BCCM

HOCHSCHULE MITTWEIDA
University of Applied Sciences

MASTERARBEIT

Herr
Jan Leon Bittner

Die Konzeptionierung eines Lightning ATMs als Zugangspunkt zum Bitcoin-Ökosystem

Mittweida, Oktober 2024

Fakultät **Angewandte Computer- und Biowissenschaften**

MASTERARBEIT

Die Konzeptionierung eines Lightning ATMs als Zugangspunkt zum Bitcoin-Ökosystem

Autor:
Jan Leon Bittner

Studiengang:
Blockchain & Distributed Ledger Technologies

Seminargruppe:
BC22w1-M

Erstprüfer:
Prof. Dr.-Ing. Andreas Ittner

Zweitprüfer:
M.Sc. Tim Käbisch

Einreichung:
Mittweida, 23.10.2024

Verteidigung/Bewertung:
Mittweida, 2024

Faculty of **Applied Computer Sciences and Biosciences**

MASTER THESIS

Designing a Lightning ATM as an access point to the Bitcoin ecosystem

Author:
Jan Leon Bittner

Course of Study:
Applied Computer Science

Seminar Group:
BC22w1-M

First Examiner:
Prof. Dr.-Ing. Andreas Ittner

Second Examiner:
M.Sc. Tim Käbisch

Submission:
Mittweida, 23.10.2024

Defense/Evaluation:
Mittweida, 2024

Bibliografische Beschreibung

Bittner, Jan Leon:
Die Konzeptionierung eines Lightning ATMs als Zugangspunkt zum Bitcoin-Ökosystem. – 2024. – 77 S.
Mittweida, Hochschule Mittweida – University of Applied Sciences, Fakultät Angewandte Computer- und Biowissenschaften, Masterarbeit, 2024.

Kurzreferat

Diese Masterarbeit untersucht die Entwicklung eines Automaten als einfachen Zugangspunkt zum Bitcoin-Netzwerk, einem digitalen Zahlungssystem für schnelle und kostengünstige Transaktionen ohne Banken. Der Automat kombiniert Hardware- und Softwarelösungen, um den Kauf von Bitcoin nahezu sofort und benutzerfreundlich abzuwickeln. Dabei werden theoretische Grundlagen sowie praktische Herausforderungen bei der Umsetzung beleuchtet. Durch die Entwicklung eines Prototyps wird gezeigt, wie solche Automaten den Zugang zu Bitcoin vereinfachen können, und es werden Optimierungsmöglichkeiten sowie das Potenzial dieser Lösung für eine breite Nutzung diskutiert.

Inhaltsverzeichnis

1 Einführung

Die zunehmende Verbreitung von Kryptowährungen hat in den letzten Jahren zu einem fundamentalen Wandel im Finanzsektor geführt. Insbesondere Bitcoin, als die erste und bekannteste dezentrale digitale Währung, gewinnt weltweit an Bedeutung. Ursprünglich als Alternative zu traditionellen Fiat-Währungen konzipiert, ermöglicht Bitcoin schnelle und kostengünstige Transaktionen, unabhängig von geografischen Grenzen oder zentralen Institutionen. Mittlerweile haben auch die größten Vermögensverwalter der Welt Interesse an Bitcoin gefunden und bieten den Kauf von Bitcoin über Exchange-Traded Funds (ETFs) an [1]. Diese Eigenschaften machen Bitcoin nicht nur zu einem attraktiven Investitionsobjekt, sondern unter anderem auch zu einem innovativen Zahlungsmittel, das immer häufiger im Alltag eingesetzt wird. Trotz der wachsenden Akzeptanz gibt es jedoch weiterhin Herausforderungen, die den Zugang zu und die Nutzung von Bitcoin einschränken.

Um die Nutzung von Bitcoin als alltägliches Zahlungsmittel zu fördern, sind effiziente und benutzerfreundliche Lösungen erforderlich. Eine der größten Hürden für den Mainstream-Einsatz von Bitcoin sind die technischen und finanziellen Einschränkungen, die durch die Skalierbarkeit und die mit Transaktionen verbundenen Gebühren entstehen. An dieser Stelle kommt das Lightning-Netzwerk ins Spiel, eine zweite Schicht auf dem Bitcoin-Protokoll, die entwickelt wurde, um die Skalierbarkeit des Netzwerks zu verbessern. Durch die Verwendung des Lightning-Netzwerks können Transaktionen nahezu in Echtzeit und mit minimalen Gebühren abgewickelt werden, was das Potenzial von Bitcoin als Zahlungsmittel erheblich erweitert. Diese Technologie trägt dazu bei, die Einstiegshürden für den Zugang zum Bitcoin-Netzwerk weiter zu senken und eröffnet neue Möglichkeiten für innovative Anwendungskonzepte [2].

Ein zentraler Aspekt für den Erfolg Bitcoins und des Lightning-Netzwerks als Zahlungsmittel sind benutzerfreundliche Zugangspunkte, die es Nutzern ermöglichen, schnell und einfach auf das Bitcoin-Ökosystem zuzugreifen. Traditionell geschieht der Erwerb von Bitcoin über Online-Börsen, die jedoch für viele Nutzer technisch kompliziert und weniger intuitiv zu bedienen sein können. Um eine breitere Nutzung zu ermöglichen, sind daher auch physische Zugangspunkte von Interesse, die eine einfache und direkte Möglichkeit bieten, Bitcoin zu erwerben. Hier setzt die Idee eines Lightning-ATMs an: Ein Gerät, das es Nutzern erlaubt, Bargeld gegen Bitcoin einzutauschen, indem die Vorteile des Lightning-Netzwerks genutzt werden [3]. Solche Geräte könnten insbesondere für Nutzer attraktiv sein, die mit den Abläufen an Online-Börsen nicht vertraut sind oder eine schnelle und unkomplizierte Möglichkeit suchen, kleinere Beträge in Bitcoin umzuwandeln.

Diese Masterarbeit beschäftigt sich mit der Konzeptionierung und Entwicklung eines Lightning-ATMs als innovativem Zugangspunkt zum Bitcoin-Ökosystem. Ziel ist es, ein Gerät zu entwerfen, das technisch robust und gleichzeitig benutzerfreundlich ist. Dabei wird untersucht, wie ein solcher ATM die Barrieren für den Zugang zum Bitcoin-Netzwerk senken kann und welche technischen und gestalterischen Herausforderungen bei der Umsetzung überwunden werden müssen. Der Lightning-ATM soll es ermöglichen, Bitcoin unkompliziert und schnell

zu erwerben, ohne, dass tieferes technisches Vorwissen erforderlich ist. Das zugrundeliegende Konzept basiert auf der Nutzung des Lightning-Netzwerks, um eine reibungslose und kostengünstige Transaktion zu gewährleisten.

Im Verlauf der Arbeit werden die verschiedenen Phasen der Entwicklung und Implementierung eines Lightning-ATMs detailliert beschrieben. Neben der technischen Umsetzung, die Aspekte der Hardware- und Softwareentwicklung umfasst, wird auch das Design eines benutzerfreundlichen Interfaces erörtert. Darüber hinaus werden Herausforderungen und Optimierungsmöglichkeiten thematisiert, die sich bei der praktischen Umsetzung ergeben. Die Kapitel 2 und 3 vermitteln das Grundverständnis für die Thematik. Die Konzeptionierung in 4 erläutert detaillreich den Bauprozess des ATMs, während Kapitel 5 diesen in Anwendung zeigt. Das Kapitel 6 fasst die Ergebnisse des Konzepts zusammen und bündelt diese zu einer abschließenden Bewertung. Die Arbeit soll eine fundierte Grundlage dafür bieten, zu evaluieren, ob ein solcher ATM als praktischer Zugangspunkt für das Bitcoin-Ökosystem dienen kann und welche weiteren Entwicklungen erforderlich sind, um ihn zu einem massentauglichen Produkt weiterzuentwickeln.

2 Grundlagen zum Bitcoin-Ökosystem

Bitcoin, die erste und bekannteste Kryptowährung, wurde 2008 von einer anonymen Person oder Gruppe unter dem Pseudonym Satoshi Nakamoto vorgestellt [4]. Seitdem hat sich Bitcoin zu einer dezentralen digitalen Währung entwickelt, die weltweit genutzt wird und mit welcher bereits über eine Milliarde Transaktionen getätigt wurden [5]. Das Bitcoin-Ökosystem umfasst eine Vielzahl von Komponenten, von der Blockchain-Technologie über das Mining bis hin zu verschiedenen Wallet-Typen und Zahlungsnetzwerken wie dem Lightning-Netzwerk [6].

Die zugrundeliegende Technologie von Bitcoin ist die Blockchain, ein dezentralisiertes und verteiltes Ledger-System, das alle Transaktionen öffentlich aufzeichnet. Diese Technologie sorgt dafür, dass Bitcoin-Transaktionen sicher, transparent und nahezu manipulationssicher sind. Im Gegensatz zu traditionellen Währungen, die von Zentralbanken ausgegeben und kontrolliert werden, basiert Bitcoin auf einem dezentralen Netzwerk von Computern, die das Netzwerk durch das Lösen mathematischer Probleme sichern und neue Bitcoin-Einheiten durch einen Prozess namens Mining erzeugen [7].

Das Bitcoin-Ökosystem kann grundsätzlich in zwei Hauptkomponenten unterteilt werden: das On-Chain-Netzwerk und so genannte Second-Layer-Netzwerke, insbesondere das Lightning-Netzwerk. Second-Layer bedeutet, dass diese Netzwerke wie eine weitere Netzwerkschicht agieren, jedoch bspw. viele Second-Layer-Transaktionen zu einer On-Chain-Transaktion zusammengefasst werden. Das On-Chain-Netzwerk bezieht sich auf Transaktionen, die direkt auf der Bitcoin-Blockchain durchgeführt werden, während das Lightning-Netzwerk ein Second-Layer-Protokoll ist, das schnelle und kostengünstige Off-Chain-Transaktionen ermöglicht [6].

Dieses Kapitel wird die wichtigsten Aspekte des On-Chain- und Lightning-Netzwerks beleuchten, einschließlich der Unterschiede in Bezug auf Transaktionen, Gebühren, Sicherheit und Geschwindigkeit.

2.1 On-Chain

Das On-Chain-Netzwerk von Bitcoin bezieht sich auf Transaktionen, die direkt auf der Bitcoin-Blockchain stattfinden. In diesem Abschnitt werden die wesentlichen Aspekte des On-Chain-Systems untersucht, einschließlich der Transaktionen, der Gebühren, der Netzwerksicherheit und der Geschwindigkeit.

2.1.1 Transaktionen und Transfer

Bitcoin-Transaktionen sind der Kern des On-Chain-Systems. Es ist wichtig zu verstehen, dass Bitcoin keine „Coins“ im herkömmlichen Sinne besitzt, sondern dass das Guthaben in Form von Unspent Transaction Outputs (UTXOs) gespeichert wird. UTXOs sind die nicht ausgegebenen Outputs früherer Transaktionen, die einer bestimmten Bitcoin-Adresse zugeordnet sind. Wenn jemand Bitcoin besitzt, besitzt er tatsächlich das Recht, eine oder mehrere UTXOs auszugeben [6].

Eine Bitcoin-Transaktion besteht im Wesentlichen aus zwei Hauptkomponenten: Inputs und Outputs.

Inputs

Die Inputs einer Transaktion verweisen auf vorherige UTXOs, die verwendet werden sollen, um die neue Transaktion zu finanzieren. Diese Inputs sind mit digitalen Signaturen versehen, die durch den privaten Schlüssel des Eigentümers erstellt wurden. Dieser Schlüssel beweist das Recht des Eigentümers, diese UTXOs auszugeben [6].

Outputs

Die Outputs definieren die neuen UTXOs, die nach der Transaktion erzeugt werden, und geben an, welche neuen Adressen die UTXOs ausgeben dürfen. Der Unterschied zwischen dem Input-Wert und dem Output-Wert einer Transaktion ergibt die Gebühr, die Miner erhalten. Dabei ist der Output meist kleiner als der Input. Die Differenz aus beiden ergibt die Gebühr (siehe 2.1.2) [6].

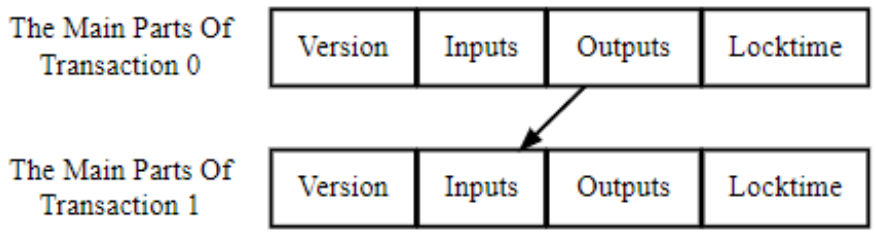


Abbildung 2.1: Hauptbestandteile einer Bitcoin-Transaktion: Inputs, Outputs und Fees [8]

Grafik 2.1 stellt die Struktur einer Bitcoin-Transaktion dar, indem sie die wesentlichen Komponenten und deren Zusammenspiel visualisiert. Eine Transaktion besteht aus Inputs, Outputs und optionalen Transaktionsgebühren (Fees). Jeder Input verweist auf eine vorherige Transaktion und enthält eine digitale Signatur des Senders, die seine Berechtigung zum Ausgeben der Bitcoin bestätigt. Die Outputs spezifizieren die Empfängeradresse und die Menge an übertragenen Bitcoin. Der Gesamtwert aller Outputs darf die Summe der Inputs nicht überschreiten, andernfalls wird die Differenz als Gebühr an den Miner gezahlt, der die Transaktion in einen Block aufnimmt. Die Grafik zeigt anschaulich, wie diese Elemente in einer Transaktion miteinander verknüpft sind und wie die Balance zwischen Inputs und Outputs ausgeglichen wird. Sie verdeutlicht zudem, dass eine Transaktion mehrere Empfänger bedienen kann, was Flexibilität bei der Verteilung von Geldern ermöglicht.

Der Ablauf einer On-Chain-Transaktion lässt sich in folgende Schritte unterteilen:

1. **Erstellung der Transaktion**
   Ein Benutzer initiiert eine Transaktion, indem er eine Bitcoin-Wallet verwendet, die die zugehörigen UTXOs auswählt und die Zieladresse sowie den Betrag angibt.

2. **Verbreitung im Netzwerk**
   Die Transaktion wird dann über das Bitcoin-Netzwerk verbreitet und an alle Knoten (Nodes) gesendet [7].
3. **Verifizierung durch Miner**
   Miner verifizieren die Transaktion, indem sie sicherstellen, dass die verwendeten UTXOs gültig sind und noch nicht ausgegeben wurden. Dies stellt sicher, dass keine doppelten Ausgaben stattfinden [7].
4. **Inklusion in einen Block**
   Wenn die Transaktion verifiziert ist, wird sie zusammen mit anderen Transaktionen in einem neuen Block gespeichert [7].
5. **Anhängung an die Blockchain**
   Der Block wird an die bestehende Blockchain angehängt, wodurch die Transaktion endgültig und unveränderlich wird [7].

Diese Struktur stellt sicher, dass Bitcoin-Transaktionen sicher und transparent sind. Die Unumkehrbarkeit nach Bestätigung ist ein wesentlicher Bestandteil des Bitcoin-Netzwerks und unterscheidet es von traditionellen Zahlungssystemen.

2.1.2 Gebühren

Die Gebührenstruktur im Bitcoin-Netzwerk ist ein entscheidender Aspekt für die Nutzung und Skalierbarkeit von On-Chain-Transaktionen. Im Gegensatz zu traditionellen Finanzsystemen, bei denen Gebühren häufig von zentralen Institutionen festgelegt werden, werden Bitcoin-Gebühren von den Benutzern selbst bestimmt und von den Minern als Anreiz für die Verifizierung von Transaktionen gesammelt [9].

Die Höhe der Transaktionsgebühren hängt von verschiedenen Faktoren ab:

Größe der Transaktion

Da Bitcoin-Blöcke eine maximale Größe von etwa 1 Megabyte haben, hängt die Gebühr oft vom Speicherplatzverbrauch der Transaktion in Bytes ab. Je mehr Speicherplatz die Transaktion benötigt, desto höher die Gebühr, um eine schnellere Bestätigung zu gewährleisten [10].

Netzwerkauslastung

Bei hohem Transaktionsvolumen im Netzwerk steigt die Nachfrage nach Speicherplatz in den Blöcken, was zu höheren Gebühren führt [11].

Dringlichkeit

Benutzer, die möchten, dass ihre Transaktion schnell bestätigt wird, können höhere Gebühren zahlen, um ihre Transaktion im nächsten Block unterzubringen [10]. Miner wählen in der Regel Transaktionen mit den höchsten Gebühren aus, um sie in den nächsten Block aufzunehmen. Der Speicherplatz ist jedoch auf 1MB begrenzt, weshalb der nächste Block meist

mit den Transaktionen mit den höchsten Gebühren aufgefüllt wird. Dadurch entsteht ein Marktmechanismus sowie Konkurrenz zwischen den Marktteilnehmern, bei dem Benutzer die Gebühren entsprechend der Dringlichkeit ihrer Transaktionen anpassen können [11].

Child Pays for Parent (CPFP)

Diese Strategie wird verwendet, wenn eine nachfolgende Transaktion eine höhere Gebühr zahlt, um eine vorherige Transaktion mit niedrigerer Gebühr schneller zu bestätigen. Der Miner wird durch die kombinierten Gebühren beider Transaktionen incentiviert, beide Transaktionen in den nächsten Block aufzunehmen [12].

Replace-by-Fee (RBF)

Mit RBF kann ein Benutzer eine bereits gesendete, aber noch nicht bestätigte Transaktion durch eine neue Transaktion mit einer höheren Gebühr ersetzen. Dies ist besonders nützlich, wenn die ursprüngliche Transaktion aufgrund einer zu niedrigen Gebühr im Mempool festhängt [13]. RBF ermöglicht es, eine Transaktion „zu überbieten “, sodass sie schneller von den Minern aufgenommen wird. Diese Methode kann jedoch auch Bedenken hinsichtlich der Sicherheit und des Vertrauens in ausstehende Transaktionen wecken, da unbestätigte Transaktionen unter bestimmten Umständen ersetzt werden können.

2.1.3 Netzwerksicherheit

Die Sicherheit des Bitcoin-Netzwerks ist einer der zentralen Aspekte, die zur breiten Akzeptanz der Kryptowährung beigetragen haben. Das Netzwerk verwendet ein Proof-of-Work (PoW)-Konsensprotokoll, das sicherstellt, dass Transaktionen unveränderlich und manipulationssicher sind [4].

Proof-of-Work (PoW)

Das PoW-Protokoll basiert darauf, dass Miner eine spezielle Nonce (eine Zahl, die nur einmal verwendet wird) finden, die, zusammen mit den Daten des zugehörigen Blocks, einen Hashwert erzeugt, der unter einem bestimmten Schwellenwert liegt. Dieser Schwellenwert wird durch die Zielschwierigkeit des Netzwerks definiert. Um diese Nonce zu finden, müssen Miner eine große Anzahl von Hash-Operationen durchführen, die im Wesentlichen einer Reihe von Versuchen gleichen, eine gültige Lösung durch „Würfeln“ zu finden. Es gibt keine Abkürzung oder Methode, um die richtige Nonce zu berechnen, außer durch wiederholtes Probieren. Die Schwierigkeit des Minings wird dabei durch die Anzahl der führenden Nullen im Hash-Wert bestimmt: je mehr führende Nullen erforderlich sind, desto kleiner wird der Bereich, einen gültigen Hash zu finden und desto unwahrscheinlicher wird es, die passende Nonce zu würfeln [14].

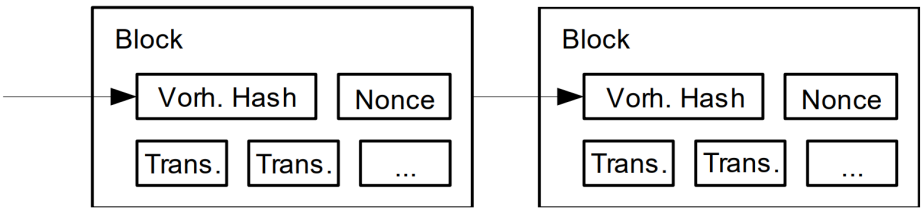


Abbildung 2.2: Veranschaulichung des Mechanismus aus Hash-Pointer sowie der durch Miner gefundene Nonce in der Bitcoin-Blockchain [4]

Die Abbildung 2.2 zeigt die Funktionsweise des Hash-Pointers im Bitcoin-Netzwerk. Jeder Block ist durch einen Hash-Wert mit dem vorherigen Block verbunden. Miner probieren verschiedene Nonces in Kombination mit den Blockdaten aus, bis sie einen Hash-Wert finden , der den vorgegebenen Kriterien entspricht (PoW). Dies führt zu einer langen Kette von Blöcken, die sicherstellt, dass jede Änderung in einem Block sofort den gesamten nachfolgenden Kettenverlauf beeinflussen würde, wodurch Manipulationen verhindert werden.

Dezentralisierung

Ein wesentlicher Sicherheitsfaktor im Bitcoin-Netzwerk ist seine Dezentralisierung. Das Netzwerk besteht aus Tausenden von Knoten (Nodes), die weltweit verteilt sind und unabhängig voneinander Transaktionen überprüfen und validieren. Eine Schlüsselkomponente, die diese Dezentralisierung ermöglicht, ist die relativ kleine Blockgröße von etwa 1 Megabyte. Diese Beschränkung stellt sicher, dass die Hardwareanforderungen für den Betrieb einer

vollständigen Bitcoin-Node relativ gering bleiben [15]. Dadurch wird es für Benutzer weltweit erschwinglich und technisch machbar, eine eigene Node zu betreiben, was die Kontrolle über das Netzwerk breit verteilt und die Widerstandsfähigkeit gegen zentrale Kontrolle oder Angriffe erheblich stärkt [16].

Blockzeiten und Difficulty Adjustment

Die durchschnittliche Zeit, die benötigt wird, um einen neuen Block zur Bitcoin-Blockchain hinzuzufügen, beträgt etwa 10 Minuten. Diese Zeit wird durch das sogenannte Difficulty Adjustment reguliert. Alle 2016 Blöcke, also etwa alle zwei Wochen, passt das Netzwerk die Schwierigkeit Blöcke zu finden an, um sicherzustellen, dass die Blockzeit im Durchschnitt weiterhin bei etwa 10 Minuten bleibt. Diese Anpassung hängt von der gesamten Hashrate des Netzwerks ab; wenn die Hashrate steigt, erhöht sich die Schwierigkeit, und wenn die Hashrate sinkt, wird die Schwierigkeit entsprechend verringert [4].

51% Angriff

Ein theoretisch möglicher, aber äußerst kostspieliger Angriff auf das Bitcoin-Netzwerk ist der 51%-Angriff. Ein solcher Angriff würde es einem Angreifer ermöglichen, die Blockchain zu reorganisieren, was das Risiko birgt, Transaktionen rückgängig zu machen und Doppelausgaben (auch Double Spend genannt) zu tätigen. Um einen solchen Angriff durchzuführen, müsste der Angreifer jedoch mehr als die Hälfte der gesamten Hashrate des Netzwerks kontrollieren. Zum aktuellen Zeitpunkt liegt die Hashrate des Bitcoin-Netzwerks bei etwa 670 Exahashes pro Sekunde (EH/s) [17]. Laut aktuellen Berechnungen würde es etwa 752.000 USD pro Stunde kosten [18], eine 51%-Attacke auf das Bitcoin-Netzwerk durchzuführen. Dies sind jedoch theoretische und wahrscheinlich viel zu niedrige Schätzungen, da es in der Praxis extrem unwahrscheinlich ist, dass eine einzelne Entität in der Lage wäre, die nötige Hashrate zu kontrollieren, insbesondere, da die Kosten für ein solches Unterfangen massiv ansteigen würden, wenn sich die Hashrate erhöht [19].

