Peripheral Interface (SPI) User Guide*. March 2012. test: . – URL http://www.ti.com/lit/ug/sprugp2a/sprugp2a.pdf. – Zugriff: 14.01.2019

[49] Texas Instruments: *TSC2046E - Low Voltage I/O Touch Screen Controller*. 2008. test: . – URL http://www.ti.com/lit/ds/symlink/tsc2046e.pdf. – Zugriff: 15.01.2019

[50] Tiedemann, Michaela: *Machine Learning Methoden - Teil 2/3: Classification & Regression*. 2019. – URL https://www.alexanderthamm.com/de/artikel/machine-learning-methoden-teil-2-3-classification-regression. – Zugriff: 17.03.2019

[51] Varoquaux, Gael ; MÃ¼ller, Andreas ; Grabler, Jaques: *Classifier comparison.* – URL https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html. – Zugriff: 17.03.2019

[52] Vidales, Carlos E.: *How to calibrate touch screens.* – URL https://www.embedded.com/design/system-integration/4023968/How-To-Calibrate-Touch-Screens. – Zugriff: 15.01.2019

[53] Waskom, Michael ; Botvinnik, Olga ; Hobson, Paul ; Cole, John B. ; Halchenko, Yaroslav ; Hoyer, Stephan ; Miles, Alistair ; Augspurger, Tom ; Yarkoni, Tal ; Megies, Tobias ; Coelho, Luis P. ; Wehner, Daniel ; cynddl ; Ziegler, Erik ; diego0020 ; Zaytsev, Yury V. ; Hoppe, Travis ; Seabold, Skipper ; Cloud, Phillip ; Koskinen, Miikka ; Meyer, Kyle ; Qalieh, Adel ; Allan, Dan: *Seaborn: v0.5.0.* – URL https://doi.org/10.5281/zenodo.12710. – Zugriff: 19.04.2019

[54] Wikipedia: *Authentifizierung.* – URL https://de.wikipedia.org/wiki/Authentifizierung. – Zugriff: 12.03.2019

[55] Wikipedia: *Box-Plot.* – URL https://de.wikipedia.org/wiki/Box-Plot. – Zugriff: 19.04.2019

[56] Wikipedia: *Gleichfehlerrate.* – URL https://de.wikipedia.org/wiki/Gleichfehlerrate. – Zugriff: 13.03.2019

[57] Wikipedia: *Greedy-Algorithmus.* – URL https://de.wikipedia.org/wiki/Greedy-Algorithmus. – Zugriff: 18.03.2019

[58] Wikipedia: *Radiale Basisfunktion.* – URL https://de.wikipedia.org/wiki/Radiale_Basisfunktion. – Zugriff: 18.03.2019

[59] Wikipedia: *Support Vector Machine.* – URL https://de.wikipedia.org/wiki/Support_Vector_Machine. – Zugriff: 18.03.2019

[60] Wikipedia: *YCbCr-Farbmodell.* – URL https://de.wikipedia.org/wiki/YCbCr-Farbmodell. – Zugriff: 13.04.2019

[61] Wu, Changsheng ; Ding, Wenbo ; Liu, Ruiyuan ; Wang, Jiyu ; Wang, Aurelia C. ; Wang, Jie ; Li, Shengming ; Zi, Yunlong ; Wang, Zhong L.: Keystroke dynamics enabled authentication and identification using triboelectric nanogenerator array. In:

Materials Today (2017). – URL http://nanoscience.gatech.edu/paper/2018/18_MT_01.pdf. – Zugriff: 13.03.2019

[62] Xilinx: *Vivado Design Suite - AXI Reference Guide*. 4. test: . – URL https://www.xilinx.com/support/documentation/ip_documentation/axi_ref_guide/latest/ug1037-vivado-axi-reference-guide.pdf. – Zugriff: 14.01.2019

[63] Xilinx: *Xilinx OS and Libraries Document Collection*. – URL https://www.xilinx.com/support/documentation/sw_manuals/xilinx2014_1/oslib_rm.pdf. – Zugriff: 05.03.2019

[64] Xilinx: *Zynq-7000 SiC Product Advantages*. – URL https://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html. – Zugriff: 15.01.2019

[65] Xilinx: *Zynq-7000 SoC Data Sheet: OverviewBoard*. – URL https://www.xilinx.com/support/documentation/data_sheets/ds190-Zynq-7000-Overview.pdf. – Zugriff: 17.03.2019

[66] Yim, Annie: *Essential Classification Algorithms with Explanation*. – URL https://www.kaggle.com/anniepyim/essential-classification-algorithms-explained. – Zugriff: 17.03.2019

A Anhang

Die im folgenden aufgeführten Dateien befinden sich auf der CD im Verzeichnis Anhang. Diese liegt der Arbeit bei oder kann beim Verfasser oder Betreuer dieser Arbeit eingesehen werden.

A.1 Vivado-Projekt

(**Anhang/Vivado_Projekt**): Der Vivado Projektordnet. In diesem Befinden sich das Projekt für das Zedboard. In den entspchenden Unterordnen können auch alle VHDL- und C-Dateien gefunden werden. Der Bequemlichkeit halber sind diese jedoch auch in seperaten Ordnern zu finden.

A.2 Schaltpläne

(**Anhang/AB**): Der Schaltplan und das Layout für das Touch-Anschlussboard.

A.3 PL-Dateien

(**Anhang/PL**): Die in der PL verwendeten VHDL- und Constraint-Dateien.

A.4 PS-Dateien

(**Anhang/PS**): Die für die Programmierung des PS verwendeten C-Dateien.

A.5 Kalibrierung

(**Anhang/Kalibrierung**): Die für die Entwicklung der Filterungs- und Transformations-Algrorithmuses verwendeten Messdaten und Python-Skripte.

A.6 Klassifikation

(**Anhang/Klassifikation**): Die für die Entwicklung und Auswertung des Authentifikationsalgorithmuses verwendeten Messdaten und Python-Skripte.

A.7 Grafiken

(**Anhang/Grafiken**): Die Blockschaltbilder, Zustandsdiagramme, Programmablaufpläne und Box-Plots.

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Gemäß der Allgemeinen Prüfungs- und Studienordnung ist zusammen mit der Abschlussarbeit eine schriftliche Erklärung abzugeben, in der der Studierende bestätigt, dass die Abschlussarbeit „– bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit [(§18 Abs. 1 APSO-TI-BM bzw. §21 Abs. 1 APSO-INGI)] – ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt wurden. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich zu machen."

Quelle: §16 Abs. 5 APSO-TI-BM bzw. §15 Abs. 6 APSO-INGI

Erklärung zur selbstständigen Bearbeitung der Arbeit

Hiermit versichere ich,

Name: _____

Vorname: _____

dass ich die vorliegende Bachelorarbeit – bzw. bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit – mit dem Thema:

Realisation einer FPGA-basierten psychometrischen Authentifizierung mittels Touchscreen

ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

_____	_____	_____
Ort	Datum	Unterschrift im Original