2.1.4 Geschwindigkeit

Die Geschwindigkeit von On-Chain-Transaktionen im Bitcoin-Netzwerk ist ein häufig diskutiertes Thema, insbesondere im Kontext der Skalierbarkeit und des Vergleichs mit traditionellen Zahlungssystemen. Aufgrund der Struktur des Netzwerks und der Beschaffenheit der Blockchain gibt es inhärente Geschwindigkeitsbegrenzungen.

Blockzeit und Transaktionsbestätigungen

Die durchschnittliche Zeit, die benötigt wird, um einen neuen Block zur Bitcoin-Blockchain hinzuzufügen, beträgt etwa 10 Minuten [4]. Diese Zeit ist jedoch nur ein Durchschnittswert und kann je nach Netzwerkbedingungen und Würfelglück schwanken. Die Blockzeit beeinflusst direkt die Geschwindigkeit, mit der Transaktionen bestätigt werden. Eine Transaktion gilt als sicher bestätigt, wenn sie in einem Block enthalten ist, und es wird empfohlen, bei größeren Transaktionen auf mehrere Blockbestätigungen zu warten, um die Sicherheit zu erhöhen. Die

Blockbestätigung ist die Anzahl der nach dem zu bestätigten Block geminten Blöcke und ist ein Messwert für die Vertrauenswürdigkeit der Transaktionen. Je mehr Bestätigungen, desto Manipulationssicherer ist dieser Block [14].

Transaktionsdurchsatz

Der Bitcoin-Blockchain ist aufgrund ihrer Struktur und des PoW-Konsensprotokolls ein relativ niedriger Transaktionsdurchsatz eigen. Die Bitcoin-Blockchain kann etwa 7 Transaktionen pro Sekunde (TPS) verarbeiten [11]. Im Vergleich dazu können traditionelle Zahlungssysteme wie Visa Tausende von TPS verarbeiten [20]. Diese Begrenzung führt oft zu Verzögerungen und erhöhten Gebühren, insbesondere während Zeiten hoher Netzwerkauslastung.

Netzwerkauslastung und Mempool

Wenn das Netzwerk stark ausgelastet ist, können Transaktionen im sogenannten Mempool (Memory Pool) „steckenbleiben“, bevor sie in einen Block aufgenommen werden. Der Mempool speichert alle noch nicht bestätigten Transaktionen, und Miner wählen typischerweise diejenigen aus, die die höchsten Gebühren bieten. Dadurch kann es zu Verzögerungen kommen, wenn der Mempool voll ist, was die effektive Transaktionsgeschwindigkeit weiter reduziert [16].

Aktuelle technologische Fortschritte

Taproot und Schnorr-Signaturen: Ein bedeutender Fortschritt im Bitcoin-Netzwerk ist die Einführung von Taproot im November 2021, welches die Effizienz, Privatsphäre und Skalierbarkeit von Bitcoin-Transaktionen verbessert. Taproot kombiniert mehrere Aspekte von Transaktionen in einem einzigen Output, was die Verarbeitung vereinfacht und die Größe der Transaktionen reduziert, was wiederum die Geschwindigkeit erhöht [21]. Zusätzlich nutzt Taproot Schnorr-Signaturen, die im Vergleich zu bisherigen Signaturschemata kleiner und effizienter sind. Diese technologischen Verbesserungen ermöglichen es, mehr Transaktionen in einen Block zu packen, was die Effizienz des Netzwerks insgesamt steigert [15].

2.2 Lightning

Das Lightning-Netzwerk ist eine Second-Layer-Technologie, die auf der Bitcoin-Blockchain aufbaut und entwickelt wurde, um die Skalierbarkeitsprobleme und die hohen Transaktionsgebühren des Bitcoin-On-Chain-Netzwerks zu überwinden. Seit seiner Einführung im Jahr 2018 hat sich das Lightning-Netzwerk kontinuierlich weiterentwickelt und ist bis 2024 zu einer robusten, weit verbreiteten Lösung für schnelle und kostengünstige Bitcoin-Transaktionen geworden. Das Lightning-Netzwerk ermöglicht nahezu sofortige Transaktionen mit minimalen Gebühren, indem es Transaktionen Off-Chain, also außerhalb der On-Chain Transaktionen abwickelt, bevor sie schließlich in der Bitcoin-Blockchain zusammengefasst und verifiziert werden.

Während das Bitcoin-On-Chain-Netzwerk robust und sicher ist, sind seine begrenzte Transaktionskapazität und die potenziell hohen Gebühren Hindernisse für den täglichen Gebrauch, insbesondere bei Mikrotransaktionen oder in Szenarien, in denen Geschwindigkeit entscheidend ist. Mikrotransaktionen sind meist Transaktionen mit Beträgen in Cent oder sogar Millicent bis niedrige einstellige Eurobeträge. Hier kommt das Lightning-Netzwerk ins Spiel.

2.2.1 Transaktionen und Transfers

In diesem Abschnitt werden die fundamentalen Mechanismen von Transaktionen und Transfers im Lightning-Netzwerk dargestellt, einschließlich der Einrichtung von Zahlungskanälen und der Implementierung von Hash Time-Locked Contracts (HTLCs) zur Gewährleistung sicherer und flexibler Transaktionen. Es wird aufgezeigt, wie das Netzwerk die Durchführung von Multihop-Zahlungen und Routing ermöglicht, um Transaktionen zwischen Parteien ohne direkte Verbindung zu ermöglichen.

Kanalbasierte Transaktionen

Das Lightning-Netzwerk basiert auf dem Konzept von Zahlungskanälen. Zwei Parteien eröffnen einen Zahlungskanal, indem sie einen Multisignatur-Transaktionsvertrag auf der Bitcoin-Blockchain erstellen. Dieser Kanal ermöglicht es den Parteien, eine unbeschränkte Anzahl von Transaktionen zwischen sich durchzuführen, ohne dass jede Transaktion auf der Blockchain veröffentlicht werden muss [9]. Nur die Eröffnungs- und die Abschlussbilanz des Kanals werden auf der Bitcoin-Blockchain gespeichert. Dies reduziert die Belastung der Haupt-Blockchain erheblich und ermöglicht nahezu sofortige Transaktionen zwischen den Beteiligten.

Multihop-Zahlungen und Routing

Ein wesentliches Merkmal des Lightning-Netzwerks ist die Fähigkeit, Zahlungen über mehrere Kanäle zu routen. Das bedeutet, dass zwei Parteien, die keinen direkten Kanal zueinander haben, dennoch Transaktionen über andere Knotenpunkte im Netzwerk durchführen können. Diese sogenannten Multihop-Zahlungen werden, ähnlich wie Single-Hop Zahlungen durch die Verwendung von Hash Time-Locked Contracts (HTLCs) gesichert, die sicherstellen, dass das Geld entweder sicher ankommt oder an den Absender zurückgesendet wird [3]. Dies macht das Netzwerk extrem flexibel und ermöglicht es, Transaktionen schnell und sicher durchzuführen, selbst wenn die Beteiligten keine direkte Verbindung zueinander haben.

HTLCs und Smart Contracts

Eine der technologischen Grundlagen des Lightning-Netzwerks sind Hash Time-Locked Contracts (HTLCs). Diese Smart Contracts stellen sicher, dass Transaktionen entweder erfolgreich durchgeführt werden oder nach einer bestimmten Zeit automatisch zurückerstattet werden. HTLCs verwenden kryptografische Hashes und Zeitbeschränkungen welche dabei helfen, dass das Risiko von Betrug minimiert wird [22].

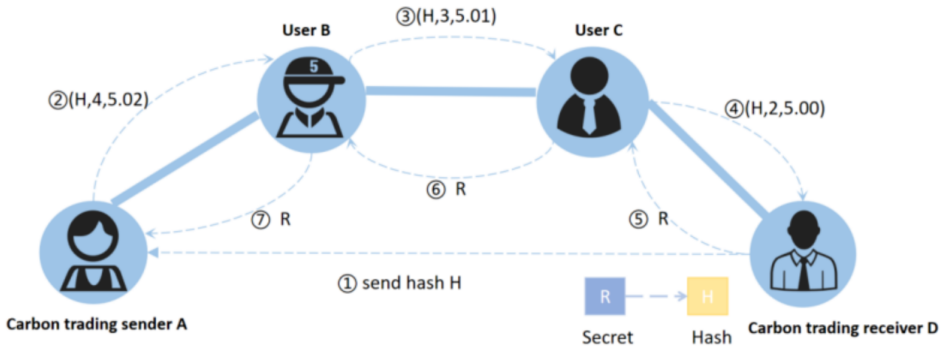


Abbildung 2.3: Darstellung eines HTLC-basierten Zahlungspfades im Lightning-Netzwerk [23]

Die Abbildung 2.3 zeigt eine typische Zahlungsabwicklung mit Hash Time-Locked Contracts (HTLCs) im Lightning-Netzwerk, die als Sicherheitsmechanismus zur bedingten Transaktionsverifikation dient. A möchte eine Zahlung an D über das Lightning Network senden. D schickt den Zahlungshash (Payment Hash) an A, woraufhin A ein HTLC an Node B sendet, das den Betrag und den Hash des Zahlungsvorbilds R (Pre-Image) enthält. B leitet das HTLC an Node C weiter, ebenfalls mit dem Hash und dem festgelegten Betrag. Schließlich sendet C das HTLC an D. Sobald D das korrekte Zahlungsvorbild R bereitstellt, wird die Zahlung Schritt für Schritt in umgekehrter Reihenfolge durch alle Nodes freigegeben und an D überwiesen.

2.2.2 Gebühren

Dynamische und niedrige Gebührenstruktur

Im Gegensatz zu den oft hohen und variablen On-Chain-Gebühren bietet das Lightning-Netzwerk wesentlich niedrigere und dynamischere Gebührenstrukturen. Die Gebühren im Lightning-Netzwerk setzen sich hauptsächlich aus zwei Komponenten zusammen:

Routing-Gebühr: Diese Gebühr wird von den Nodes erhoben, die Transaktionen zwischen Sender und Empfänger weiterleiten. Jede Node im Netzwerk kann seine eigenen Gebühren festlegen, basierend auf seiner eigenen wirtschaftlichen Strategie und den Marktbedingungen [24].

Basisgebühr und Proportionalgebühr: Neben der Routing-Gebühr können Nodes auch eine Basisgebühr sowie eine proportional zum Transaktionsbetrag berechnete Gebühr erheben. Diese Gebühren sind in der Regel sehr niedrig, was das Lightning-Netzwerk besonders attraktiv für Mikrotransaktionen macht, bei denen traditionelle On-Chain-Transaktionen aufgrund hoher Gebühren unpraktisch wären. Für hohe Beträge eignet sich das Lightning-Netzwerk jedoch so solange, bis die proportionale Gebühr die On-Chain Gebühr übersteigt [25].

Anreize für Nodes

Die Gebühren im Lightning-Netzwerk dienen auch als Anreiz für Node-Betreiber, Liquidität bereitzustellen und das Netzwerk zu stabilisieren. Nodes, die eine hohe Liquidität und zuverlässige Verbindungen anbieten, können durch das Erheben von Gebühren von den Nutzern profitieren, die ihre Kanäle für das Routing von Zahlungen nutzen [26].

2.2.3 Sicherheit

Sicherheit durch Multisignatur und HTLCs

Die Sicherheit des Lightning-Netzwerks wird durch mehrere Mechanismen gewährleistet. Zum einen wird jede Transaktion in einem Zahlungskanal durch einen Multisignatur-Vertrag abgesichert, der sicherstellt, dass keine Partei die Mittel ohne Zustimmung der anderen Partei ausgeben kann. Zum anderen sorgen HTLCs dafür, dass Zahlungen entweder erfolgreich abgeschlossen werden oder automatisch verfallen, wodurch das Szenario eines teilweisen Zahlungsausfalls ausgeschlossen wird [25].

Risiko der Kanalschließung und Angriffe

Während das Lightning-Netzwerk im Allgemeinen als sicher gilt, gibt es bestimmte Angriffsvektoren, die berücksichtigt werden müssen. Ein bekanntes Beispiel ist der „Flood and Loot-Angriff “, bei dem ein Angreifer versucht, eine große Anzahl von Zahlungskanälen gleichzeitig zu schließen, um den Netzwerkverkehr zu überlasten und eine Erstattung zu erzwingen [27]. Solche Angriffe sind jedoch theoretisch und in der Praxis schwer durchzuführen, da sie ein tiefes Verständnis der Netzwerkstruktur und erhebliche Ressourcen erfordern.

Backup-Mechanismen und Watchtowers

Zur weiteren Sicherung des Netzwerks und zum Schutz der Teilnehmer vor betrügerischen Kanal-Schließungen gibt es sogenannte Watchtowers. Diese spezialisierten Nodes überwachen das Netzwerk und greifen ein, wenn eine böswillige Schließung eines Zahlungskanals erkannt wird [28]. Dies stellt sicher, dass Benutzer, die möglicherweise offline sind oder ihre Kanäle nicht ständig überwachen können, dennoch geschützt sind.

2.2.4 Geschwindigkeit

Sofortige Zahlungen

Einer der größten Vorteile des Lightning-Netzwerks ist die Möglichkeit, Zahlungen so schnell wie es die Internetverbindung zulässt, durchzuführen. Da die Transaktionen Off-Chain abgewickelt werden und nur die Eröffnung und Schließung eines Zahlungskanals auf der Blockchain vermerkt werden, entfallen die typischen Wartezeiten, die mit On-Chain-Transaktionen verbunden sind [9]. Diese Eigenschaft macht das Lightning-Netzwerk besonders attraktiv für Anwendungen, bei denen schnelle Zahlungen erforderlich sind, wie etwa Point-of-Sale-Transaktionen oder Online-Dienste.

Skalierbarkeit und Netzwerkauslastung

Durch das Second-Layer Design ist das Lightning-Netzwerk theoretisch in der Lage, eine unbegrenzte Anzahl an Transaktionen pro Sekunde abzuwickeln, was die Kapazität des On-Chain-Bitcoin-Netzwerks deutlich übersteigt [29]. Dies bietet eine Lösung für die Skalierbarkeitsprobleme, die Bitcoin auf der Haupt-Blockchain hat, und ermöglicht es dem Netzwerk, eine große Anzahl von Transaktionen gleichzeitig zu verarbeiten, ohne dass die Kosten oder die Geschwindigkeit darunter leiden.

Verbesserte Effizienz durch AMP und andere Technologien

Technologische Fortschritte wie Atomic Multipath Payments (AMP) ermöglichen es, größere Zahlungen über mehrere kleinere Pfade gleichzeitig zu senden. Dies verbessert nicht nur die Effizienz, sondern erhöht auch die Zuverlässigkeit der Zahlung, da es weniger wahrscheinlich ist, dass eine Zahlung aufgrund von Liquiditätsproblemen in einem einzelnen Kanal fehlschlägt [30].

3 Lightning im ATM-Kontext

Im folgenden Kapitel wird erläutert, wie das Lightning-Netzwerk im ATM-Kontext genutzt werden kann, um schnelle und kostengünstige Bitcoin-Transaktionen zu ermöglichen. Durch die Second-Layer Verarbeitung von Zahlungen bietet es nahezu sofortige Abwicklung und erhöht gleichzeitig die Flexibilität bei der Wahl der Wallet-Typen, was sowohl die Nutzererfahrung als auch die Sicherheit positiv beeinflusst.

3.1 Vorteile durch Lightning für das Konzept

In diesem Abschnitt werden die Vorteile des Lightning-Netzwerks für das Bitcoin-ATM Konzept dargestellt. Besonders betont werden die schnelle Transaktionsabwicklung, die geringen Gebühren, die Skalierbarkeit für Mikrozahlungen sowie die Flexibilität bei der Auswahl der Wallet durch die Nutzer.

3.1.1 Instant Settlement

Das Lightning-Netzwerk bietet erhebliche Vorteile für den Einsatz in Bitcoin-ATMs, insbesondere aufgrund des „Instant Settlement“, also der sofortigen Abwicklung von Zahlungen. Verglichen mit traditionellen Bankenstrukturen und auch mit On-Chain Bitcoin Transaktionen ermöglicht das Lightning-Netzwerk eine nahezu verzögerungsfreie Transaktionsverarbeitung, was es besonders attraktiv für Bitcoin-ATMs macht.

In der traditionellen Bankinfrastruktur, insbesondere bei Überweisungen, erfolgt die Abwicklung von Zahlungen über mehrere Schritte. Zunächst muss die Überweisung durch das interne Bankensystem verarbeitet werden, anschließend wird die Transaktion durch das Clearing-House-System weitergeleitet, das zwischen verschiedenen Banken agiert. Dieser Prozess kann mehrere Tage dauern, insbesondere bei internationalen Zahlungen, da unterschiedliche Bankensysteme involviert sind und oftmals unterschiedliche Abrechnungssysteme wie SWIFT genutzt werden [31]. Ein weiterer Verzögerungsfaktor sind die sogenannten „Batch Settlements“, bei denen Zahlungen gebündelt und zu festen Zeiten verarbeitet werden. Diese langsamen Prozesse machen das traditionelle Bankensystem ungeeignet für den sofortigen Austausch von Wert, insbesondere bei Mikrotransaktionen, die für den täglichen Gebrauch entscheidend sind.

Im Vergleich dazu bietet das Lightning-Netzwerk ein sofortiges Settlement, indem es Zahlungen Off-Chain verarbeitet. Dies bedeutet, dass Zahlungen nicht sofort auf der Bitcoin-Blockchain vermerkt werden müssen, sondern direkt zwischen den Teilnehmern über bereits bestehende Zahlungskanäle ablaufen [9]. Diese Off-Chain-Technologie ermöglicht es, Zahlungen innerhalb von Millisekunden oder Sekunden abzuwickeln, ohne dass eine Verzögerung durch Blockchain-Bestätigungen entsteht. Für den Betrieb eines Bitcoin-ATMs bedeutet dies, dass Nutzer unmittelbar nach Eingabe ihrer Lightning-Invoice oder dem Scannen eines QR-Codes die entsprechende Menge an Bitcoin auf ihr Wallet erhalten können. Diese Sofortabwicklung macht das Nutzererlebnis flüssiger und intuitiver, was insbesondere in der Interaktion mit Endkunden an physischen Terminals von Bedeutung ist.

3.1.2 Nutzung von Custodial-Wallets sowie Non-Custodial Wallet

Bei der Auswahl zwischen Custodial und Non-Custodial Wallets für den Einsatz in einem Bitcoin Lightning-ATM müssen mehrere Faktoren berücksichtigt werden, die sowohl die Benutzerfreundlichkeit als auch die Sicherheit betreffen. Wallets sind eine Art digitale Geldbörse, die es dem Nutzer ermöglicht seine privaten Schlüssel und UTXOs zu verwalten und diese Auszugeben oder bei Verwendung einer Lightning-Wallet, mit dem Lightning-Netzwerk zu interagieren. Diese Wallet-Typen (Custodial und Non-Custodial) unterscheiden sich wesentlich in der Kontrolle über die privaten Schlüssel und der Verantwortung für die Sicherung der Bitcoin. Im Kontext eines Lightning-ATMs wirken sich diese Unterschiede direkt auf die Nutzungserfahrung und die damit verbundenen Risiken aus.

Ein wesentlicher Vorteil von Custodial Wallets im Zusammenhang mit Lightning-ATMs ist die Benutzerfreundlichkeit. Die Custodial Wallets verwalten die Lightning-Infrastruktur der Nutzer bzgl. Channelmanagement und Liquidität, was den Prozess für den Endverbraucher vereinfacht. Zudem bietet diese Struktur eine nahtlose und schnelle Transaktionsabwicklung, da der Betreiber über eben diese ausreichende Liquidität verfügt und keine komplexen Wallet-Konfigurationen seitens des Nutzers notwendig sind [3].

Der Custodian, also der Betreiber und Verwalter der Wallets, kann Kanäle gezielt aufrechterhalten und optimieren, um eine stabile und zuverlässige Transaktionsabwicklung sicherzustellen. Durch die Zentralisierung der Kontrolle über die Mittel können potenzielle Routing-Probleme vermieden und die Zahlungen schneller abgeschlossen werden [9].

Der größte Nachteil von Custodial Wallets ist das erhöhte Sicherheitsrisiko, da der Betreiber die privaten Schlüssel verwaltet. Sollte die Infrastruktur des Betreibers kompromittiert werden, besteht das Risiko eines Totalverlusts der Kundengelder. Dieses Problem ist besonders relevant im Fall eines Bitcoin Lightning-ATMs, der oft an öffentlichen Orten aufgestellt ist und möglicherweise ein attraktives Ziel für Cyberangriffe darstellt. Ein weiteres Risiko besteht darin, dass der Betreiber insolvent werden könnte, wodurch Kunden den Zugriff auf ihre Bitcoin verlieren würden [10].

Ein weiteres Problem betrifft die regulatorischen Anforderungen, die an Custodial Dienste gestellt werden. Der Betreiber eines Custodial Wallet-Systems könnte gezwungen sein, die Identität seiner Nutzer zu überprüfen und KYC (Know Your Customer) und AML (Anti-Money-Laundering)-Richtlinien einzuhalten. Dies könnte zusätzliche Kosten und rechtliche Komplexität mit sich bringen, die den Betrieb eines Lightning-ATMs erschweren oder verlangsamen könnten [15].

Non-Custodial Wallets bieten ein höheres Maß an Sicherheit für den Nutzer, da dieser die volle Kontrolle über seine privaten Schlüssel behält. Im Kontext eines Lightning-ATMs bedeutet dies, dass der Betreiber des ATMs nicht für die Sicherung der Gelder verantwortlich ist, was das Risiko eines Diebstahls oder einer Insolvenz des Betreibers für den Kunden minimiert. Dieses dezentrale Sicherheitsmodell ist besonders attraktiv für technikaffine Nutzer, die Wert auf die Eigenverantwortung und die Kontrolle ihrer Bitcoin legen [30].

Zudem bieten Non-Custodial Wallets im Vergleich zu Custodial Lösungen eine größere Unabhängigkeit von regulatorischen Eingriffen, da keine zentrale Instanz über die Gelder verfügt. Der ATM-Betreiber agiert hier lediglich als Vermittler der Transaktion, ohne Zugang zu den privaten Schlüsseln der Nutzer. Dies reduziert potenzielle regulatorische Hürden, da die Verantwortung für KYC- und AML-Vorgaben in erster Linie beim Nutzer selbst liegt [16].

Ein großer Nachteil von Non-Custodial Wallets ist die Komplexität für den Endverbraucher. Nutzer müssen eigene Wallets erstellen, ihre privaten Schlüssel sicher verwahren und sich aktiv um das Management ihres Kanals und der Liquidität kümmern. Im Rahmen eines Lightning-ATMs könnte dies die Benutzererfahrung erschweren, insbesondere für Personen, die keine technische Expertise haben. Dies könnte zu einer geringeren Akzeptanz des ATMs führen, da die Handhabung weniger intuitiv ist als bei Custodial Wallets [24].

Ein weiteres Problem besteht im Liquiditätsmanagement. Bei Non-Custodial Wallets muss der Nutzer sicherstellen, dass er über ausreichende Liquidität in seinen Lightning-Kanälen verfügt, um eine erfolgreiche Transaktion durchführen zu können. Sollte der Nutzer nicht ausreichend Liquidität zur Verfügung haben, könnte die Transaktion fehlschlagen, was zu Frustration und potenziell negativem Feedback für den ATM-Betreiber führen könnte. Dieses Szenario tritt besonders dann auf, wenn der Nutzer den ATM verwendet, um größere Beträge zu transferieren, für die er möglicherweise nicht ausreichend vorbereitete Kanäle hat [29].

Zusätzlich besteht das Risiko von Angriffsvektoren, wie etwa durch sogenannte „Griefing Attacks“, bei denen ein böswilliger Angreifer versucht, die Funktionsfähigkeit des Kanals zu beeinträchtigen, indem er die Zeitfenster für das Schließen eines Kanals absichtlich verlängert. Dies könnte den reibungslosen Betrieb eines Lightning-ATMs beeinträchtigen, da der Nutzer auf das korrekte Funktionieren des Netzwerks angewiesen ist [28].

Die Wahl zwischen Custodial und Non-Custodial Wallets für einen Lightning-ATM hängt stark von den Anforderungen des Nutzers und den operativen Zielen des Betreibers ab. Custodial Wallets bieten eine einfache und nutzerfreundliche Lösung, bergen jedoch Sicherheits- und regulatorische Risiken. Non-Custodial Wallets hingegen bieten mehr Sicherheit und Unabhängigkeit, sind jedoch komplexer in der Handhabung und können durch Liquiditätsprobleme und Netzwerkfehler beeinträchtigt werden. Schlussendlich muss der ATM-Betreiber selbst entscheiden, wo die individuellen Präferenzen liegen und dementsprechend eine Custodial Wallet oder eben eine Non-Custodial als ATM-Wallet verwenden.

3.2 Nachteile durch Lightning für das Konzept

Im Folgenden werden die spezifischen Nachteile des Lightning-Netzwerks im Kontext eines Bitcoin-ATMs aufgezeigt. Dabei wird insbesondere auf Herausforderungen wie Liquiditätsmanagement, Transaktionskomplexität und Sicherheitsrisiken eingegangen.

3.2.1 Liquidität

Liquidität im Lightning-Netzwerk bezieht sich auf die Verfügbarkeit von Bitcoin in offenen Kanälen, die es ermöglichen, Zahlungen schnell und effizient durchzuführen. Im Kontext eines Bitcoin Lightning-ATMs, bei dem Kunden Münzen gegen Lightning-Bitcoin tauschen, treten spezifische Liquiditätsprobleme auf, die den Betrieb beeinflussen können.

Liquiditätsanforderungen für den ATM-Betrieb

Ein Lightning-ATM benötigt ausreichend Inbound-Liquidität, um Bitcoin-Zahlungen an die Nutzer zu senden. Da der ATM nur in eine Richtung arbeitet (also Fiat-Geld entgegennimmt und Bitcoin an den Nutzer ausgibt), ist es wichtig, dass der Betreiber genügend Bitcoin in seinen Kanälen hat, um die Auszahlungen an die Kunden zu gewährleisten. Anders als beim Empfang von Zahlungen (wo Outbound-Liquidität wichtig wäre), muss der ATM Betreiber dafür sorgen, dass er regelmäßig Kanäle auflädt, um Inbound-Liquidität zu sichern.

Beispiel:

Wenn ein Nutzer am ATM Bitcoin kaufen möchte, aber der Betreiber nicht genügend Inbound-Liquidität in den Kanälen hat, kann der ATM die Zahlung nicht abschließen. Obwohl der ATM technisch betriebsbereit ist, fehlt es an der nötigen Liquidität, um die Bitcoin an den Nutzer zu senden. Der Betreiber muss dann Kanäle rebalancieren oder neue Kanäle öffnen, um zusätzliche Inbound-Liquidität bereitzustellen [9].

Rebalancing und Liquiditätsmanagement

Rebalancing:

Das Rebalancing von Kanälen ist eine gängige Lösung für Liquiditätsprobleme. Dabei wird Liquidität zwischen verschiedenen Kanälen verschoben, um sicherzustellen, dass stets genügend Inbound-Liquidität vorhanden ist. Dieser Prozess kann jedoch ineffizient sein, da für jede Transaktion im Lightning-Netzwerk Gebühren anfallen, und häufiges Rebalancing die Betriebskosten eines ATMs erhöhen kann. Zudem kann der Rebalancing-Prozess Ausfallzeiten verursachen, während der ATM nicht funktionsfähig ist, was zu einem schlechten Nutzererlebnis führt [26].

Netzwerkauslastung und Rebalancing-Kosten:

In Szenarien mit hoher Netzwerkauslastung oder vielen Nutzern, die gleichzeitig den ATM verwenden, muss der Betreiber möglicherweise häufiger Rebalancing durchführen. Dies führt nicht nur zu erhöhten Betriebskosten, sondern kann auch die Netzwerkauslastung erhöhen, da jede Rebalancing-Operation eine On-Chain-Transaktion erfordert. Solche Kosten könnten auf die Nutzer umgelegt werden, indem der Betreiber höhere Gebühren verlangt oder die maximale Abhebemenge begrenzt, was wiederum die Attraktivität des ATMs verringert [24].

Größere Transaktionen und AMP (Atomic Multipath Payments)

Das Lightning-Netzwerk ist besonders für Mikrotransaktionen und kleinere Zahlungen optimiert. Wenn jedoch größere Beträge über einen ATM abgewickelt werden sollen, kann die Liquidität in den Zahlungskanälen schnell erschöpft sein. In solchen Fällen müsste die Zahlung in mehrere kleinere Zahlungen aufgeteilt werden, die durch verschiedene Kanäle geleitet werden. Dieser Mechanismus wird als Atomic Multipath Payment (AMP) bezeichnet.

AMP-Probleme:

Während AMP in vielen Fällen funktioniert, kann es zu Verzögerungen führen, wenn nicht alle beteiligten Kanäle über ausreichende Liquidität verfügen. Falls AMP fehlschlägt, wird die gesamte Transaktion abgebrochen, was zu einem negativen Nutzererlebnis führt [29].

Kanaleröffnungen und Liquiditätsbeschaffung

Ein weiteres Problem für Betreiber eines Lightning-ATMs besteht darin, neue Kanäle zu eröffnen, wenn zusätzliche Liquidität benötigt wird. Dies erfordert On-Chain-Transaktionen, die zeitaufwendig und kostspielig sind. Diese zusätzliche Verzögerung kann die Sofortigkeit, die das Lightning-Netzwerk normalerweise bietet, beeinträchtigen [16].

Faktoren der Liquiditätsverteilung im Netzwerk

Ein strukturelles Problem im Lightning-Netzwerk ist die ungleiche Verteilung der Liquidität. In Ballungsgebieten mit vielen ATMs könnten einige Betreiber große Liquiditätsreserven haben, während andere Betreiber nur begrenzten Zugang zu Liquidität haben. Dieses Ungleichgewicht führt zu einer Konkurrenz um die verfügbare Liquidität. Betreiber, die weniger liquide Kanäle haben, müssen möglicherweise höhere Routing-Gebühren zahlen, was die Kosten für den Betrieb eines ATMs erhöht [28].

Das Argument, dass in einer Stadt mit vielen ATMs Konkurrenz um die Liquidität entsteht, ist jedoch nicht direkt auf die geographische Dichte der ATMs zurückzuführen, sondern auf die Gesamtnutzung des Netzwerks und die spezifische Konfiguration der Kanäle. Die Konkurrenz entsteht eher durch die Verfügbarkeit der Liquidität und die Art der Verbindungen zwischen den Knotenpunkten im Netzwerk.

Sicherheitsbedenken im Lightning-Netzwerk

Ein weiteres potenzielles Problem sind Griefing-Angriffe, bei denen böswillige Akteure absichtlich Zahlungen initiieren, um die verfügbare Liquidität in Kanälen zu blockieren. Diese Angriffe können die Funktion des ATMs beeinträchtigen, da legitime Zahlungen nicht mehr durchgeführt werden können, solange die Kanäle blockiert sind. Der Einsatz von Watchtowers, die böswillige Aktivitäten überwachen, könnte hier Abhilfe schaffen [27].

3.2.2 Technische Voraussetzungen

Die technische Infrastruktur eines Bitcoin Lightning-ATMs erfordert sowohl seitens der Betreiber als auch der Nutzer eine Reihe von Voraussetzungen, um den reibungslosen Ablauf von Lightning-Zahlungen zu gewährleisten. Im Folgenden wird diese Infrastruktur sowohl aus der Perspektive des Betreibers als auch der des Nutzers getrennt beschrieben.

Perspektive des Betreibers

Internetverbindung:

Für den Betrieb eines Bitcoin Lightning-ATMs ist eine zuverlässige und stabile Internetverbindung unerlässlich. Der ATM-Betreiber muss sicherstellen, dass eine kabelgebundene oder drahtlose (4G/5G/WLAN) Verbindung besteht, um die kontinuierliche Verfügbarkeit des Systems zu gewährleisten. Da das Lightning-Netzwerk auf der Echtzeitverarbeitung von Zahlungen basiert, können Unterbrechungen in der Verbindung zu Verzögerungen oder abgebrochenen Zahlungen führen. Um Ausfallzeiten zu minimieren, wird empfohlen, ein Backup-System einzurichten, das im Falle eines Verbindungsverlusts automatisch übernimmt [9]. Eine niedrige Latenz ist ebenfalls von entscheidender Bedeutung, da diese die Effizienz und Geschwindigkeit der Transaktionen maßgeblich beeinflusst.

Lightning-Wallet und Liquiditätsmanagement:

Betreiber eines Bitcoin Lightning-ATMs benötigen eine Lightning-Wallet, die in das ATM-System integriert ist und ausreichend Liquidität auf der Inbound- und Outbound-Seite bereitstellt. Diese Wallet sollte fähig sein, dynamische Rebalancing-Prozesse durchzuführen, um jederzeit ausreichende Liquidität sicherzustellen und Engpässe zu vermeiden [26]. Implementierungen wie LND oder c-lightning, die auf einem Bitcoin Full Node basieren, werden häufig für solche Zwecke verwendet [25, 24].

Lightning Node:

Eine voll funktionsfähige Lightning Node, die kontinuierlich online ist und regelmäßig aktualisiert wird, ist eine weitere Voraussetzung für den Betrieb eines Bitcoin Lightning-ATMs. Diese Node ermöglicht das Öffnen von Zahlungskanälen, das Validieren von Zahlungen sowie die Gewährleistung ausreichender Liquidität. Um Ausfallzeiten zu minimieren, wird empfohlen, eine Backup-Node zu betreiben [27]. Für Betreiber bieten sich zudem Lösungen wie LNbits an, die eine All-in-One-Lösung darstellen und einige der administrativen Aufgaben übernehmen können [32]. Beim offiziellen Betrieb ist jedoch eine regulatorische Sicherheit zu gewährleisten.

Stromversorgung und physische Sicherheit:

Eine unterbrechungsfreie Stromversorgung (USV) ist notwendig, um den durchgehenden Betrieb des ATMs zu gewährleisten, insbesondere bei Stromausfällen [24]. Die Hardware des ATMs sowie die angeschlossene Lightning Node sollten durch eine USV gesichert werden, um Datenverlust und Netzwerkunterbrechungen zu vermeiden. Zudem ist eine robuste physische Sicherheit erforderlich, um den ATM vor Vandalismus und Diebstahl zu schützen. Sicherheitsmaßnahmen wie Kameras und Alarmsysteme können das Risiko physischer Angriffe verringern.

Datensicherheit und Zahlungsintegrität:

Zur Sicherstellung der Zahlungssicherheit muss der Betreiber gewährleisten, dass alle Daten, die zwischen dem ATM und den Lightning-Nodes ausgetauscht werden, verschlüsselt sind, um Man-in-the-Middle-Angriffe zu verhindern [27]. Der Einsatz von Watchtowers kann die Sicherheit weiter erhöhen, indem potenziell böswillige Aktivitäten erkannt und gestoppt werden [28].

Perspektive des Nutzers

Internetverbindung:

Auch für den Nutzer ist eine stabile Internetverbindung erforderlich, um mit dem Lightning-ATM interagieren zu können. Mobilfunkstandards wie 4G oder 5G, die auf modernen Smartphones verfügbar sind, bieten ausreichende Bandbreite und Latenz für die Echtzeit-Abwicklung von Zahlungen. Verbindungsunterbrechungen könnten zu Verzögerungen oder dem Abbruch der Transaktion führen.

Lightning-Wallet:

Der Nutzer benötigt eine kompatible Lightning-Wallet auf seinem mobilen Endgerät, um Bitcoin-Transaktionen mit dem ATM durchzuführen. Diese Wallet dient als Schnittstelle zum Lightning-Netzwerk und muss in der Lage sein, Zahlungen sicher und in Echtzeit abzuwickeln. Wallet-Anwendungen sind in der Regel für gängige Betriebssysteme wie Android und iOS verfügbar.

Hardwareanforderungen:

Ein mobiles Endgerät, wie ein Smartphone oder Tablet, mit einer funktionierenden Kamera ist notwendig, um QR-Codes zu scannen und somit den Transaktionsprozess zu initialisieren [16]. Zudem muss das Gerät über ausreichend Rechenleistung und Speicher verfügen, um die Lightning-Wallet-App reibungslos ausführen zu können.

Lightning Node (Optional):

Nutzer müssen im Allgemeinen keine eigene Lightning Node betreiben, um einen ATM zu nutzen. Für technisch versierte Nutzer besteht jedoch die Möglichkeit, eine eigene Node zu betreiben, um Routing-Gebühren zu reduzieren und eine bessere Kontrolle über die Zahlungen zu haben. Dies kann auch die Privatsphäre erhöhen, da weniger Drittanbieter in den Zahlungsprozess involviert sind [29].

Sicherheitsmaßnahmen:

Zur Absicherung der Lightning-Wallet sollten Nutzer auf die Verwendung von Zwei-Faktor-Authentifizierung (2FA) sowie die Verschlüsselung der Wallet-Daten achten. Eine sichere Netzwerkverbindung ist erforderlich, um potenzielle Angriffe über unsichere WLAN-Verbindungen zu vermeiden.

3.2.3 Fehleranfälligkeit

Ein Lightning-ATM bietet die Möglichkeit, Fiat-Währungen in Bitcoin umzuwandeln, indem eingeworfene Münzen direkt in eine Lightning URL Withdrawal (LNURLw) umgerechnet werden. Obwohl die zugrunde liegende Technologie innovative Vorteile in der Mikrotransaktionsabwicklung bietet, existieren dennoch mehrere potenzielle Fehlerquellen, die die Zuverlässigkeit eines solchen Systems beeinträchtigen können. Im Folgenden werden einige der kritischen Punkte diskutiert.

Fehlerhafte Zahlungen und ungültige LNURLws

Eine häufig auftretende Herausforderung bei der Nutzung von Lightning-ATMs ist das Szenario, in dem eine Zahlung nicht erfolgreich durchgeführt wird. Der ATM erstellt in der Regel eine LNURLw, die dem Benutzer über einen QR-Code angezeigt wird. Der Benutzer scannt diesen Code mit einer Lightning-Wallet und die Zahlung wird initiiert. Sollte die Zahlung jedoch aus irgendeinem Grund fehlschlagen, kann dies zu einem Problem führen. Beispielsweise könnte eine schlechte Netzwerkverbindung oder ein Fehler im Lightning-Netzwerk dazu führen, dass die Zahlung nicht bestätigt wird, obwohl die Münzen bereits im ATM eingezahlt wurden.

Um solche Probleme zu vermeiden, sollte die Software des ATMs eine sichere Transaktionsabwicklung gewährleisten. Wenn die Zahlung nicht erfolgreich ist, muss die LNURLw ungültig gemacht und die eingezahlten Münzen an den Benutzer zurückgegeben werden. Ein fehlender Mechanismus zum Ungültigmachen von LNURLws könnte zu einer Situation führen, in der das Geld des Benutzers im System „verschwindet“, da die Zahlung nicht abgeschlossen wird, die Münzen aber dennoch nicht zurückgegeben werden.

Mangelnde Liquidität

Ein weiteres potenzielles Problem ist das Szenario, in dem der ATM keine ausreichende Liquidität im Lightning-Netzwerk zur Verfügung hat. Da der Lightning-ATM auf Mikrozahlungen über das Lightning-Netzwerk angewiesen ist, könnte es passieren, dass der Betreiber des ATMs nicht über genügend verfügbare Liquidität verfügt, um die Transaktion durchzuführen. In diesem Fall wäre es dem Benutzer unmöglich, die eingezahlten Münzen in Bitcoin umzuwandeln, da keine Route zum Zahlungsempfänger gefunden werden kann.

Ein möglicher Ansatz zur Vermeidung dieses Problems wäre die Implementierung eines Mechanismus, der die Liquidität des ATMs überprüft, bevor eine LNURLw erstellt wird. Sollte keine ausreichende Liquidität vorhanden sein, müsste der ATM den Benutzer darüber informieren und ihm die Möglichkeit geben, die Transaktion abzubrechen und die eingeworfenen Münzen zurückzuerhalten.

Fehlende Internetverbindung

Die Internetverbindung ist entscheidend für den Betrieb eines Lightning-ATMs, da die Zahlung über das Lightning-Netzwerk in Echtzeit erfolgen muss. Wenn keine Internetverbindung besteht, kann der ATM keine LNURLws generieren. Dies stellt ein erhebliches Problem dar, da der Benutzer möglicherweise Münzen in den Automaten eingeworfen hat, die Transaktion jedoch nicht abgeschlossen werden kann.

In einem solchen Fall könnte ein Fehler im Timing der Transaktion auftreten, der zu unvollständigen Zahlungsvorgängen führt. Dies könnte zum Beispiel bedeuten, dass die Zahlung im Backend-System als „in Bearbeitung“ registriert wird, jedoch nie abgeschlossen wird, weil keine Internetverbindung vorhanden ist. Solche Fehlerszenarien können dazu führen, dass Benutzer ihre Zahlung nicht erhalten und ihre Münzen verlieren, wenn der ATM nicht korrekt darauf vorbereitet ist, den Ausfall des Internets zu handhaben. Ein Lightning-ATM sollte daher mit Mechanismen ausgestattet sein, die solche Szenarien erkennen und die eingezahlten Münzen sicher an den Benutzer zurückgeben.

Unbefugtes Scannen des QR-Codes

Ein weiteres kritisches Problem besteht in der Möglichkeit, dass ein unbefugter Dritter den QR-Code scannt, bevor der eigentliche Benutzer die Zahlung durchführen kann. Da der Lightning-ATM die LNURLw durch einen QR-Code bereitstellt, kann theoretisch jede Person mit einem Lightning-Wallet, die diesen Code scannt, die Zahlung beanspruchen. Ein umfassender Schutz gegen dieses Szenario ist technisch schwierig zu realisieren, da die Lightning-LNURLs öffentlich lesbar sind, sobald sie als QR-Code dargestellt werden.

Ein Ansatz, um dieses Risiko zu verringern, wäre die Verwendung einer Sichtschutzfolie auf dem Bildschirm des ATMs, die den QR-Code nur bei direkter Sicht auf den Bildschirm erkennbar macht. Diese Methode würde jedoch keinen 100-prozentigen Schutz bieten. Letztendlich liegt es in der Verantwortung des Benutzers, sicherzustellen, dass niemand in unmittelbarer

Nähe ist, der den QR-Code schneller scannen könnte. Zudem sind die QR-Codes nur einmalig gültig, sodass sie nicht zu einem späteren Zeitpunkt erneut beansprucht werden können, was das Risiko eines Missbrauchs reduziert. Diese Problematik ähnelt der Verantwortung des Nutzers bei traditionellen Geldautomaten, bei denen sichergestellt werden muss, dass keine Dritten bei der PIN-Eingabe zusehen.

4 Konzeptionierung des Lightning-ATM

Die Konzeption eines Lightning-ATMs basiert auf vorhandenen Projekten und Ressourcen, die relevante technische Ansätze und Lösungen bieten. In diesem Kapitel werden die verwendeten Hardware-Komponenten sowie die Software-Integration detailliert beschrieben. Ziel ist es, ein funktionales und effizientes Konzept für die Entwicklung eines Lightning-ATMs zu erläutern.

4.1 Vorhandene Ressourcen als Grundlage

In den letzten Jahren haben sich verschiedene Projekte mit der Entwicklung von Bitcoin-basierten ATMs beschäftigt, die das Lightning-Netzwerk unterstützen. Zu den vielversprechenden Konzepten gehören der Bleskomat [33] sowie das Open-Source-Projekt von 21isenough [34]. Diese Projekte sind zwar primär als Konzepte gedacht und nicht direkt für den Massenmarkt ausgelegt, bieten jedoch wertvolle Erkenntnisse und Ansätze, die in der Entwicklung von Lightning-ATMs berücksichtigt werden können.

Insbesondere liefern sie nützliche Informationen über die Interaktion von Hardware-Komponenten sowie Schaltpläne, die ein solides Verständnis der Grundlagen eines Lightning-ATMs ermöglichen. Die im Projekt 21isenough [34] veröffentlichten offenen Software- und Hardware-Konzepte können als Ausgangspunkt für die eigene Entwicklung eines ATMs dienen.

Diese von 21isenough [34] entwickelte Hardware-Konfiguration für einen Lightning-ATM besteht aus einer Reihe von standardisierten Komponenten. Diese Komponenten lassen sich in zwei Versionen einteilen: eine kompakte und eine erweiterte Version. Die folgende Tabelle zeigt die wesentlichen Hardware-Komponenten, die für die Konstruktion eines Lightning-ATMs verwendet werden können:

Komponente	**Beschreibung**	**Wichtige Informationen**
Raspberry Pi Zero WH	Ein kompakter Einplatinencomputer, der als Hauptsteuerungseinheit dient	1 GHz CPU, 512 MB RAM, integriertes WLAN und Bluetooth
Micro SD Card	Speichermedium für das Betriebssystem und die Software	Mindestens 16 GB Speicher, Class 10 empfohlen
DC-DC Step Up Power Module (5V->12V)	Spannungswandler, der die Spannung von 5V auf 12V erhöht	Effizienz von über 90 %
Multi Coin Acceptor	Münzprüfer, der verschiedene Münzarten akzeptiert	Akzeptiert Münzen verschiedener Währungen, einstellbare Akzeptanzrate

Komponente	Beschreibung	Wichtige Informationen
Display mit HAT (e-paper ink display)	Anzeigebildschirm, der den aktuellen Status des ATMs anzeigt	2.7"bis 5.83Ë-Paper Display mit HAT für den Raspberry Pi
Micro USB Kabel	Verbindungskabel zur Stromversorgung und Datenübertragung	Standard Micro-USB mit 5V Versorgung
Jumper Kabel	Verbindungskabel zur Kommunikation zwischen den Komponenten	Verschiedene Längen und Farben
Netzteil	Stromversorgung für den Raspberry Pi und andere Komponenten	5V, 2.5A Netzteil für stabile Versorgung
Gehäuse (3D-gedruckt aus PLA Filament)	Schutzhülle für die Komponenten	Anpassbare Größe, um alle Komponenten zu schützen
Erweiterte Version		
Push Button mit LED (3V)	Taster zur Benutzereingabe, optional mit LED	Wird zur Bestätigung von Transaktionen genutzt
Lockout Relay Module (HW-482 oder KY-019)	Relaismodul zum Steuern von Stromkreisen	Geeignet für die Sicherung von Stromkreisen bei Fehlfunktionen
Raspberry Pi Kamera 5MP OV5647	Kamera zur Erfassung von Bildern oder QR-Codes	5MP Kamera mit einstellbarem Fokus
ATM Gehäuse	Robustes Gehäuse, speziell für die ATM-Anwendung	Stahl oder Kunststoff, je nach Anforderung

Tabelle 4.1: Hardware des ATM-Projektes von 21isenough [34]

4.2 Hardware

Die nachstehende Tabelle bietet eine detaillierte Übersicht der im Lightning-ATM verwendeten Hardware-Komponenten. Neben den bereits bekannten Bauteilen wurden auch zusätzliche Komponenten wie NPN-Transistoren, 12VDC-Relays, 1k Ω-Widerstände und ein eigens entworfenes PCB-Board hinzugefügt, um die Effizienz und Stabilität des Systems zu gewährleisten.

Komponente	Beschreibung	Technische Details	Preis
Raspberry Pi 4B [35]	Einplatinencomputer für die Steuerung des ATMs. Aufgrund der höheren Rechenleistung gegenüber dem Raspberry Pi Zero WH für rechenintensive Aufgaben geeignet.	1.5 GHz Quad-Core, bis zu 8 GB RAM, Micro-HDMI, GPIO-Pins, WLAN	60,49€
HAMTYSAN 10.1 Zoll Touch Display [36]	Touch-fähiges Display zur Interaktion mit dem ATM. Es bietet eine deutlich bessere Benutzererfahrung als kleinere, nicht-touchfähige Displays.	Touch-fähig, Stromversorgung über USB, Verbindung über Micro-HDMI	59.99€
Heschen HS-0530B 12VDC Solenoid [37]	Solenoid zur Handhabung des Münzflusses welcher über das Raspberry Pi gesteuert wird.	10mm Hub, 12VDC und 5 Newton Linearkraft	5.99€
KungfuMall 12VDC Solenoid [38]	Solenoid zur Handhabung des Münzflusses welcher über das Raspberry Pi gesteuert wird.	35mm Hub, 12VDC und 8A Stromstärke	15.99€
Multi Coin Acceptor [39]	Münzprüfer, der Münzen verschiedener Währungen akzeptiert und zur Münzerkennung und -validierung dient.	Akzeptiert mehrere Währungen, einstellbare Akzeptanzraten	22.85€
12V 20A Netzteil [40]	Hauptstromversorgung des Systems. Aufgrund der hohen Lasten für Bauteile wie Solenoids wird ein Netzteil mit hoher Ausgangsleistung benötigt.	12V, 20A Ausgangsleistung	22.99€

Komponente	Beschreibung	Technische Details	Preis
DC-DC-Spannungswandler [41]	Spannungswandler zur Versorgung der 5V-Komponenten. Wandelt 12V auf 5V herunter, um den Raspberry Pi und andere 5V-Komponenten zu betreiben.	12V auf 5V, 3A Ausgang	6.99€
Breadboard [42]	Experimentierplattform zur Prototypisierung elektronischer Schaltungen. Ermöglicht den Aufbau von Schaltungen und das Testen von Komponenten.	Zwei Stromkreisläufe, flexible Schaltungsoptionen	2.99€
2x NPN Transistor 2N2222 [43]	Transistor zur Steuerung von Relays und Solenoids. Verhindert Rückkopplungen, die den Raspberry Pi beschädigen könnten.	12V, max. 600 mA, 40V Spannungstoleranz	0.20€
2x 12VDC Relays [44]	Relays zur Steuerung der Solenoids, um den Münzfluss zu steuern. Werden durch Transistoren gesteuert.	12VDC, 10A Schaltstrom	3.20€
2x 1kΩ Widerstände [45]	Widerstände zur Strombegrenzung und Schutz der Transistoren und des Raspberry Pi vor Überlastung.	1k Ω, 5% Toleranz	0.20€
Jumperkabel	Verbindungskabel zur Verdrahtung der Komponenten auf dem Breadboard oder PCB.	Verschiedene Längen, 2.54mm Steckverbindung	1.49€
PCB Board [4.4]	Speziell entworfene Platine, um den Kabelsalat zu vermeiden und das Breadboard zu ersetzen.	2-lagige PCB, für alle relevanten Komponenten ausgelegt	1.80€

Komponente	Beschreibung	Technische Details	Preis
MDF Platten und Kleinteile	MDF Platten als Grundmaterial für das Gehäuse sowie alle möglichen Kleinteile.	16mm MDF Platten, 3mm Fichtenpressplatte für Münzflusskonstrukt, 2- und 3-Punkt Winkel, Diverse schrauben wie z.B. 24 Stückx3mmx16mm für das Gehäuse, Werkzeug, Unterlegscheiben usw.	ca. 65.00€
Gesamtkosten			**270,16€**

Tabelle 4.2: Hardwareauswahl für das Konzept

Detaillierte Beschreibung der zusätzlichen Komponenten

Raspberry Pi 4B

Das Herzstück des Lightning-ATMs bildet ein Raspberry Pi 4B, der als Einplatinencomputer fungiert. Im Vergleich zu älteren Modellen wie dem Raspberry Pi Zero WH bietet der Raspberry Pi 4B eine deutlich höhere Rechenleistung. Diese ist notwendig, um ein ansprechendes und responsives Frontend zu unterstützen sowie rechenintensive Aufgaben wie das Betreiben einer Bitcoin-Node zu ermöglichen. Mit einem Quad-Core ARM Cortex-A72 Prozessor, der mit 1.5 GHz getaktet ist, und einer Unterstützung für bis zu 8 GB RAM, bietet der Raspberry Pi 4B genügend Leistung für komplexe Anwendungen. Zudem verfügt er über Micro-HDMI-Anschlüsse, GPIO-Pins und integriertes WLAN, was eine flexible Installation und Steuerung von externen Hardware-Komponenten ermöglicht [35].

HAMTYSAN 10.1 Zoll Touch Display

Um ein nutzerfreundliches und interaktives Erlebnis zu ermöglichen, wurde ein 10.1 Zoll Touch Display gewählt. Im Vergleich zu kleineren, nicht-touchfähigen Displays, wie sie in früheren Projekten verwendet wurden, bietet dieses Display eine deutlich verbesserte Benutzerfreundlichkeit. Das Display wird über USB-A mit Strom versorgt und über den Micro-HDMI-Anschluss mit dem Raspberry Pi verbunden. Dank der nativen Unterstützung des Displays durch den Raspberry Pi entfällt die aufwändige Programmierung der Display-Anbindung, was den Entwicklungsprozess vereinfacht [36].

Multi Coin Acceptor

Der Multi Coin Acceptor ist ein wesentlicher Bestandteil des ATMs, der die Aufgabe übernimmt, Münzen zu erkennen und zu validieren. Diese Komponente stammt aus dem Projekt von 21isenough [34] und wurde übernommen, da sie als Standardbauteil speziell für diesen Zweck entwickelt wurde. Der Münzprüfer kann so konfiguriert werden, dass er Münzen unterschiedlicher Währungen akzeptiert und ist somit flexibel einsetzbar [39].

12V 20A Netzteil

Das gesamte System wird von einem 12V 20A Netzteil mit Strom versorgt. Die Wahl eines 12V-Systems ergibt sich aus der Tatsache, dass viele der im ATM verwendeten Komponenten, insbesondere die Solenoids, eine 12V-Versorgung benötigen. Ein Netzteil mit 20A Ausgangsleistung ist notwendig, um die hohen Lasten dieser Komponenten zu bewältigen [40].

DC-DC-Spannungswandler

Da einige der Komponenten, wie der Raspberry Pi, mit 5V betrieben werden müssen, wird ein DC-DC-Spannungswandler verwendet, um die 12V Eingangsspannung auf 5V herunterzuregeln. Dieser Wandler liefert eine Ausgangsleistung von bis zu 3 Ampere, was ausreicht, um den Raspberry Pi und weitere 5V-Komponenten zu betreiben [41].

Breadboard

Für die Prototypisierung und das Testen von Schaltungen wird ein Breadboard eingesetzt. Ein Breadboard ermöglicht es, Schaltungen schnell und flexibel zu testen, bevor diese in eine feste Platine überführt werden. Es verfügt über zwei getrennte Stromschienen und zahlreiche Verbindungen, die zur Erstellung von Schaltungen genutzt werden können [42].

Solenoids

Zur Steuerung des Münzflusses wurden Solenoids verwendet. Diese elektromechanischen Bauteile bewegen einen Metallbolzen linear, was eine schnelle und kraftvolle Bewegung ermöglicht. Solenoids eignen sich besonders gut für Anwendungen, bei denen nur zwei Positionen erforderlich sind, wie es bei der Steuerung des Münzflusses der Fall ist. Die beiden im Projekt verwendeten Solenoids arbeiten mit einer 12V-Stromversorgung und benötigen unter Maximallast 8A bzw. 2A. Ihre Kraft und Bewegung sind abhängig von der Größe der elektromagnetischen Spule sowie der Stromstärke und Spannung [38, 37].

NPN Transistoren 2N2222

Die NPN-Transistoren 2N2222 sind essenziell für die Steuerung der 12VDC-Relays und der Solenoids. Ein Transistor fungiert allgemein als elektronischer Schalter oder Verstärker, der es ermöglicht, ein kleines Eingangssignal zu nutzen, um ein größeres Ausgangssignal zu schalten. In diesem Fall ist der Transistor erforderlich, da der Raspberry Pi nur kleine Ströme und Spannungen bereitstellen kann, während die Relays größere Ströme benötigen [43].

Der Transistor nimmt das schwache Signal vom Raspberry Pi an der Basis (B) auf und verstärkt dieses, um den nötigen Strom für das Relay an den Kollektor (C) weiterzuleiten. Darüber hinaus schützen die Transistoren das System vor Rückkopplungen, die durch die Solenoids verursacht werden. Diese Rückkopplungen entstehen, wenn der Metallkern der Solenoids zurückfällt und dabei eine Gegen-EMK (elektromotorische Kraft) erzeugt. Durch die Verwendung des Transistors wird diese Gegen-EMK kontrolliert und potenzieller Schaden an den empfindlichen Komponenten wie dem Raspberry Pi vermieden.

Der Transistor 2N2222 eignet sich für diese Anwendung, da er eine Schaltleistung von bis zu 600 mA bei 12V besitzt und Spannungen bis zu 40V tolerieren kann.

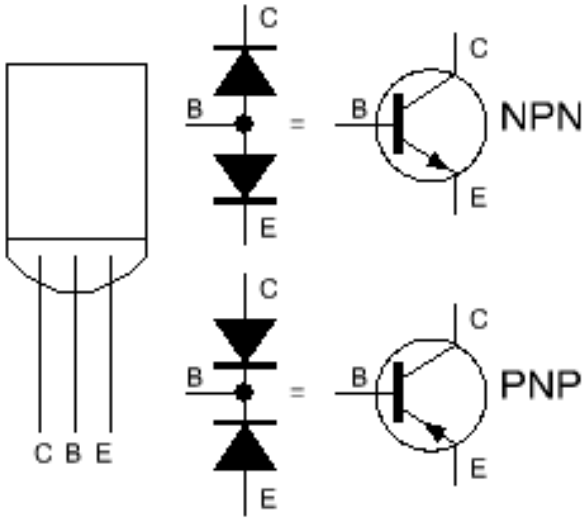


Abbildung 4.1: Schaltplan des NPN-Transistors 2N2222 [46]

12VDC Relays

Die 12VDC-Relays dienen der Steuerung der Solenoids, die im ATM den Münzfluss regulieren. Relays bieten eine effektive Möglichkeit, einen Hochstromkreis mit einem Niedrigstromkreis zu schalten. Sie werden durch den Transistor gesteuert, der das schwache Signal des Raspberry Pi verstärkt, um die Solenoids zu betreiben [44].

Das Relay übernimmt die Funktion eines mechanischen Schalters, der einen 12V-Stromkreis öffnet oder schließt, je nachdem, ob das Signal vom Transistor kommt. Durch die galvanische Trennung der beiden Stromkreise wird sichergestellt, dass der Raspberry Pi nicht direkt mit den Hochstromkomponenten verbunden ist, was die Zuverlässigkeit und Sicherheit erhöht.

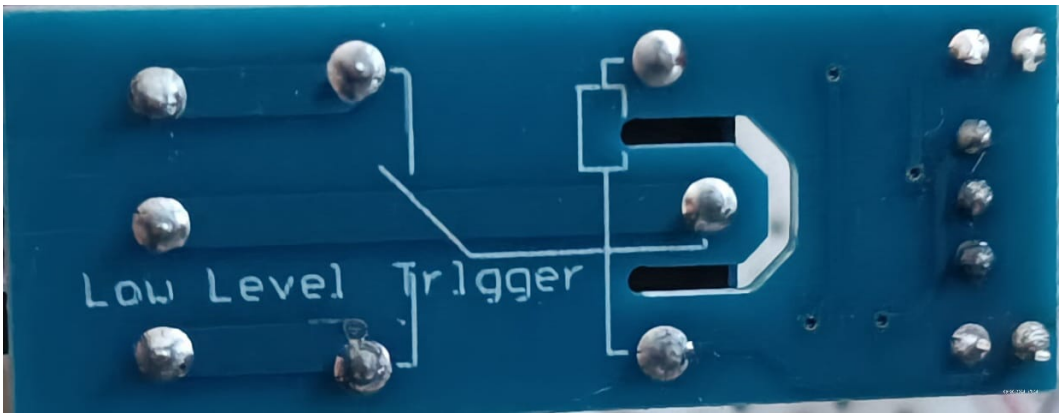


Abbildung 4.2: Schaltplan eines 12VDC Relays, gesteuert durch den Raspberry Pi und Transistor [44]

1kΩ Widerstände

Die 1k Ω-Widerstände werden zur Strombegrenzung in der Schaltung verwendet. Sie spielen eine wesentliche Rolle in der Sicherstellung, dass der Transistor nicht überlastet wird, wenn der Strom vom Raspberry Pi zur Basis des Transistors fließt. Ohne diese Widerstände könnte der Basisstrom des Transistors zu hoch werden, was den Transistor und möglicherweise auch den Raspberry Pi beschädigen könnte [45].

Die Widerstände fungieren zudem als Spannungsstabilisatoren und tragen dazu bei, dass plötzliche Spannungsspitzen, die bei der Abschaltung der Solenoids auftreten können, die Schaltung nicht beeinträchtigen.

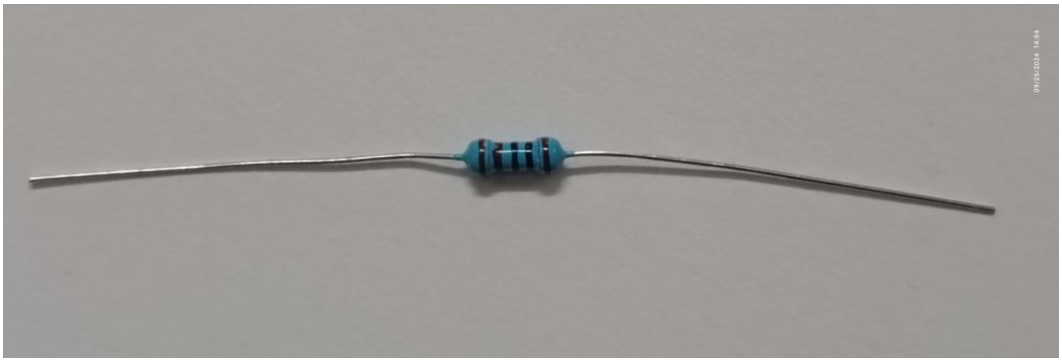


Abbildung 4.3: Verwendeter 1kΩ Widerstand

PCB Board

Um den Aufbau des Systems zu optimieren und die Verdrahtung zu vereinfachen, wurde ein PCB (Printed Circuit Board) entwickelt, dass das Breadboard ersetzt. Das PCB ermöglicht eine sauberere und stabilere Integration aller relevanten Komponenten, einschließlich der Transistoren, Relays und Widerstände, und sorgt dafür, dass der Aufbau des Systems weniger anfällig für lose Verbindungen ist.

Das PCB trägt zur Reduzierung der Unübersichtlichkeit bei, indem es alle Komponenten auf einer festen Platine integriert, was die Fehleranfälligkeit verringert und die Zuverlässigkeit des Systems erhöht. Es ist speziell auf die Anforderungen des Lightning-ATMs ausgelegt und ermöglicht eine einfache Verdrahtung der wichtigsten Bauteile.

Das in dieser Arbeit entwickelte PCB entsteht mithilfe der Software EasyEDA [47] und wird über einen Leiterplattenhersteller [48] gefertigt. Die PCB Dateien können unter [49] eingesehen werden. Dabei werden spezifische Anforderungen berücksichtigt, die das PCB erfüllen muss. Da das Netzteil bis zu drei Stromkreise bereitstellt, ist das Design des PCBs so ausgelegt, dass die Stromlast der Solenoids auf zwei getrennte Stromkreise verteilt wird, um eine Überhitzung einzelner Kabel und Verbindungen zu vermeiden. Die Leiterbahnen auf dem PCB werden in ihrer Dicke an den potenziellen Stromfluss angepasst: Während die 12V-Leitungen und Massebahnen deutlich breiter dimensioniert sind, um höheren Stromspitzen standzuhalten, sind weniger stromintensive Verbindungen schmaler gestaltet. Letztere dienen hauptsächlich der Übermittlung von Signalen an das Raspberry Pi und sind somit keiner hohen Belastung ausgesetzt (siehe Abbildung 4.5).

Zudem wird das PCB so gestaltet, dass es direkt auf den 40-Pin-Anschluss des Raspberry Pi passt, wodurch umständliche Verbindungen zwischen dem PCB und dem Raspberry Pi vermieden werden. Das PCB bildet eine direkte Abbildung des im Rahmen der Tests (siehe Kapitel 4.3.2) erstellten Schaltkreises. Bei der Erstellung eines PCBs ist es entscheidend, dass die verschiedenen Stromkreise klar voneinander getrennt bleiben, um Kurzschlüsse zu vermeiden. Die auf dem PCB verbauten Komponenten wie Widerstände, Transistoren und eventuelle Kabel werden manuell gelötet. Zwar besteht die Möglichkeit, PCBs direkt mit den entsprechenden Komponenten zu versehen, doch bedeutet dies in diesem Fall einen unverhältnismäßig hohen Aufwand. Für zukünftige Weiterentwicklungen könnte dies jedoch in Betracht gezogen werden, um die Anzahl offener Kabelverbindungen weiter zu reduzieren und das Produkt optisch und funktional zu optimieren.

Zusätzlich werden alle relevanten Lötpunkte auf dem PCB beschriftet, und das Board wird durch die Integration eines projektbezogenen Bitcoin-Logos visuell ansprechend gestaltet (siehe Abbildung 4.4). Das PCB ist als doppelseitige Leiterplatte konzipiert, mit Leiterbahnen auf der Vorder- und Rückseite, wodurch eine höhere Packungsdichte und somit eine platzsparende Konstruktion ermöglicht wird. Die Leiterbahnen sind in Abbildung 4.5 mit den Farben Rot (obere Schicht) und Blau (untere Schicht) veranschaulicht. Sämtliche Kenntnisse zur Erstellung von PCBs werden im Rahmen dieser Arbeit neu erlernt und angewendet.

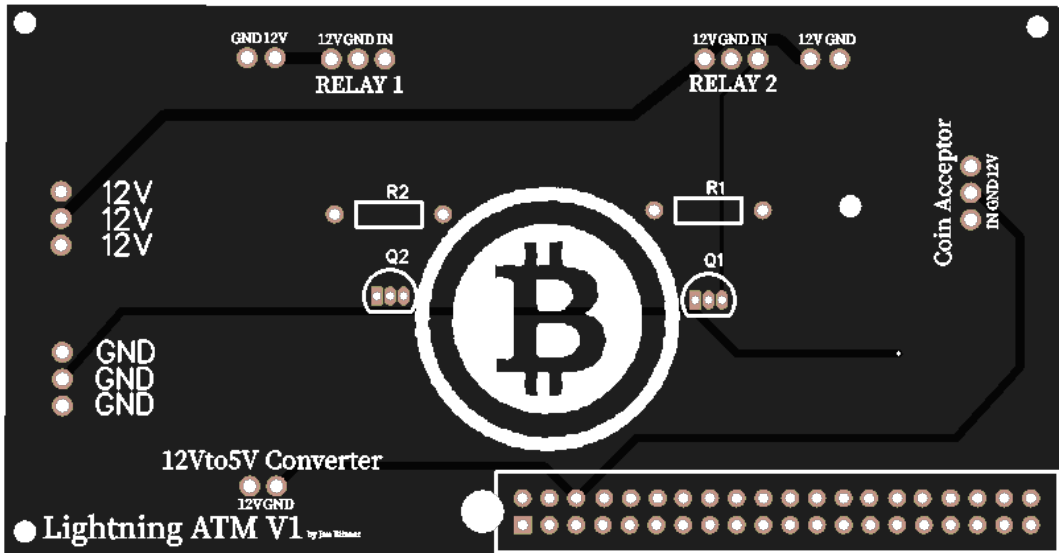


Abbildung 4.4: Design des PCB

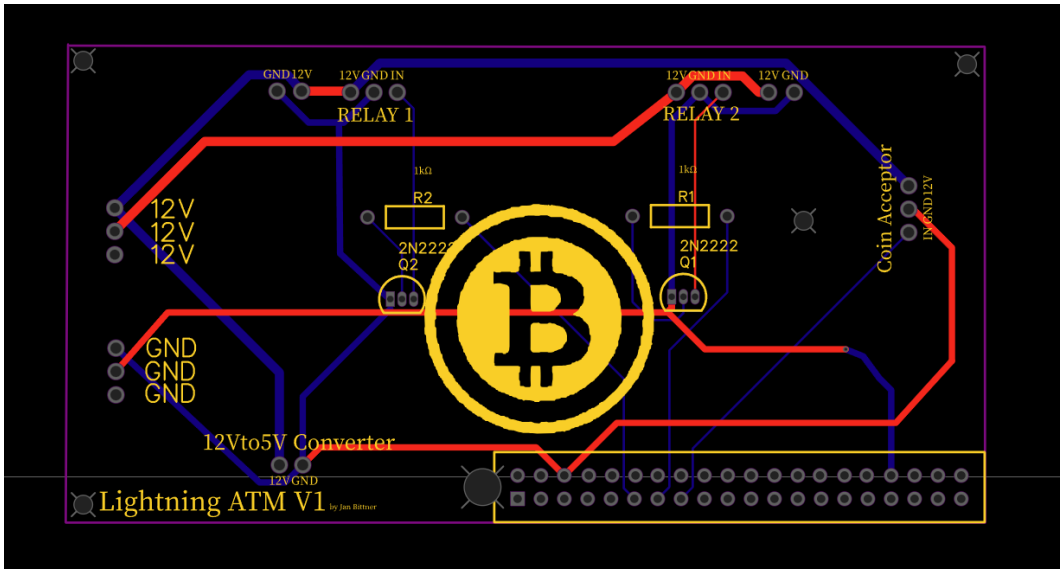


Abbildung 4.5: Leiterbahnen im PCB

Weitere Erläuterungen zu den Komponenten und Gesamtkosten

Die Verwendung von Widerständen in der Schaltung dient primär dazu, eine Überlastung der Transistoren zu verhindern und den sicheren Betrieb der Schaltung zu gewährleisten. Widerstände übernehmen eine wichtige Rolle bei der Stabilisierung der Spannung, indem sie plötzliche Spannungsspitzen abfangen, die insbesondere beim Abschalten der Solenoids auftreten können. Diese Schutzfunktion trägt maßgeblich zur Langlebigkeit der Transistoren sowie der Steuerkomponenten, wie dem Raspberry Pi, bei.

Im Gegensatz zu bestehenden Projekten, wie dem von 21isenough [34], in denen eine 5V-Versorgung als Basis dient, wurde in dieser Arbeit eine 12V-Versorgung als Standard gewählt. Dies begründet sich insbesondere durch den Bedarf mehrerer Bauteile, insbesondere der

Solenoids und Relays, die höhere Ströme bei 12V erfordern. Eine 12V-Spannungsbasis ermöglicht eine effizientere Schaltung dieser hohen Lasten, ohne dass es zu einer Überlastung des Netzteils oder zu Spannungseinbrüchen kommt.

Zusätzlich wird die 12V-Spannung mittels eines DC-DC-Spannungswandlers auf 5V reduziert, um empfindliche Komponenten, wie den Raspberry Pi, sicher zu betreiben. Dieser Ansatz gewährleistet nicht nur eine stabilere Stromversorgung der 5V-Komponenten, sondern schützt auch vor Überlastungen. Der Spannungswandler sorgt für eine konstant zuverlässige 5V-Stromversorgung, während die leistungsintensiveren 12V-Komponenten parallel und unabhängig betrieben werden können.

Die Gesamtkosten der im Prototyp verwendeten Hardwarekomponenten belaufen sich auf 270,16 €. Damit bleiben die Ausgaben relativ niedrig, was auch für eine potenzielle Massenproduktion von Vorteil ist. Die teuersten Einzelteile sind der Raspberry Pi 4B (60,49 €) und das Touch Display (59,99 €), welche aufgrund ihrer Funktionalität und Benutzerfreundlichkeit eine wichtige Rolle im System einnehmen. Kleinere Bauteile wie Transistoren, Widerstände und Relays tragen ebenfalls zur Funktionsfähigkeit bei, ohne das Budget zu belasten.

Da es sich um einen ersten Prototyp handelt, bietet das Konzept noch Potenzial für Kostenoptimierungen und Effizienzsteigerungen in der weiteren Entwicklung.

4.3 Anforderungen

Ein wesentlicher funktionaler Aspekt des entwickelten Lightning-ATMs besteht darin, eingezahlte Münzen vorübergehend zu speichern, bis die zugehörige Zahlungstransaktion über das Lightning-Netzwerk erfolgreich abgeschlossen ist. Dies gewährleistet eine sichere Handhabung der eingezahlten Gelder. Im Falle eines erfolgreichen Zahlungsvorgangs werden die Münzen in den Tresor des Automaten geleitet. Sollte die Transaktion jedoch abgebrochen oder aus einem anderen Grund nicht erfolgreich durchgeführt werden, ist das System so konzipiert, dass die eingezahlten Münzen über den Münzausgabeschacht an den Benutzer zurückgegeben werden. Diese Funktionalität stellt sicher, dass keine Gelder verloren gehen oder ohne erfolgreiche Zahlungstransaktion im ATM verbleiben. Die Lösung trägt somit entscheidend zur Transaktionssicherheit bei.

Ein weiteres Ziel der Entwicklung ist die Kompaktheit des Lightning-ATMs. Der Automat sollte eine Größe aufweisen, die es ermöglicht, ihn einfach zu transportieren, um ihn flexibel für Testzwecke und Demonstrationen an verschiedenen Standorten einsetzen zu können. Ein geringes Gewicht und eine übersichtliche Größe stellen sicher, dass der ATM mühelos auf- und abgebaut werden kann. Dies erleichtert die Demonstration des Systems in unterschiedlichen Umgebungen, wie auf Fachmessen oder Konferenzen, und fördert die Mobilität und Flexibilität des Automaten. Dabei war die Handhabung ein zentraler Faktor: Der ATM sollte lediglich über ein einziges Stromkabel angeschlossen werden können, ohne dass zusätzliche Kabelverbindungen erforderlich sind. Dies trägt zu einer benutzerfreundlichen Installation und einem einfachen Transport bei.

Ein weiterer wichtiger Aspekt ist die Netzwerkkonfiguration des Lightning-ATMs. Da der ATM flexibel in verschiedenen Umgebungen eingesetzt werden soll, wurde eine WLAN-Anbindung vorgesehen. Diese muss sich unkompliziert konfigurieren lassen, sodass der ATM schnell und ohne großen technischen Aufwand an unterschiedliche Netzwerke und Standortanforderungen angepasst werden kann. Eine einfache und intuitive Konfiguration der WLAN-Verbindung minimiert den Zeitaufwand bei der Inbetriebnahme des Geräts und gewährleistet, dass der ATM in verschiedensten Umgebungen ohne größere technische Eingriffe einsatzbereit ist.

Darüber hinaus spielt die visuelle und akustische Gestaltung des Automaten eine bedeutende Rolle, um die Aufmerksamkeit potenzieller Nutzer auf den ATM zu lenken. Ziel war es, den Lightning-ATM optisch ansprechend zu gestalten und dabei Elemente wie LED-Lichteffekte zu integrieren, die visuelle Rückmeldungen für den Benutzer liefern und gleichzeitig das Design des Automaten modern und innovativ wirken lassen. Lichttechnik kann dabei als Mittel zur Interaktion und Benutzernavigation fungieren, indem sie den aktuellen Status der Transaktion oder wichtige Hinweise visuell signalisiert. Dies trägt zu einer besseren Benutzererfahrung bei und unterstützt die intuitive Bedienung des Geräts.

Zusätzlich zu den visuellen Effekten sind akustische Rückmeldungen durch Lautsprecher ein weiterer wichtiger Aspekt des Designs. Durch gezielte Geräusche oder Sprachhinweise können Benutzer über den aktuellen Status des Vorgangs informiert werden, ohne dass sie die Anzeige kontinuierlich beobachten müssen. Akustisches Feedback unterstützt den

barrierefreien Zugang zum Gerät und verbessert die Gesamterfahrung der Benutzer. In einer lauten oder ablenkenden Umgebung tragen solche Rückmeldungen zur Sicherheit und Benutzerfreundlichkeit bei.

4.3.1 Designprozess des ATM-Gehäuses

Das Design des Gehäuses für den Lightning-ATM orientiert sich maßgeblich an bestehenden Projekten, insbesondere an der kleinen Version des ATM-Gehäuses von *21isenough*[34]. Dabei wurden wesentliche Designelemente übernommen und an die spezifischen Anforderungen des neuen ATM angepasst. Die äußere Gestaltung wurde sorgfältig geplant, um sowohl ästhetischen Ansprüchen gerecht zu werden als auch eine hohe Funktionalität zu bieten. Im folgenden Abschnitt werden die Materialwahl, das Design der Kanten, die Abmessungen sowie die Integration der Hardwarekomponenten beschrieben.

Materialwahl und Oberflächenbeschaffenheit

Das Gehäuse besteht aus MDF (Mitteldichte Faserplatte), einem vielseitigen und leicht zu verarbeitenden Holzwerkstoff, der im Vergleich zu Massivholz kostengünstiger ist. MDF zeichnet sich durch eine glatte und homogene Oberfläche aus, die sich gut für den Einsatz in Prototypen und Kleinserien eignet. Es bietet zudem eine hohe Maßhaltigkeit, was besonders wichtig ist, da die Komponenten des ATMs exakt eingebaut werden müssen. Trotz der vielen Vorteile ist MDF anfällig gegenüber Wasser und Feuchtigkeit, was bei der Planung berücksichtigt wurde.

Um das Gehäuse optisch ansprechender zu gestalten, wurde die Außenfläche mit einem dünnen Furnier aus Holz versehen. Das Furnier verleiht dem Gehäuse die Optik von echtem Holz und überdeckt die Schnittkanten des MDF-Materials, die sonst sichtbar wären. Diese furnierte Oberfläche trägt dazu bei, dem ATM eine hochwertigere Haptik zu verleihen, ohne dass die Kosten für teures Massivholz anfallen. Zusätzlich wurden die Außenkanten mit einer Fräse abgerundet, um das Gehäuse sowohl ästhetisch als auch ergonomisch aufzuwerten.

Abgerundete Kanten für modernes Design

Eine markante Designeigenschaft, die aus dem *21isenough*-Projekt [34] übernommen wurde, sind die abgerundeten Kanten des Gehäuses. Diese Konstruktionsentscheidung ist nicht nur ästhetischer Natur, sondern hat auch funktionale Vorteile. Abgerundete Kanten verleihen dem Gehäuse ein modernes und ansprechendes Erscheinungsbild, was insbesondere bei einem Gerät wie einem ATM, der öffentlich zugänglich ist, von Vorteil ist. Ein ATM, der gut gestaltet ist, zieht eher die Aufmerksamkeit auf sich und kann dadurch die Interaktion mit potenziellen Benutzern fördern.

Zudem bieten abgerundete Kanten einen zusätzlichen Schutz vor physischen Schäden, die durch scharfe Kanten entstehen könnten. Dies kann sowohl für den Benutzer als auch für das Gerät selbst von Vorteil sein, da es die Wahrscheinlichkeit von Verletzungen oder Beschädigungen reduziert. Die abgerundeten Ecken wurden durch den Einsatz einer Fräse erzeugt, was eine präzise und gleichmäßige Ausführung ermöglicht.

Maße und Abmessungen des Gehäuses

Die exakten Abmessungen des Gehäuses wurden so gewählt, dass sie sowohl eine optimale Unterbringung aller notwendigen Hardwarekomponenten als auch eine kompakte Bauweise ermöglichen. Das Gehäuse hat eine Breite von 30 cm, eine Höhe von 30 cm und eine Tiefe von 21 cm. Diese Maße wurden nach sorgfältiger Analyse der Platzanforderungen für die internen Komponenten festgelegt, darunter der Raspberry Pi, die Solenoids, das PCB, der Münzannahme-Mechanismus sowie das Display.

Durch die relativ kompakte Bauweise bleibt der ATM leicht transportierbar, was für Demonstrationen oder mobile Einsätze vorteilhaft ist. Gleichzeitig bieten die Innenmaße genügend Raum, um alle Komponenten ordnungsgemäß und sicher zu installieren.

Integration der Komponenten und vorbereitete Öffnungen

Das Gehäuse wurde im CAD-Modell (siehe Abbildung 4.6 und Abbildung 4.7) so gestaltet, dass es bereits die notwendigen Aussparungen und Öffnungen für die Hardwarekomponenten enthält. Diese wurden präzise in das Design integriert, um den Einbau der Komponenten zu erleichtern und deren Position im Gehäuse vorzugeben. So wurden beispielsweise Öffnungen für das Display, den Münzeinwurf sowie den Bereich für die Ausgabe von Münzen und den Zugang zu den Steuerungselementen vorgesehen.

Die vorbereiteten Öffnungen tragen wesentlich zur Benutzerfreundlichkeit des Baus bei, da sie bereits während der Fertigung des Gehäuses den exakten Platz für die Komponenten festlegen. Dies reduziert die Notwendigkeit von nachträglichen Anpassungen und stellt sicher, dass die Komponenten passgenau in das Gehäuse eingebaut werden können.

Abbildung 4.6: Vorderansicht des ATM-Gehäuses mit vorbereiteten Öffnungen für Display und Münzeinwurf

Abbildung 4.7: Rückansicht des ATM-Gehäuses mit Zugang für Wartungsarbeiten

Bauweise und Stabilität des Gehäuses

Das MDF-Material wurde basierend auf den im CAD-Programm erstellten Prototypenmodellen passgenau zugesägt und anschließend mittels Metallwinkeln zu einem stabilen Gehäuse verschraubt. Die Materialstärke beträgt 16mm, was die Verwendung von Schrauben der Größe 3mmx16mm erfordert, um sicherzustellen, dass die Schrauben nicht von außen sichtbar sind. Durch die zusätzliche Profilhöhe der verwendeten Winkel kann die maximale Tiefe und Stabilität des Gehäuses gewährleistet werden. Diese Konstruktion bietet eine hohe Belastungsfähigkeit, sodass sie selbst das Körpergewicht eines durchschnittlichen Erwachsenen problemlos tragen kann. Des Weiteren gewährleistet die solide Bauweise eine hohe Vandalismussicherheit, da das Gehäuse nur mit erheblichem Kraftaufwand oder professionellem Werkzeug beschädigt werden kann.

Ein weiterer Vorteil dieser Konstruktion liegt in der Modularität und der Möglichkeit, zukünftige Erweiterungen des Prototyps zu berücksichtigen. So besteht die Option, im Innenbereich Metallplatten anzubringen, ohne die Integrität des Gehäuses zu beeinträchtigen. Die Konstruktion erlaubt den einfachen Zugang zum Innenraum des ATMs, wodurch Wartungen oder Upgrades effizient durchgeführt werden können.

Wartungsfreundlichkeit

Um den Zugang für Wartungsarbeiten zu erleichtern, wird auf der Rückseite eine innenliegende Tür integriert. Diese reduziert nicht nur die Angriffsfläche für potenzielle Einbruchswerkzeuge wie Brecheisen, sondern ermöglicht auch ein einheitliches, ansprechendes Design. Die platzsparende Anbringung der Tür unterstützt die Sicherheitsanforderungen des Systems, während gleichzeitig ein einfacher Zugang zum Innenleben gewährleistet wird.

4.3.2 Hardwaretests und Prototypischer Aufbau

Solenoids, die im Rahmen dieser Tests verwendet wurden, arbeiten typischerweise mit einer Spannung von 12V. Aufgrund dieser höheren Spannung ist es erforderlich, NPN-Transistoren (2N2222) zu verwenden, die den Schaltstrom vom Raspberry Pi an den Solenoid weiterleiten. Zudem ist der Einsatz einer Diode (1N4007) notwendig, um den Rückstrom zu blockieren, der durch die Bewegung des Metallkerns im Solenoid beim Abschalten entsteht. Ein Widerstand von 1kΩ dient dazu, den Basisstrom des Transistors zu begrenzen, um Überlastungen zu vermeiden. Ein 5V-auf-12V-Konverter wird genutzt, um den Solenoid mit der erforderlichen Spannung zu versorgen, da der Raspberry Pi standardmäßig nur 5V liefert.

Test 1: 5V Solenoid

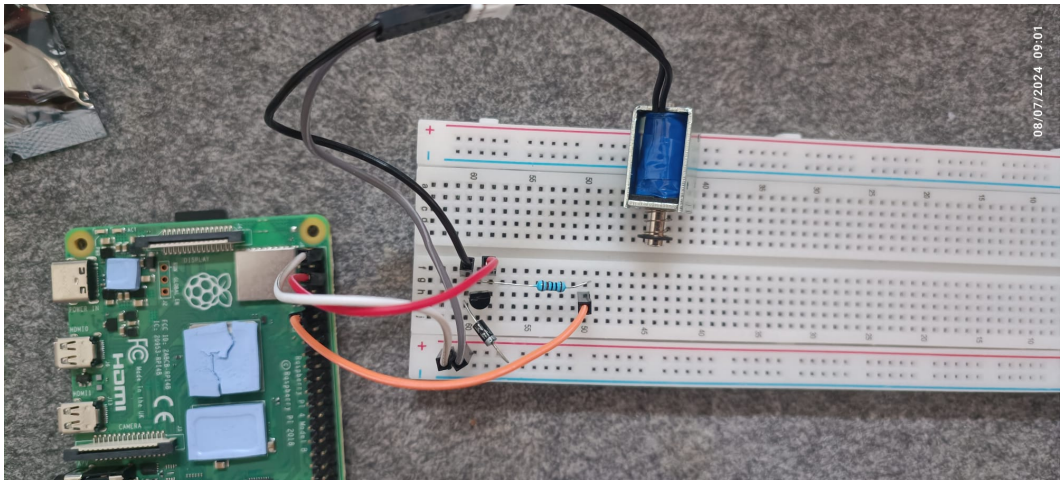


Abbildung 4.8: Schaltplan und Aufbau des 5V Solenoids auf dem Breadboard

Im ersten Test wird ein 5V Solenoid mit den folgenden Komponenten auf einem Breadboard aufgebaut: 5V Solenoid, Jumper-Kabel, Raspberry Pi 4B, NPN-Transistor (2N2222), Diode (1N4007) und ein 1kΩ Widerstand. Der Schaltplan ist in 4.8 dargestellt. Der Test zeigt, dass der 5V Solenoid zwar funktionierte, jedoch zu schwach ist, um die gewünschten Ergebnisse zu erzielen, da der Hubweg des Solenoids zu kurz war.

Test 2: 12V Solenoid

Im zweiten Test wird ein 12V Solenoid verwendet. Der Aufbau erfolgt mit den gleichen Komponenten wie im ersten Test, jedoch mit zusätzlicher Verwendung von Wago-Klemmen, um die dickeren Kabel des Solenoids zu verbinden. Der Raspberry Pi 4B, NPN-Transistor (2N2222), Diode (1N4007) und der 1kΩ Widerstand wurden ebenfalls verwendet. Der Schaltplan ist in 4.9 zu sehen. In diesem Test versagte der Transistor aufgrund der hohen Stromstärke, die der 12V Solenoid benötigte. Es wurde festgestellt, dass die Last direkt über den Transistor zu groß war, was zum Ausfall des Transistors führte (siehe 4.10).

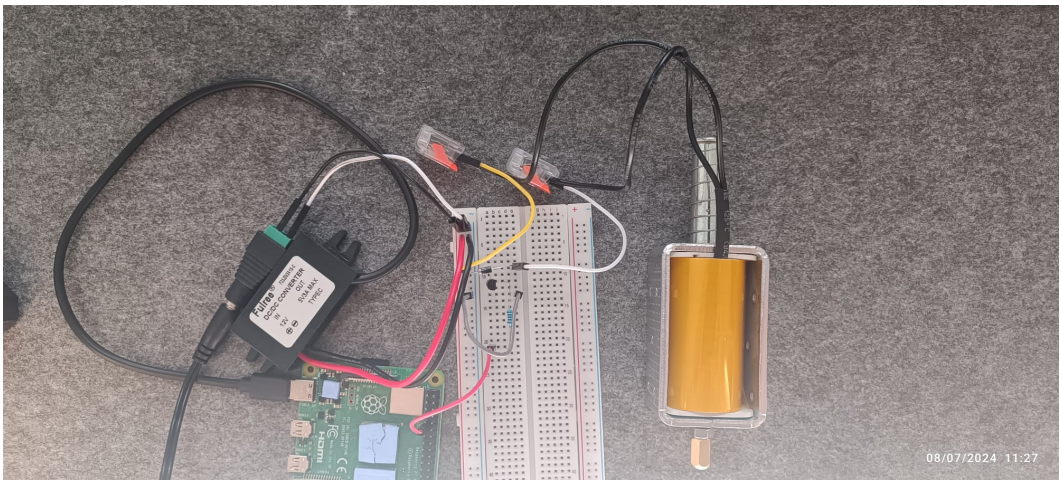


Abbildung 4.9: Schaltplan und Aufbau des 12V Solenoids, der den Transistor überlastet

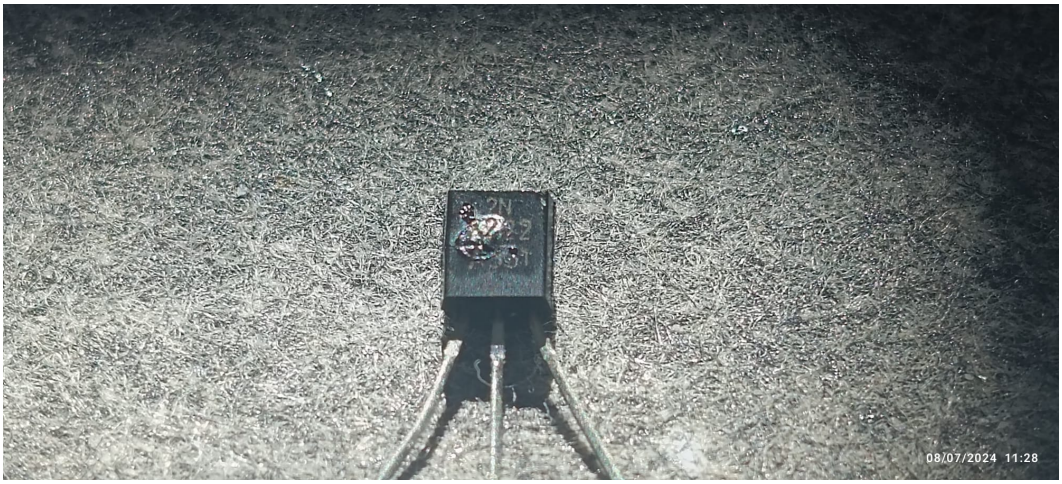


Abbildung 4.10: NPN Transistor mit Hitzeschäden

Test 3: 12V Solenoid mit Relay

Der dritte Test beinhaltet die Verwendung eines 12V Solenoids in Kombination mit einem 12V Relay, um die Last des Solenoids nicht direkt über den Transistor laufen zu lassen. Die Komponenten umfassen: 12V Solenoid, Jumper-Kabel, Raspberry Pi 4B, NPN-Transistor (2N2222), Diode (1N4007), 1kΩ Widerstand, 12V Relay und Wago-Klemmen. Der Schaltplan ist in 4.11 dargestellt. Obwohl der Schaltkreis korrekt funktioniert, zeigt sich, dass das Netzteil nicht ausreichend dimensioniert ist. Nach dem Einschalten des Solenoids über das Relay schaltet sich das Netzteil aufgrund von Lastspitzen sofort ab. Dies führt zu der Schlussfolgerung, dass ein leistungsstärkeres Netzteil erforderlich ist, um die Lastanforderungen von zwei 12V Solenoids zu bewältigen.

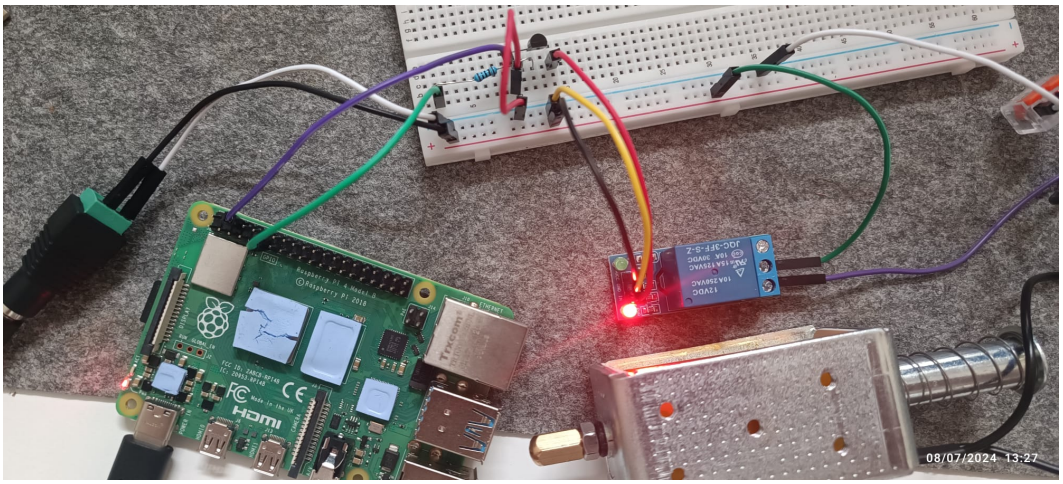


Abbildung 4.11: Schaltplan und Aufbau des 12V Solenoids mit Relay

Test 4: 12V Solenoid mit leistungsstarkem Netzteil

Im vierten Test wird ein 12V Solenoid mit einem 12V 20A Netzteil verwendet, um die Lastanforderungen des Systems zu erfüllen. Der Aufbau erfolgt mit den gleichen Komponenten wie im vorherigen Test: 12V Solenoid, Jumper-Kabel, Raspberry Pi 4B, NPN-Transistor (2N2222), Diode (1N4007), 1kΩ Widerstand, 12V Relay und Wago-Klemmen. Der Schaltplan ist in 4.12 zu sehen. Mit diesem leistungsstärkeren Netzteil funktioniert das System zuverlässig und kann zwei Solenoids gleichzeitig betreiben. Die Lastverteilung und der Stromfluss wird durch das Relay optimal gesteuert, was eine stabile Funktionsweise des gesamten Systems ermöglicht. Der visualisierte Schaltkreis ist Abbildung 4.13 zu entnehmen.

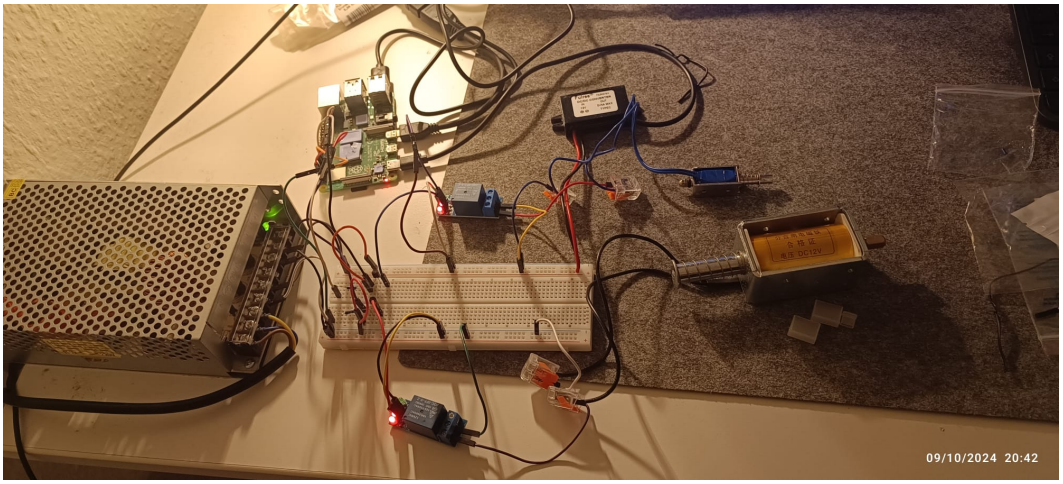


Abbildung 4.12: Schaltplan und Aufbau des 12V Solenoids mit einem 12V 20A Netzteil

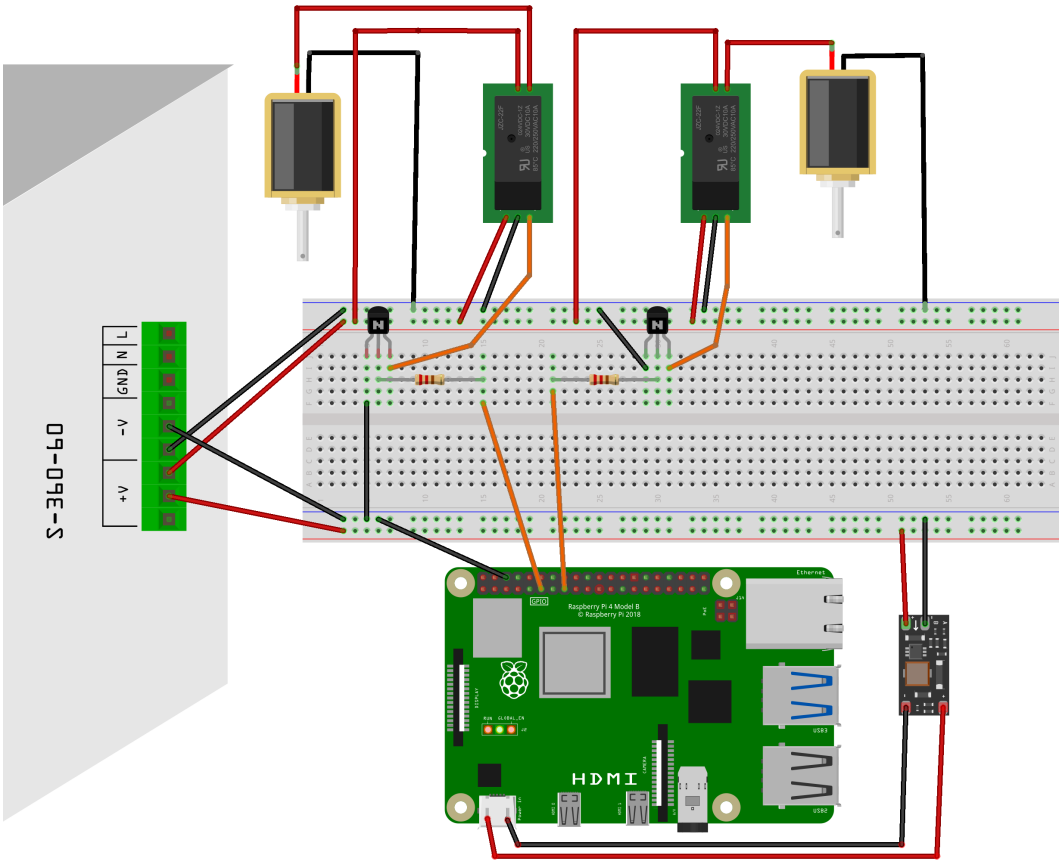


Abbildung 4.13: Visualisierter Schaltkreis von Test 4 aus 4.3.2 für die Solenoid-Steuerung auf 12VDC

4.3.3 Installation sowie Zusammenführung der Hardware

Nachdem die Grundstruktur des Gehäuses ohne die feste Front und die hintere Tür fertiggestellt wird, muss die Elektronik vervollständigt werden. Im Fokus steht hierbei das Verbinden der Elektrik mithilfe eines Lötkolbens sowie die Implementierung der zuvor entworfenen Leiterplatte (PCB), um ein homogenes und strukturiertes Innenleben des Automaten zu gewährleisten.

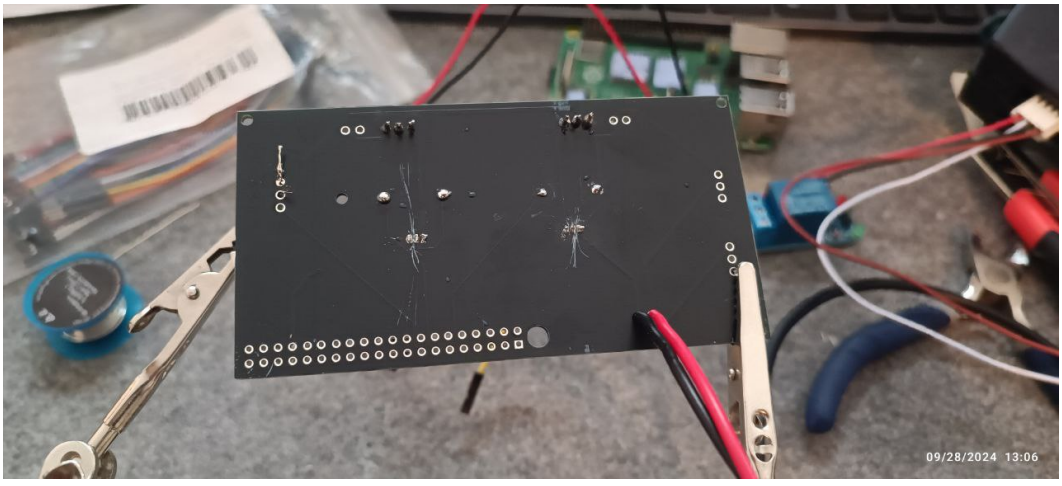


Abbildung 4.14: Einspannen des PCB in die Lötstation

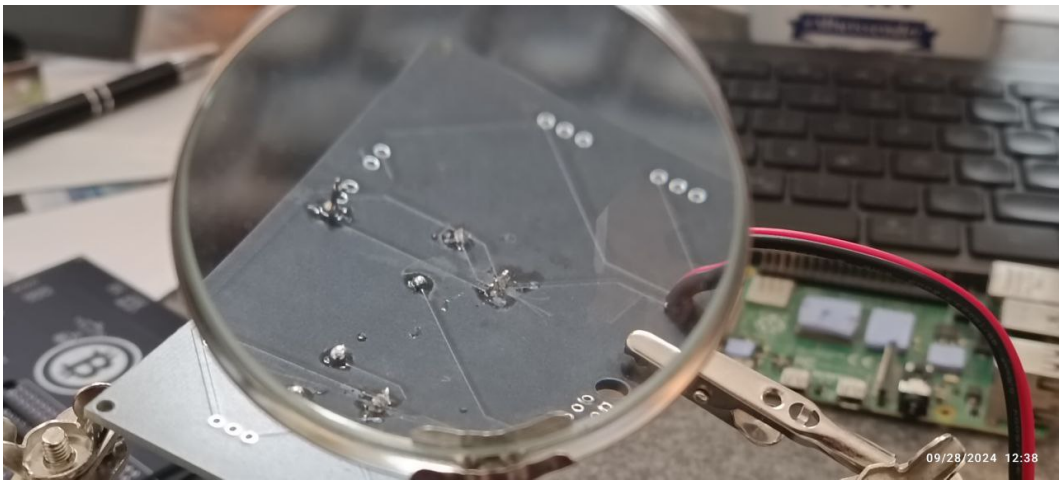


Abbildung 4.15: Übersicht über den Lötvorgang durch eine Lupe

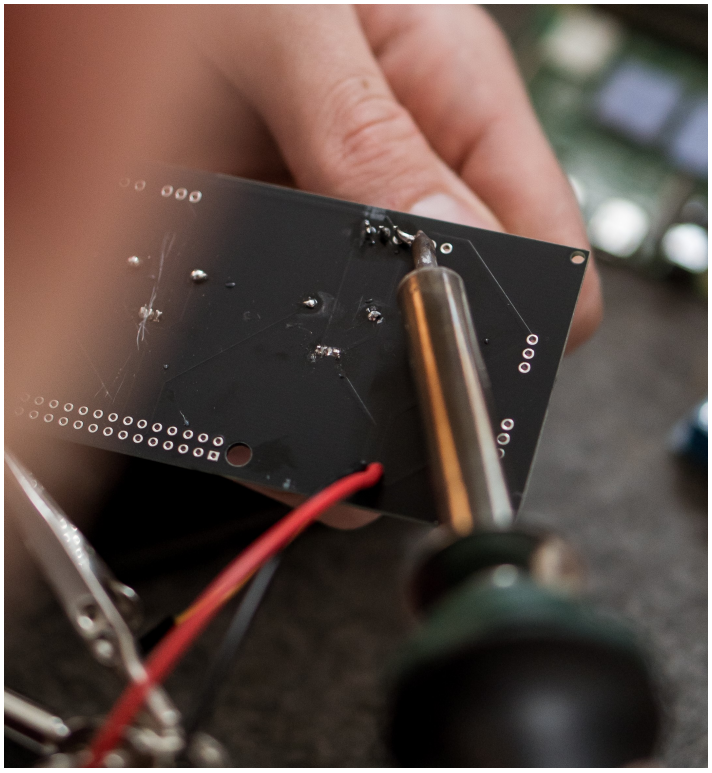


Abbildung 4.16: Lötvorgang

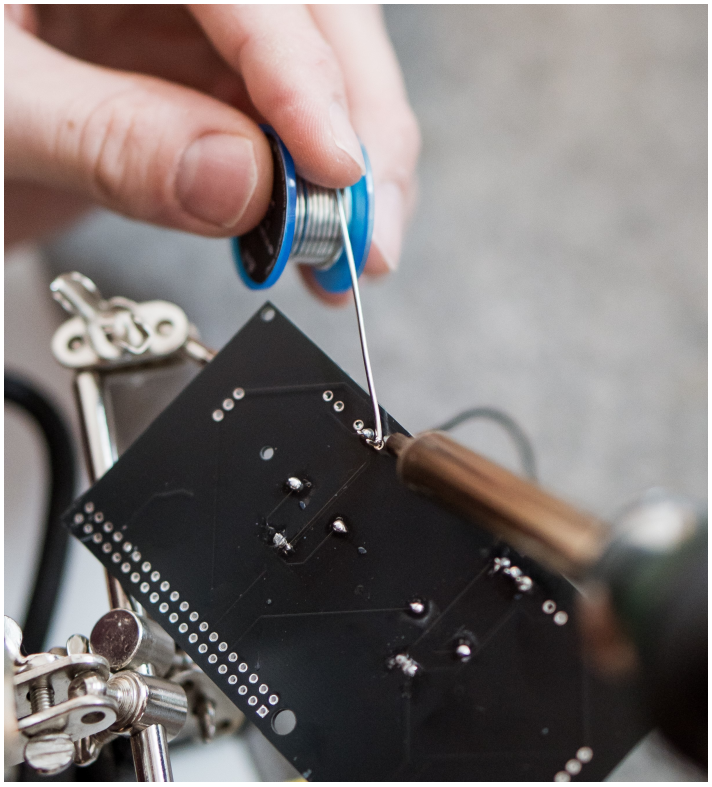


Abbildung 4.17: Weiterer Lötvorgang

Da die Lötstellen aufgrund der engen Platzverhältnisse auf der Leiterplatte schwierig sauber zu löten sind (siehe Abbildungen 4.14, 4.15, 4.16 und 4.17), führt der erste Lötversuch zu einem Kurzschluss, der den verbauten Raspberry Pi beschädigte. Beim zweiten Lötvorgang aller Komponenten wird deshalb mehr Zeit investiert, um sämtliche Lötstellen sorgfältig und fehlerfrei an der Leiterplatte anzubringen. Zur Sicherstellung der Qualität werden alle relevanten Schaltungen mit einem Multimeter getestet, um mögliche Durchgänge und Kurzschlüsse vorab zu identifizieren und zu vermeiden. Ein besonderes Augenmerk liegt dabei auf der Positionierung der einzelnen Hardwarekomponenten (siehe Abbildung 4.18) innerhalb des Gehäuses sowie dem Bau der Münzflussmechanik (siehe Abbildung 4.19).

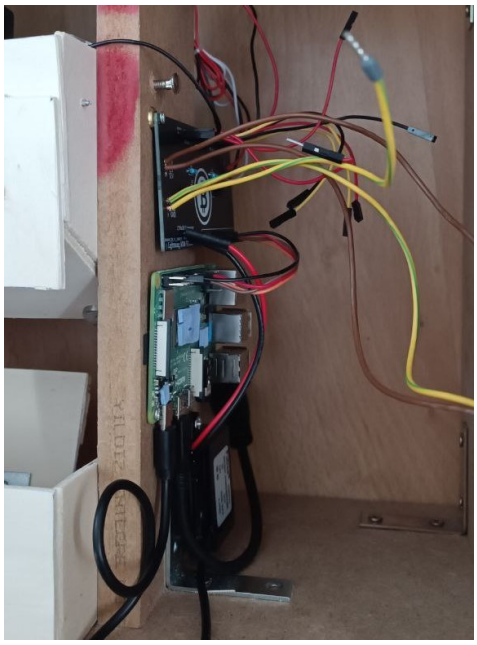


Abbildung 4.18: Platzierung der Komponenten im inneren des ATMs

Abbildung 4.19: Auffanggehäuse mit Widerstandsfeder

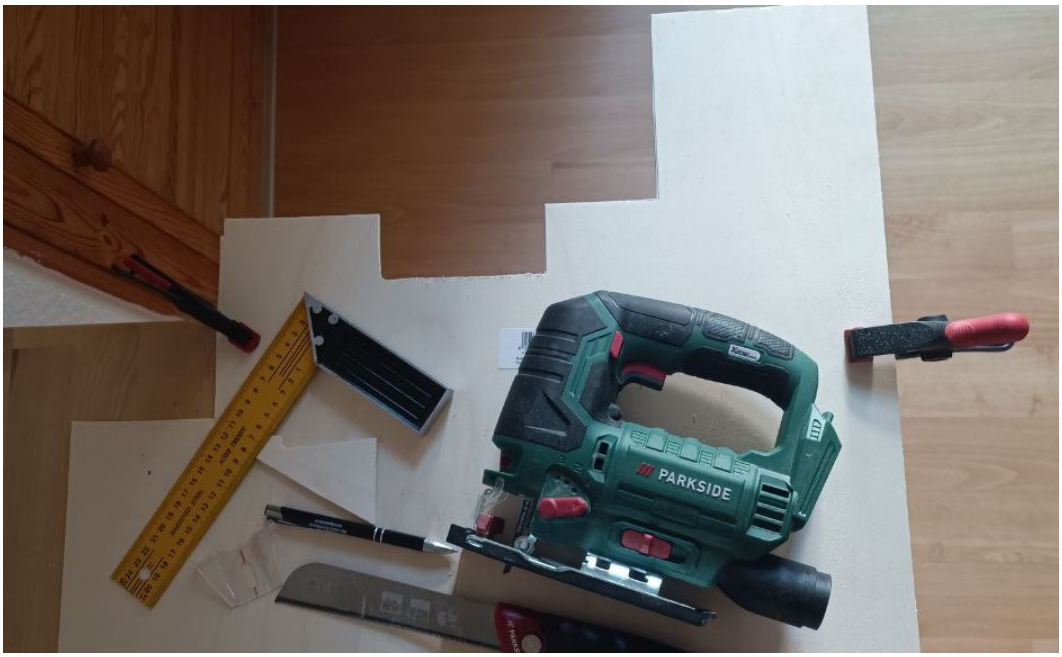


Abbildung 4.20: Zuschnitt der relevanten Holzstücke

Für die Konstruktion der Münzflussmechanik wird dünnes Leimholz verwendet, welches sich durch seine strukturelle Stabilität und geringe Materialstärke auszeichnet. Die Holzelemente werden entsprechend zugesägt (siehe Abbildung 4.20) und mit verschiedenen Metallwinkeln, 10mm Schrauben, Federn sowie Scharnieren zu einem Auffangbehälter für die Münzen zusammengesetzt. Der Behälter wird durch den Widerstand der Feder stets geschlossen gehalten, wobei ein Solenoid das Öffnen übernimmt(siehe Abbildung 4.19). Unterhalb dieses Behälters werden mithilfe der gleichen Komponenten zwei Auffangbehälter konstruiert, die durch eine mittels eines zweiten Solenoids gesteuerte Klappe voneinander getrennt sind. Diese Klappe steuert, ob die Münzen in den „Tresor“ fallen oder wieder an den Kunden ausgegeben werden. Abschließend wird die Front des Gehäuses maßgerecht zugesägt und das Display eingebaut. Zudem werden Aussparungen für die Scharniere und das Schloss an der hinteren Tür angebracht, um den gesamten Aufbau zu vervollständigen. Der Zusammenbau sowie der fertige ATM wird detailreicher in den Abbildungen 4.21, 4.22, 4.23 und 4.24 visualisiert.

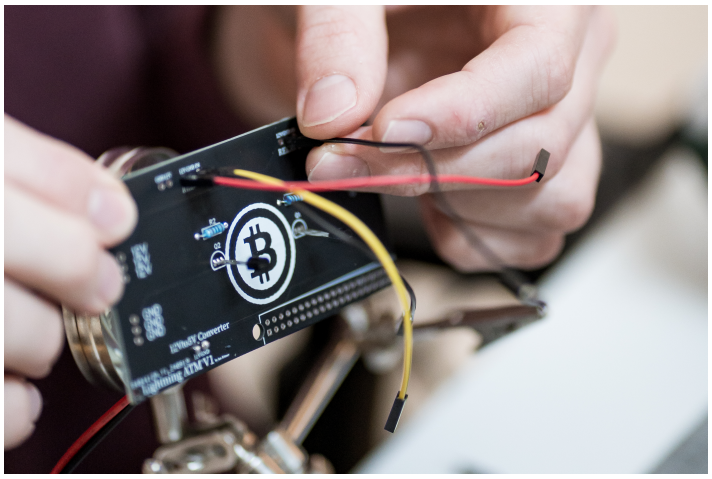


Abbildung 4.21: Löten der Relay-Anschlüsse am PCB

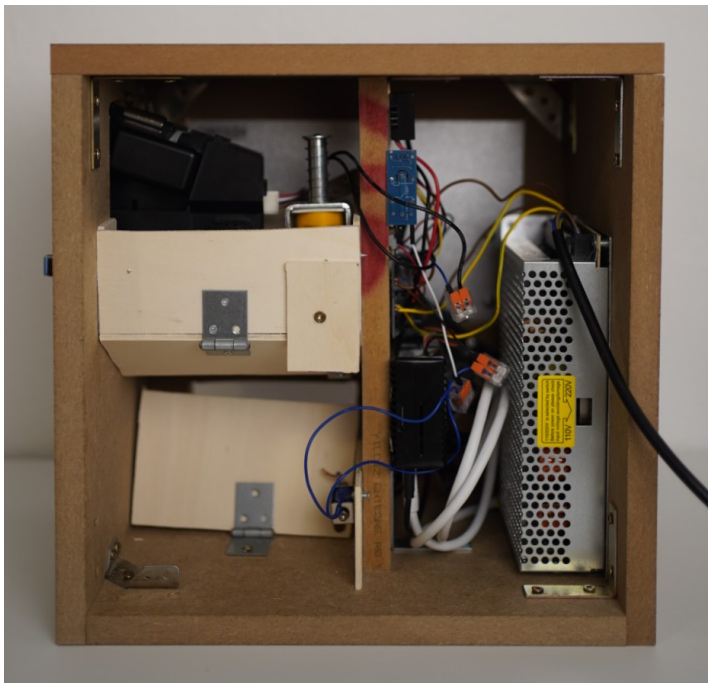


Abbildung 4.22: Ansicht des ATM von hinten

Abbildung 4.23: Verkabelung des Netzteils

Abbildung 4.24: Verbautes PCB hinter Kabeln

4.4 Software

Für das vorliegende Projekt wird eine speziell angepasste Software entwickelt, die auf die Integration und Abstimmung der verwendeten Hardwarekomponenten ausgerichtet ist. Das primäre Ziel der Software besteht darin, einen reibungslosen Ablauf der verschiedenen Prozesse zu gewährleisten, wobei sowohl eine benutzerfreundliche und visuell ansprechende Oberfläche als auch die Erfüllung aller sicherheitsrelevanten Anforderungen im Fokus stehen. Die Software ist zentraler Bestandteil des Lightning-ATMs und sorgt für die Koordination der Hardware, insbesondere der Solenoids, sowie der Transaktionsvorgänge.

4.4.1 Anforderungen

Im Rahmen der folgenden Abschnitte werden die grundlegenden Anforderungen an die Software erläutert. Darüber hinaus wird eine detaillierte Darstellung der Verbindung zwischen Frontend und Backend gegeben, um die Funktionsweise der Benutzeroberfläche sowie der zugrunde liegenden technischen Prozesse verständlich zu machen. Die Software des Lightning-ATMs muss eine zuverlässige Erkennung der eingezahlten Münzen sicherstellen, um darauf basierend eine LNURLw zu generieren. Diese wird dem Benutzer als QR-Code angezeigt, den er mit einer Lightning-Wallet scannen und abheben kann. Nach erfolgreicher Zahlung wird das eingezahlte Geld im ATM gesichert und die Transaktion als abgeschlossen verbucht. Sollte die Transaktion jedoch abgebrochen oder nicht erfolgreich durchgeführt werden, deaktiviert die Software die zuvor erstellte LNURLw und gibt das eingezahlte Geld mithilfe eines Solenoids an den Benutzer zurück. Dieser Prozess garantiert die Sicherheit und Integrität der Zahlungsvorgänge und stellt sicher, dass das Geld bei Fehlfunktionen oder abgebrochenen Transaktionen unverzüglich zurückerstattet wird.

4.4.2 Backend

Solenoid-Steuerung

Im Rahmen des Projekts werden zwei Python-Skripte zur Steuerung der Solenoids implementiert: `activate_solenoid_17.py` und `activate_solenoids.py`. Diese Skripte sind entscheidend für die Münzflusssteuerung, da sie das Einziehen und Ausgeben der Münzen mechanisch steuern. Beide Skripte verwenden die GPIO-Pins des Raspberry Pi, um die Solenoids für eine kurze Zeit zu aktivieren.

Solenoid-Steuerung mit `activate_solenoid_17.py`

Die Datei `activate_solenoid_17.py` steuert einen einzelnen Solenoid über den GPIO-Pin 17 und aktiviert ihn für einen Zeitraum von 2 Sekunden. Diese Funktion ist notwendig, um eine präzise Steuerung des Münzflusses zu gewährleisten. Die Dauer von 2 Sekunden wurde empirisch bestimmt, um eine ausreichende Bewegung des Solenoids zu garantieren, ohne dass unnötig viel Energie verbraucht wird.

Der relevante Code ist wie folgt dargestellt:

```
import RPi.GPIO as GPIO
import time

//GPIO-Nummerierung
GPIO.setmode(GPIO.BCM)

//GPIO-Pin definieren
SOLENOID_PIN = 17

//Pin 17 als Ausgang
GPIO.setup(SOLENOID_PIN, GPIO.OUT)

try:
    //Solenoid fuer 2 Sekunden aktivieren
    GPIO.output(SOLENOID_PIN, GPIO.HIGH)
    time.sleep(2)
    GPIO.output(SOLENOID_PIN, GPIO.LOW)
finally:
    GPIO.cleanup()
```

Analyse der Funktionalität

Der oben dargestellte Code setzt die GPIO-Nummerierung auf das `BCM`-Schema, wodurch die GPIO-Pins über ihre physikalischen Nummern gesteuert werden. Das BCM-Schema (Broadcom Chip Model) ist eine Pin-Nummerierungsmethode für die GPIO-Pins des Raspberry Pi, die auf den internen Broadcom-Prozessor-Nummern basiert. Im Gegensatz zur physischen Pin-Nummerierung entspricht das BCM-Schema den spezifischen Funktionen der Pins auf dem Raspberry Pi 4B und wird eben auch in diesem Konzept für `RPi.GPIO` verwendet. Der Pin 17 wird als Ausgang konfiguriert (`GPIO.setup(SOLENOID_PIN, GPIO.OUT)`), da der Solenoid durch diesen Pin aktiviert wird.

Die eigentliche Aktivierung des Solenoids erfolgt durch das Setzen des Pins auf `HIGH(GPIO.output(SOLENOID_PIN, GPIO.HIGH))`, was eine elektrische Verbindung herstellt und den Solenoid für 2 Sekunden aktiviert. Die Verwendung von `time.sleep(2)` garantiert, dass der Solenoid für genau 2 Sekunden aktiv bleibt, bevor der Pin wieder auf `LOW` gesetzt wird (`GPIO.output(SOLENOID_PIN, GPIO.LOW)`). Nach der Steuerung wird die GPIO-Konfiguration durch `GPIO.cleanup()` zurückgesetzt, um sicherzustellen, dass der Pin für andere Operationen freigegeben wird.

Steuerung beider Solenoids mit `activate_solenoids.py`

Die Datei `activate_solenoids.py` steuert zwei Solenoids gleichzeitig. Dabei werden GPIO-Pin 17 und GPIO-Pin 27 für zwei unterschiedliche Solenoids verwendet. Auch hier wird eine Aktivierung für 2 Sekunden durchgeführt, um sicherzustellen, dass beide Solenoids synchron arbeiten und somit eine einheitliche Steuerung des Münzflusses möglich ist.

Der Code zur Steuerung beider Solenoids lautet wie folgt:

```
import RPi.GPIO as GPIO
import time
```

```
//Setze die GPIO-Nummerierung
GPIO.setmode(GPIO.BCM)

//Definiere die GPIO-Pins fuer die Solenoids
SOLENOID_1_PIN = 17
SOLENOID_2_PIN = 27

//Setze die Pins als Ausgaenge
GPIO.setup(SOLENOID_1_PIN, GPIO.OUT)
GPIO.setup(SOLENOID_2_PIN, GPIO.OUT)

try:
    //Solenoids fuer 2 Sekunden aktivieren
    GPIO.output(SOLENOID_1_PIN, GPIO.HIGH)
    GPIO.output(SOLENOID_2_PIN, GPIO.HIGH)
    time.sleep(2)
    GPIO.output(SOLENOID_1_PIN, GPIO.LOW)
    GPIO.output(SOLENOID_2_PIN, GPIO.LOW)
finally:
    GPIO.cleanup()
```

Analyse der Funktionalität

In dieser Implementierung werden zwei Solenoids simultan gesteuert. Beide GPIO-Pins (17 und 27) werden als Ausgänge konfiguriert `GPIO.setup(SOLENOID_1_PIN, GPIO.OUT)` und `GPIO.setup(SOLENOID_2_PIN, GPIO.OUT)`. Die Steuerung erfolgt ähnlich wie bei der Einzel-Solenoid-Funktion, jedoch werden hier beide Pins gleichzeitig auf `HIGH` gesetzt (`GPIO.output(SOLENOID_1_PIN, GPIO.HIGH)` und `GPIO.output(SOLENOID_2_PIN, GPIO.HIGH)`), wodurch beide Solenoids für 2 Sekunden aktiviert werden. Nach Ablauf dieser Zeit werden beide Pins wieder auf `LOW` gesetzt, um die Solenoids zu deaktivieren.

Diese parallele Steuerung ist wichtig, um sicherzustellen, dass der Münzfluss präzise und synchron gesteuert wird, was für den gleichzeitigen Einsatz mehrerer Solenoids im ATM von zentraler Bedeutung ist.

Wichtigkeit der Solenoid-Steuerungen

Die implementierten Steuerungen für die Solenoids sind essentiell für die mechanische Steuerung des Münzflusses im Lightning-ATM. Die präzise zeitgesteuerte Aktivierung der Solenoids ermöglicht eine kontrollierte und sichere Ausgabe bzw. Speicherung der Münzen. Beide Steuerungen stellen sicher, dass die Last auf den GPIO-Pins des Raspberry Pi korrekt gehandhabt wird und keine unerwarteten elektrischen Rückkopplungen auftreten, die die Elektronik beschädigen könnten.

Backend.py und Münzannahme

Die Datei `backend.py` ist für die Konfiguration und Steuerung des Coin-Acceptors verantwortlich. Diese Komponente spielt eine zentrale Rolle bei der Erkennung und Verarbeitung von Münzeinzahlungen im Lightning-ATM. Der Coin-Acceptor ist über den GPIO-Pin 22 mit dem

Raspberry Pi verbunden. Warum dieser GPIO Pin verwendet wurde liegt daran, dass nach dem BCM-Schema dieser elektronische Signale annehmen aber auch ausgeben kann, welche für die Erkennung der Hardware benötigt werden. Es gibt mehrere Pins auf dem Raspberry Pi 4B mit diesen Eigenschaften und da die GPIO Pins 22 und 27 die selben Eigenschaften haben und zusätzlich nebeneinander liegen wurden diese als GPIO-Pins ausgewählt. Der folgende Abschnitt erläutert die wichtigsten Funktionen dieser Datei und erklärt die zugrunde liegende Logik.

Initialisierung und GPIO-Konfiguration

Die erste wichtige Aufgabe in `backend.py` ist die Konfiguration des Coin-Acceptors über den GPIO-Pin. Der Pin wird als Eingang konfiguriert, da der Raspberry Pi Signale vom Coin-Acceptor empfangen muss. Der entsprechende Code ist wie folgt:

```
import RPi.GPIO as GPIO
import time
import sys
import json

//Setze die GPIO-Nummerierung
GPIO.setmode(GPIO.BCM)

//Definiere den GPIO Pin fuer den Coin Acceptor
COIN_ACCEPTOR_PIN = 22

//Setze den Pin als Eingang
GPIO.setup(COIN_ACCEPTOR_PIN, GPIO.IN, pull_up_down=GPIO.PUD_UP)
```

Analyse der GPIO-Konfiguration

Die GPIO-Nummerierung wird auf das `BCM`-Schema gesetzt, das es ermöglicht, den Coin-Acceptor mit seinem physikalischen Pin 22 zu verknüpfen. Der Pin wird als Eingang konfiguriert (`GPIO.setup(COIN_ACCEPTOR_PIN, GPIO.IN)`), was notwendig ist, da der Coin-Acceptor Pulse sendet, die vom Raspberry Pi gelesen werden müssen. Die `pull_up_down`-Konfiguration auf `GPIO.PUD_UP` sorgt dafür, dass der Pin standardmäßig auf `HIGH` gezogen wird, um Fehlsignale zu vermeiden, bis ein tatsächlicher Impuls erkannt wird.

Münzerkennung und Pulszählung

Die Münzerkennung basiert auf dem Empfang von elektrischen Pulsen, die vom Coin-Acceptor gesendet werden, wenn eine Münze eingeworfen wird. Diese Pulse werden gezählt und in einen spezifischen Münzwert umgerechnet. Der folgende Code definiert die Zählung und Erkennung der Pulse:

```
PULSE_TIMEOUT = 1
DEBOUNCE_TIME = 0.1

PULSE_TO_VALUE = {
    2: 0.05,  //5 Cent
    3: 0.10,  //10 Cent
```

```
        4: 0.20,  //20 Cent
        5: 0.50,  //50 Cent
        6: 1.00,  //1 Euro
        7: 2.00   //2 Euro
    }

    total_value = 0  //Gesamtsumme der erkannten Muenzen

    def count_pulses():
        pulse_count = 0
        start_time = time.time()

        while (time.time() - start_time) < PULSE_TIMEOUT:
            if GPIO.input(COIN_ACCEPTOR_PIN) == GPIO.LOW:
                pulse_count += 1
                time.sleep(DEBOUNCE_TIME)

        return pulse_count
```

Analyse der Münzerkennung

Der Algorithmus zur Münzerkennung nutzt eine Kombination aus Pulszählung und Zeitmessung, um sicherzustellen, dass jede Münze korrekt erkannt und gezählt wird. Die Konstante `PULSE_TIMEOUT` gibt an, wie lange der Raspberry Pi auf Pulse wartet, bevor er die Zählung stoppt. Diese Timeout-Strategie verhindert das Erkennen von Fehlsignalen oder das Überzählen von Pulsen, die über einen längeren Zeitraum eintreffen könnten.

Die Funktion `count_pulses()` zählt die Anzahl der erkannten Pulse, indem sie den GPIO-Pin periodisch abfragt. Jeder erkannte Puls wird durch `GPIO.input(COIN_ACCEPTOR_PIN)== GPIO.LOW` erfasst, da ein niedriger Pegel (`LOW`) anzeigt, dass der Coin-Acceptor einen Puls gesendet hat. Nach jedem erkannten Puls wird eine kurze Pause (`DEBOUNCE_TIME`) eingelegt, um sicherzustellen, dass keine doppelten Pulse erfasst werden.

Zuordnung der Pulse zu Münzwerten

Nach dem Erfassen der Pulse wird der erkannte Wert der Münze durch eine Zuordnungstabelle in einen spezifischen Geldwert umgerechnet. Der folgende Codeabschnitt zeigt, wie die Pulse einem Münzwert zugeordnet werden:

```
    def detect_coin():
        global total_value
        try:
            while True:
                if GPIO.input(COIN_ACCEPTOR_PIN) == GPIO.LOW:
                    pulses = count_pulses()

                    //Muenzwert ermitteln
                    coin_value = PULSE_TO_VALUE.get(pulses, None)
                    if coin_value:
                        total_value += coin_value
                        //Sende den Wert an das Frontend als JSON
                        output = json.dumps({"value": total_value})
```

```
                        print(output)
                        sys.stdout.flush()

                time.sleep(1)
    finally:
        GPIO.cleanup()
```

Erklärung der Münzwertzuordnung

Die Funktion `detect_coin()` ist die zentrale Logik zur Erkennung und Verarbeitung von Münzen. Jedes Mal, wenn ein Münzeinwurf erkannt wird, wird die Anzahl der Pulse durch die Funktion `count_pulses()` ermittelt und mithilfe der Tabelle `PULSE_TO_VALUE` in einen Münzwert umgerechnet. Diese Tabelle ordnet der Anzahl der Pulse den jeweiligen Geldwert zu, z. B. 2 Pulse für 5 Cent oder 7 Pulse für 2 Euro. Die Tabelleneinträge müssen vorher am Coinacceptor vorkonfiguriert und die jeweiligen Münzen durch Größe, Gewicht und Form angelernt werden.

Wenn ein gültiger Münzwert erkannt wird, wird der Gesamtbetrag `total_value` aktualisiert. Der aktuelle Gesamtwert wird anschließend als JSON-Objekt an das Frontend gesendet, sodass der Benutzer den Betrag sehen kann. Der JSON-Output (`output = json.dumps("value" : total_value)` wird an das Frontend übermittelt, indem er über `sys.stdout.flush()` sofort ausgegeben wird.

Sicherstellung der Datenintegrität

Die Verwendung von JSON zur Datenübertragung stellt sicher, dass die Kommunikation zwischen dem Backend und dem Frontend in einem standardisierten Format erfolgt. Dies gewährleistet, dass alle erkannten Münzen korrekt übermittelt und im System registriert werden.

Zusammenfassend ermöglicht der Code in `backend.py` eine präzise Erkennung und Verarbeitung der eingeworfenen Münzen. Die Pulszählung und Zuordnung zu den entsprechenden Münzwerten stellt sicher, dass die Münzannahme zuverlässig und sicher erfolgt. Die Nutzung der GPIO-Pins des Raspberry Pi erlaubt eine direkte Interaktion mit dem Coin-Acceptor, und die JSON-basierte Kommunikation sorgt für eine effiziente Datenübermittlung an das Frontend.

Funktionalität der App.py

Die Datei `app.py` ist für die Verwaltung der Zahlungsprozesse des Lightning-ATMs verantwortlich. Sie generiert dynamisch eine LNURLw und überwacht den Zahlungsstatus, um den Münzfluss zu steuern. Diese Datei kommuniziert sowohl mit dem Backend, das für die Münzannahme zuständig ist, als auch mit dem Lightning-Netzwerk, um Zahlungen sicher abzuwickeln.

Erstellung einer LNURLw

Die Hauptaufgabe der Funktion `main()` besteht darin, eine LNURLw zu generieren, die der Benutzer über eine Lightning-Wallet scannen und abheben kann. Dieser Prozess beginnt mit der Erstellung eines Abhebungslinks, der die Grundlage für die QR-Code-Generierung bildet. Der relevante Code ist wie folgt dargestellt:

```
    def main():
        //Erstellen eines LNURLw-Links
        lnurl_data = backend.create_withdraw_link("Lightning-ATM Withdrawal", satoshis,
            uses=1, wait_time=5)

        if lnurl_data:
            //Generiere und sende den Pfad zum QR-Code
            qr_code_path = backend.generate_qr_code(lnurl_data['lnurl'])
            print(qr_code_path)  //Pfad zum QR-Code-Bild an Electron senden
            sys.stdout.flush()   //<Sicherstellen, dass die Nachricht sofort gesendet wird
```

Erklärung der LNURL-Erstellung

Die Funktion `create_withdraw_link()` erstellt eine LNURLw mit spezifischen Parametern wie dem Betrag und der Anzahl der Verwendungen. Hierbei wird der Abhebungsbetrag auf satoshis festgelegt, die nur einmal genutzt werden kann (`uses=1`) und eine Wartezeit von 5 Sekunden (`wait_time=5`) erfordert. Diese Funktion ist zentral für die Sicherheit der Transaktion, da sie sicherstellt, dass die LNURLw nur einmal verwendet werden kann und nach Ablauf der Wartezeit automatisch abläuft.

Sobald die LNURLw generiert wurde, wird sie als QR-Code dargestellt, den der Benutzer scannen kann, um die Zahlung zu initiieren. Der Pfad zum QR-Code wird an die Benutzeroberfläche gesendet (`print(qr_code_path)`) und über `sys.stdout.flush()` sofort ausgegeben, um Verzögerungen zu vermeiden.

Überwachung der Transaktion und Steuerung des Münzflusses

Sobald der QR-Code bereitgestellt wurde, beginnt die Überwachung der Zahlung. Dabei wird ein Countdown gestartet, der auf eine Bestätigung der Zahlung wartet. Diese Überwachung erfolgt in einer Schleife, die 45 Sekunden lang aktiv bleibt. Der relevante Code lautet wie folgt:

```
    claimed = False
    for i in range(45, 0, -1):
        //Ueberpruefe, ob die Abhebung erfolgreich beansprucht wurde
        if backend.check_withdrawal_claimed(lnurl_data['id']):
            print("claimed")  //Erfolgreiche Bestaetigung ausgeben
            sys.stdout.flush()
            backend.deposit_money()  //Geld im ATM sichern
            claimed = True
            break  //Timer-Schleife stoppen
```

Erklärung der Transaktionsüberwachung

Die Funktion `check_withdrawal_claimed()` überprüft, ob die erzeugte LNURLw erfolgreich beansprucht wurde. Diese Überprüfung erfolgt in einem Zeitraum von 45 Sekunden. Wird die Zahlung erfolgreich bestätigt, gibt die Anwendung eine entsprechende Erfolgsmeldung (`print("claimed")`) aus und das eingezahlte Geld wird durch den Aufruf von `backend.deposit_money()` gesichert. Diese Logik gewährleistet, dass das Geld nur dann im ATM gesichert wird, wenn die Zahlung tatsächlich eingegangen ist.

Sobald die Transaktion erfolgreich abgeschlossen wurde, wird die Schleife beendet und der Countdown gestoppt. Dies verhindert unnötige zusätzliche Überprüfungen und reduziert die Belastung der Backend-Funktionen.

Rückerstattung bei fehlgeschlagener Transaktion

Falls die Zahlung nicht innerhalb der vorgegebenen 45 Sekunden abgeschlossen wird, gilt die Transaktion als fehlgeschlagen. In diesem Fall wird das eingezahlte Geld zurückgegeben und die erzeugte LNURL-Invoice wird ungültig gemacht. Dies wird im folgenden Codeblock ausgeführt:

```
    if not claimed:
        print("timeout")  //Timeout-Meldung ausgeben
        sys.stdout.flush()
        backend.return_money()  //Geld zurueckgeben
```

Erklärung der Rückerstattungslogik

Falls die Transaktion nicht erfolgreich durchgeführt wird, gibt die Anwendung eine Zeitüberschreitungsmeldung (`print("timeout")`) aus und übermittelt sie an die Benutzeroberfläche. Die Funktion `backend.return_money()` sorgt dann dafür, dass das eingezahlte Geld über den Solenoid an den Benutzer zurückgegeben wird. Diese Rückerstattungslogik stellt sicher, dass kein Geld verloren geht und keine ungültigen LNURLws im System verbleiben.

Fazit der App.py

Die in der `app.py` implementierten Funktionen garantieren einen reibungslosen Ablauf des Zahlungsprozesses im Lightning-ATM. Die dynamische Erstellung von LNURLs, die Überwachung der Transaktion und die Handhabung von Rückerstattungen sorgen dafür, dass der Münzfluss präzise gesteuert wird. Die Integration zwischen dem Backend und dem Lightning-Netzwerk stellt sicher, dass der ATM zuverlässig funktioniert und sowohl Sicherheit als auch Benutzerfreundlichkeit gewährleistet werden.

4.4.3 Frontend

Struktur der Benutzeroberfläche: `index.html`

Die Datei `index.html` bildet die Grundlage der Benutzeroberfläche des Lightning-ATMs. Sie definiert die verschiedenen Bildschirme und die Steuerungselemente, die der Benutzer während des Transaktionsprozesses sieht und verwendet. Die Benutzeroberfläche ist darauf ausgelegt, den Münzeinwurf, die QR-Code-Erstellung sowie den Zahlungserfolg und -abbruch visuell ansprechend und benutzerfreundlich zu gestalten. Die Struktur dieser Datei lässt sich in vier Hauptbildschirme unterteilen: den Startbildschirm, den Münzeinwurf-Bildschirm, den QR-Code-Bildschirm und den Erfolg-Bildschirm. Außerdem läuft ein permanentes mp4-Video im Loop als Hintergrund.

```
<!DOCTYPE html>
<html lang="de">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Bitcoin Lightning-ATM</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>

    //-- Background Video -->
    <video id="background-video" autoplay loop muted>
        <source src="file:///home/atm/backgroundvideo.mp4" type="video/mp4">
    </video>

    //-- Start-Bildschirm -->
 <div id="start-screen" class="active">
        <h1>Deine Reise ins Bitcoin Universum!</h1>
        <button id="start-button">Start</button>
    </div>
    //-- Muenz-Bildschirm -->
    <div id="coin-screen" style="display: none;">
        <h2>Wie viel Satoshis willst du kaufen?</h2>
        <div id="value-display">Gesamtsumme: 0.00 Euro</div>
        <div id="satoshi-display">In Satoshis: 0 Sats</div>
        <button id="cancel-button">Abbrechen</button>
        <button id="proceed-button">Fortfahren</button>
    </div>

    //-- QR-Code-Bildschirm -->
    <div id="qr-screen" style="display: none;">
        <h2>Scanne den QR-Code mit deiner Ln-Wallet</h2>
        <img id="qr-code" src="" alt="LNURL QR-Code">
        <button id="back-button">Abbrechen</button> //-- Zurueck-Button fuer Abbrechen
            -->
    </div>

    //-- Success-Bildschrim -->

    <div id="success-screen" style="display: none;">
        <h2>Willkommen im Bitcoin Universum!<h2>
        <button id="restart-button">Ich will mehr!</button>
```

```
        <script src="renderer.js"></script>
    </body>
    </html>
```

Startbildschirm

Der `start-screen` ist der erste Bildschirm, den der Benutzer sieht. Er enthält einen Willkommensgruß sowie eine Schaltfläche, mit der der Transaktionsprozess gestartet wird. Der Button mit der ID `start-button` ist mit einer Funktion im `renderer.js` verbunden, die auf die Aktivierung des Münzeinwurf-Bildschirms (`coin-screen`) umschaltet. Sobald der Benutzer diesen Button klickt, wird die Münzerkennung durch das Backend initialisiert. Dies wird über den Event-Listener in der `renderer.js`-Datei gesendet, wie in folgendem Beispiel dargestellt:

```
startButton.addEventListener('click', () => {
    ipcRenderer.send('start-coin-counting');
});
```

Münzeinwurf-Bildschirm

Der `coin-screen` wird aktiviert, sobald der Benutzer den Transaktionsprozess startet. Hier wird dem Benutzer angezeigt, dass er Münzen einwerfen soll. Zusätzlich zeigt das System die Summe der eingeworfenen Münzen in Euro (`value-display`) sowie die Umrechnung in Satoshis (`satoshi-display`) an. Beide Anzeigen werden dynamisch aktualisiert, sobald der Wert der Münzen über das Backend gesendet wird. Die Funktion im Backend, die für die Münzerkennung zuständig ist, wurde in der Datei `backend.py` implementiert und nutzt die Funktion `detect_coin()`.

```
ipcRenderer.on('coin-value', (event, coinData) => {
    currentEuroValue = coinData.value;
    valueDisplay.textContent = `Gesamtsumme: ${coinData.value.toFixed(2)} Euro`;
});
```

Diese Funktion empfängt die von der `backend.py` gesendeten Daten, die den Wert der eingeworfenen Münzen als JSON-Objekt an das Frontend übertragen. Der Wert wird auf dem Bildschirm angezeigt und fortlaufend aktualisiert, während der Benutzer Münzen einwirft.

QR-Code-Bildschirm

Der `qr-screen` zeigt den QR-Code der LNURLw, die durch das Backend generiert wurde. Der QR-Code wird verwendet, um die Lightning-Zahlung abzuschließen. Sobald der Benutzer genügend Münzen eingeworfen hat und die `proceed-button` Funktion betätigt, wird der QR-Code über die folgende Logik angezeigt:

```
ipcRenderer.on('lnurl-qr', (event, { lnurl }) => {
    qrCode.src = `https://api.qrserver.com/v1/create-qr-code/?size=150x150&data=${
        encodeURIComponent(lnurl)}`;
    coinScreen.style.display = 'none';
    qrScreen.style.display = 'block';
});
```

Die LNURLw wird im Backend generiert und das Frontend empfängt die LNURLw, um sie als QR-Code anzuzeigen.Um den QR-Code zu generieren wird ein online QR-server aufgerufen, welcher den qrCode erstellt und als 150x150-Pixel Bild zurückgibt. Diese Funktion stellt sicher, dass der Benutzer die Transaktion über eine Lightning-Wallet abschließen kann.

Erfolg-Bildschirm

Der `success-screen` wird aktiviert, sobald die Lightning-Zahlung erfolgreich abgeschlossen wurde. Der Bildschirm zeigt eine Erfolgsnachricht an und ermöglicht es dem Benutzer, den ATM neu zu starten. Der Neustart des Systems wird durch den `restart-button` ausgelöst.

Verknüpfung von Frontend und Backend

Die Datei `index.html` enthält die Struktur für die verschiedenen Benutzerbildschirme, die während des Zahlungsprozesses verwendet werden. Jede Benutzerinteraktion, die über Buttons erfolgt, ist direkt mit den entsprechenden Backend-Funktionen verknüpft, die in `backend.py` implementiert sind. Die Funktionalitäten zur Münzerkennung, zur Erstellung der LNURLw und zur Steuerung der Solenoids werden alle durch das Backend bereitgestellt und über das Frontend gesteuert.

JavaScript-Logik und Interaktion: `main.js`

Die Datei `main.js` übernimmt die Verwaltung der Verbindungen zwischen dem Frontend und dem Backend. Sie spielt eine entscheidende Rolle bei der Steuerung der Hardwarekomponenten wie der Solenoids und dem Münzannahmesystem. Außerdem verwaltet sie die Kommunikation mit den Funktionen zur Erstellung der LNURLw und zur Überwachung der Zahlungstransaktionen.

Einbindung des `ipcMain`-Moduls

Das `ipcMain`-Modul von Electron ermöglicht die Kommunikation zwischen den Renderer-Prozessen (Frontend) und den Hauptprozessen (Backend). In diesem Fall wird das Modul genutzt, um Ereignisse vom Frontend zu empfangen und darauf basierend Backend-Prozesse auszuführen. Die grundlegende Struktur der Kommunikation mit dem Backend ist wie folgt:

```
const { ipcMain } = require('electron');
const { spawn } = require('child_process');
```

Die Funktion `ipcMain.on()` ermöglicht es, bestimmte Ereignisse vom Renderer-Prozess zu empfangen. Dies ist wichtig für den Austausch von Informationen zwischen der Benutzeroberfläche und den Hardwarekomponenten, wie der Steuerung der Solenoids und dem Rückgabesystem für Münzen.

Aktivierung der Solenoids

Eine der wesentlichen Aufgaben der `main.js`-Datei ist die Steuerung der Solenoids, die für die Münzflusssteuerung verantwortlich sind. In diesem Fall wird die Funktion zur Aktivierung der Solenoids ausgeführt, sobald das Frontend ein entsprechendes Signal sendet. Der folgende Code zeigt, wie die Solenoids über einen Python-Prozess aktiviert werden:

```
ipcMain.on('activate-solenoids', (event) => {
    console.log('Solenoids 17 und 27 aktiviert');
    const solenoidProcess = spawn('python3', ['../python_backend/activate_solenoids.py']);
    solenoidProcess.stderr.on('data', (data) => {
        console.error('Fehler im Solenoid-Skript: ${data}');
    });
});
```

Sobald das `activate-solenoids`-Ereignis empfangen wird, startet der `spawn`-Befehl ein externes Python-Skript (`activate_solenoids.py`), das über GPIO-Pins die Solenoids ansteuert. Die Solenoids sind für die Steuerung des Münzflusses verantwortlich: Sie sichern die Münzen im ATM oder geben sie an den Benutzer zurück, abhängig vom Status der Transaktion. Diese Funktion steht in direktem Zusammenhang mit der Münzannahme- und Rückgabelogik, die in der `backend.py`-Datei implementiert ist.

Erstellung der LNURLw

Ein weiteres zentrales Element der `main.js` ist die Verwaltung der LNURLw, die über das Backend erstellt wird. Das Frontend fordert die Erstellung einer Invoice an, wenn der Benutzer Münzen eingeworfen hat und den Zahlungsprozess starten möchte. Der folgende Code zeigt die Verwaltung der QR-Code-Erstellung für die LNURLw:

```
ipcMain.on('generate-lnurl', (event, { amount }) => {
    console.log('Erstelle LNURL fuer ${amount} Euro');
    const lnurlProcess = spawn('python3', ['../python_backend/create_lnurl.py', amount]);

    lnurlProcess.stdout.on('data', (data) => {
        event.reply('lnurl-generated', data.toString());
    });

    lnurlProcess.stderr.on('data', (data) => {
        console.error('Fehler im LNURL-Skript: ${data}');
    });
});
```

In diesem Code wird ein externes Python-Skript (`create_lnurl.py`) verwendet, um eine LNURLw zu erstellen. Der Betrag (`amount`), den der Benutzer durch Münzeinwurf festgelegt hat, wird an das Backend gesendet, wo die Invoice generiert wird. Sobald die LNURLw erstellt ist, wird sie über das Ereignis `lnurl-generated` an das Frontend zurückgesendet, damit sie dem Benutzer als QR-Code angezeigt werden kann.

Überwachung des Transaktionsstatus

Die `main.js`-Datei ist ebenfalls verantwortlich für die Überwachung des Transaktionsstatus, der vom Backend übermittelt wird. Sobald der QR-Code generiert und die Zahlung abgeschlossen wurde, sendet das Backend ein Signal, um den Erfolg der Transaktion zu bestätigen. Der folgende Code überwacht den Status der Zahlung:

```
ipcMain.on('check-payment-status', (event, lnurlId) => {
    console.log(`Ueberpruefe Zahlungsstatus fuer LNURL ID: ${lnurlId}`);
    const statusProcess = spawn('python3', ['../python_backend/check_payment_status.py',
        lnurlId]);

    statusProcess.stdout.on('data', (data) => {
        event.reply('payment-status', data.toString());
    });

    statusProcess.stderr.on('data', (data) => {
        console.error(`Fehler beim Ueberpruefen des Zahlungsstatus: ${data}`);
    });
});
```

In diesem Fall wird das Python-Skript `check_payment_status.py` ausgeführt, um den Status der Lightning-Zahlung zu überprüfen. Wenn die Zahlung erfolgreich war, wird dies dem Frontend mitgeteilt, sodass die Benutzeroberfläche den Zahlungserfolg anzeigen kann. Sollte die Zahlung fehlschlagen oder abgebrochen werden, wird das entsprechende Ereignis gesendet, um die Münzen zurückzugeben.

Abbrechen der Transaktion und Rückgabe der Münzen

Wenn der Benutzer den Transaktionsprozess abbrechen möchte oder die Zahlung fehlschlägt, wird das folgende Ereignis ausgelöst, um die Münzen zurückzugeben:

```
ipcMain.on('cancel-transaction', (event) => {
    console.log('Transaktion abgebrochen');
    const cancelProcess = spawn('python3', ['../python_backend/return_money.py']);

    cancelProcess.stdout.on('data', (data) => {
        console.log(`Muenzen zurueckgegeben: ${data}`);
    });

    cancelProcess.stderr.on('data', (data) => {
        console.error(`Fehler beim Rueckgabeskript: ${data}`);
    });
});
```

Wenn das `cancel-transaction`-Ereignis empfangen wird, startet das Python-Skript `return_money.py`, welches die Solenoids aktiviert, um die eingeworfenen Münzen an den Benutzer zurückzugeben. Diese Funktion hängt ebenfalls von der Münzannahme- und Solenoid-Logik im Backend ab, um den sicheren Rückfluss der Münzen zu gewährleisten.

Interaktion zwischen Frontend und Backend: `renderer.js`

Die Datei `renderer.js` enthält die Logik für die Interaktion zwischen der Benutzeroberfläche und dem Backend. Sie verwendet das `ipcRenderer`-Modul von Electron, um Ereignisse zu senden und zu empfangen, die auf Benutzeraktionen basieren. Diese Datei steuert die Benutzerinteraktionen, wie den Start des Münzeinwurfs, die Anzeige des QR-Codes und die Verarbeitung des Transaktionsstatus.

Start des Münzeinwurfs

Sobald der Benutzer auf den Start-Button klickt, wird der Münzeinwurf-Bildschirm aktiviert und das Signal zur Erkennung der Münzen an das Backend gesendet:

```
const { ipcRenderer } = require('electron');

startButton.addEventListener('click', () => {
    console.log("Start-Button gedrueckt");
    startScreen.style.display = 'none';
    coinScreen.style.display = 'block';
    ipcRenderer.send('start-coin-counting');
});
```

Hier wird der Start-Button überwacht, und sobald er gedrückt wird, wechselt das Frontend von `startScreen` zu `coinScreen`. Das Ereignis `'start-coin-counting'` wird an das Backend gesendet, welches den Münzeinwurfprozess über die in `backend.py` implementierte Funktion `detect_coin()` startet.

Empfang und Anzeige der Münzwerte

Die Erkennung der eingeworfenen Münzen erfolgt im Backend und wird durch das Ereignis `'coin-value'` an das Frontend gesendet. Sobald das Frontend die Daten empfängt, werden die Anzeigen für den Münzwert in Euro und Satoshis aktualisiert:

```
ipcRenderer.on('coin-value', (event, coinData) => {
    console.log(`Muenzen erkannt: ${coinData.value} Euro`);
    currentEuroValue = coinData.value;
    valueDisplay.textContent = `Gesamtsumme: ${coinData.value.toFixed(2)} Euro`;
    updateSatoshiDisplay();
    updateProceedButtonState();
});
```

Die Funktion `ipcRenderer.on()` empfängt die Münzdaten (`coinData.value`) und zeigt diese auf dem Münzeinwurf-Bildschirm an. Die Anzeige der Gesamtsumme in Euro wird dynamisch aktualisiert. Die Umrechnung in Satoshis wird durch die Funktion `updateSatoshiDisplay()` durchgeführt, die die aktuellen Euro-Werte in Satoshis konvertiert und anzeigt.

Erstellung und Anzeige des QR-Codes

Sobald der Benutzer den Münzeinwurf abgeschlossen hat und auf den Button `proceed-button` klickt, wird die Erstellung der LNURLw im Backend initiiert. Nach der Generierung der Invoice im Backend wird der QR-Code für die Lightning-Zahlung an das Frontend gesendet und angezeigt:

```
ipcRenderer.on('lnurl-qr', (event, { lnurl }) => {
    console.log(`LNURL erhalten: ${lnurl}`);
    qrCode.src = `https://api.qrserver.com/v1/create-qr-code/?size=150x150&data=${
        encodeURIComponent(lnurl)}`;
    coinScreen.style.display = 'none';
    qrScreen.style.display = 'block';
});
```

Das Ereignis `'lnurl-qr'` empfängt die generierte LNURL, die vom Backend (siehe `create_lnurl.py`) gesendet wird. Diese URL wird in einen QR-Code umgewandelt, der auf dem `qr-screen` dargestellt wird. Der Benutzer kann diesen QR-Code scannen, um die Lightning-Zahlung durchzuführen.

Überwachung des Zahlungsstatus

Nachdem der QR-Code angezeigt wurde, überprüft das Backend kontinuierlich den Zahlungsstatus. Wenn die Zahlung erfolgreich war, wird das Ereignis `'payment-status'` empfangen und der Benutzer wird über den Erfolg der Transaktion informiert:

```
ipcRenderer.on('payment-status', (event, status) => {
    if (status === 'success') {
        console.log("Zahlung erfolgreich");
        qrScreen.style.display = 'none';
        successScreen.style.display = 'block';
    }
});
```

Sobald das Backend den Erfolg der Zahlung bestätigt (siehe `check_payment_status.py`), wird das `successScreen` angezeigt, um den Benutzer über die erfolgreiche Transaktion zu informieren. Die Anzeige wechselt vom `qr-screen` zum `success-screen`, und der ATM bereitet sich darauf vor, die Münzen zu sichern.

Abbruch der Transaktion

Falls der Benutzer die Transaktion abbrechen möchte oder die Zahlung fehlschlägt, wird das Ereignis `'cancel-transaction'` ausgelöst. Dieses Ereignis signalisiert dem Backend, dass die Münzen zurückgegeben werden müssen:

```
cancelButton.addEventListener('click', () => {
    console.log("Transaktion abgebrochen");
    ipcRenderer.send('cancel-transaction');
});
```

Die Funktion `cancelButton.addEventListener()` überwacht den Abbruch-Button und sendet das Signal `'cancel-transaction'` an das Backend. Im Backend wird daraufhin das `return_money.py`-Skript ausgeführt, um die Münzen über die Solenoids an den Benutzer zurückzugeben.

Neustart des ATMs

Nachdem eine Transaktion abgeschlossen oder abgebrochen wurde, kann der Benutzer den ATM über den `restart-button` neu starten:

```
restartButton.addEventListener('click', () => {
    console.log("ATM wird neu gestartet");
    successScreen.style.display = 'none';
    startScreen.style.display = 'block';
});
```

Der `restart-button` wechselt die Anzeige zurück auf den `start-screen`, sodass der ATM für die nächste Transaktion bereit ist.

Gestaltung und Benutzeroberfläche: `style.css`

Die Datei `style.css` bildet die Grundlage für das Design und die Benutzeroberfläche des Lightning-ATMs. Durch die Definition von Layout- und Stilparametern gewährleistet sie eine ansprechende und intuitiv bedienbare Benutzeroberfläche. Im Folgenden wird eine wissenschaftliche Analyse der Gestaltungselemente vorgenommen, die die zentrale Funktionalität und das visuelle Erscheinungsbild der verschiedenen Bildschirme optimieren.

Zentrale Gestaltungselemente

Die übergreifenden Stilelemente sorgen für eine konsistente Präsentation und Benutzerführung. Der Einsatz von globalen Layout-Einstellungen, wie die absolute Positionierung und zentrierte Ausrichtung, ermöglicht ein klares und übersichtliches Design:

```
body {
    margin: 0;
    padding: 0;
    overflow: hidden; /* Hide scrollbars */
    font-family: 'Arial', sans-serif;
    position: relative;
    color: #FFFFFF; /* White text */
}
```

Die CSS-Eigenschaften stellen sicher, dass die Benutzeroberfläche vollständig ohne Scrollleisten dargestellt wird (`overflow: hidden`) und auf einer weißen Schriftfarbe basiert (`color: \#FFFFFF`). Durch die Verwendung einer Standard-Schriftart wie Arial wird die Lesbarkeit und Klarheit der Textdarstellung gewährleistet.

Hintergrundgestaltung

Ein auffälliges Hintergrundelement bildet das fest positionierte Hintergrundvideo, welches unter allen sichtbaren Elementen des Systems liegt:

```
#background-video {
    position: fixed;
    right: 0;
    bottom: 0;
    min-width: 100%;
    min-height: 100%;
    z-index: -1;
    object-fit: cover;
}
```

Durch die Eigenschaften `min-width: 100%` und `min-height: 100%` nimmt das Video die gesamte verfügbare Fläche ein und passt sich flexibel der Bildschirmgröße an. Mit einem negativen `z-index` wird das Video unter allen interaktiven Elementen angezeigt, was einen dynamischen visuellen Effekt erzeugt.

Bildschirmübergänge

Um die verschiedenen Bildschirme des Systems effektiv zu verwalten, wird die Sichtbarkeit der Screens über die CSS-Klassen `active` und `display: flex` gesteuert. Hierbei wird standardmäßig `display: none` verwendet, um nicht benötigte Screens auszublenden:

```
#coin-screen,
#qr-screen,
#success-screen {
    display: none;
}
```

Bei Aktivierung eines bestimmten Bildschirms wird die Klasse `active` hinzugefügt, wodurch die Bildschirme sichtbar und mittig ausgerichtet werden. Diese flexiblen Layout-Mechanismen sorgen für eine klare Nutzerführung und ein homogenes Design über alle Bildschirme hinweg.

Animationseffekte

Für wichtige Benutzerinteraktionen, wie das Hervorheben von Schaltflächen oder das Erscheinen von Erfolgsnachrichten, werden Animationen wie `textGlow` und `glow` verwendet. Ein Beispiel für die Gestaltung eines dynamischen Leuchteffekts:

```
@keyframes glow {
    from {
        box-shadow: 0 0 10px #F7931A, 0 0 20px #F7931A;
    }
    to {
        box-shadow: 0 0 20px #F7931A, 0 0 30px #F7931A;
    }
}
```

Durch die schrittweise Anpassung des `box-shadow` Werts entsteht ein subtiler Schimmer-Effekt, der insbesondere bei Schaltflächen für erhöhte Aufmerksamkeit sorgt. Diese Animationen verleihen der Benutzeroberfläche ein modernes und interaktives Erscheinungsbild.

Responsive Design

Um eine optimale Darstellung auf verschiedenen Geräten zu gewährleisten, wird das Layout durch responsive CSS-Eigenschaften angepasst:

```
@media screen and (max-width: 768px) {
    h1, h2 {
        font-size: 2em;
    }
    button {
        font-size: 1.2em;
        padding: 15px 25px;
    }
}
```

Hierbei werden Schriftgröße und Schaltflächenpadding bei kleineren Bildschirmen (unter 768px Breite) entsprechend verkleinert, um eine optimale Lesbarkeit und Bedienbarkeit auch auf mobilen Geräten zu garantieren.

Fazit

Die Implementierung dieser Design-Elemente in der `style.css` gewährleistet eine harmonische und konsistente Benutzeroberfläche für den Lightning-ATM. Besonders die gezielten Animationen, das flexible Layout-Management und die responsive Gestaltung tragen dazu bei, dass das System auf verschiedenen Bildschirmgrößen eine einheitliche Benutzererfahrung bietet.

Responsive Design

Um sicherzustellen, dass die Benutzeroberfläche auf verschiedenen Bildschirmgrößen gut funktioniert, wird das Layout flexibel gestaltet. Dies ermöglicht es dem ATM, sowohl auf kleinen Bildschirmen (wie Tablets) als auch auf großen Bildschirmen (wie Desktop-Monitoren) eine konsistente Benutzererfahrung zu bieten. Die Verwendung von `flex` und `vh`-Einheiten sorgt dafür, dass die Bildschirmelemente immer mittig und gut sichtbar bleiben.

4.4.4 Fazit zur Software

Die Software des Lightning-ATMs basiert auf einer klaren Trennung zwischen Frontend und Backend, wobei beide Komponenten nahtlos zusammenarbeiten, um eine sichere und benutzerfreundliche Abwicklung von Transaktionen zu gewährleisten.

Das Backend übernimmt die zentrale Steuerung der Hardware, einschließlich der Münzannahme, der Solenoid-Steuerung und der Kommunikation mit dem Lightning-Netzwerk. Die Konfiguration der GPIO-Pins des Raspberry Pi erlaubt eine präzise Steuerung der Solenoids und gewährleistet, dass Münzen zuverlässig erkannt und verarbeitet werden. Über Python-Skripte, die die Münzflüsse steuern, wird der ATM in die Lage versetzt, Münzen sicher zu speichern oder im Falle eines Transaktionsabbruchs an den Benutzer zurückzugeben. Die Erstellung der LNURLw erfolgt ebenfalls im Backend und nutzt die Lightning-Infrastruktur zur sicheren Abwicklung von Mikrotransaktionen. Die Integration von JSON zur Datenübertragung sorgt dafür, dass alle erkannten Münzwerte und Transaktionsdaten zuverlässig zwischen Frontend und Backend ausgetauscht werden.

Das Frontend wurde unter Verwendung von Electron entwickelt, einer Technologie, die es ermöglicht, plattformübergreifende Desktop-Anwendungen mit Web-Technologien wie HTML, CSS und JavaScript zu erstellen. Für den Einsatz in einem ATM ist diese Wahl besonders sinnvoll, da Electron lokal auf dem Gerät ausgeführt wird und somit keine sschnelle Internetverbindung für die Benutzeroberfläche erforderlich ist. Dies gewährleistet eine höhere Zuverlässigkeit und ermöglicht es, Transaktionen auch in Umgebungen mit langsamer Netzwerkverbindung durchzuführen. Eine funktionierende Internetverbindung ist jedoch trotzdem nötig, auch wenn die Geschwindigkeit nicht relevant ist. Das Frontend bietet eine intuitive und visuell ansprechende Benutzeroberfläche, die den Benutzer durch den gesamten Transaktionsprozess führt – von der Münzeinzahlung über die Erstellung der LNURLw bis hin zur Anzeige des Zahlungserfolgs.

Die Kommunikation zwischen Frontend und Backend erfolgt über das `ipcRenderer`- und `ipcMain`-Modul von Electron. Diese Module ermöglichen eine effiziente Interaktion zwischen der Benutzeroberfläche und den Hardwaresteuerungen im Backend. Beispielsweise wird der Münzeinwurf vom Backend gesteuert und der Wert der eingeworfenen Münzen in Echtzeit an das Frontend übertragen. Sobald der Benutzer den Münzeinwurf abgeschlossen hat, wird die generierte LNURLw als QR-Code angezeigt, den der Benutzer scannen kann, um die Zahlung zu tätigen. Nach erfolgreicher Zahlung wird der Münzvorgang im Backend abgeschlossen und der ATM kehrt in den Ausgangszustand zurück.

Die Wahl von Electron für das Frontend ist insbesondere für einen ATM sinnvoll, da die Software lokal auf dem Gerät läuft und dadurch unabhängig von ständigen externen Netzwerkverbindungen funktioniert und zudem auch möglichen Cyberattacken vorbeugt. Dies erhöht die Zuverlässigkeit und reduziert mögliche Sicherheitsrisiken, die bei einem cloudbasierten System auftreten könnten. Zudem bietet Electron eine flexible und benutzerfreundliche Entwicklungsumgebung, die es ermöglicht, schnelle Anpassungen an der Benutzeroberfläche vorzunehmen.

Zusammenfassend lässt sich sagen, dass die Software des Lightning-ATMs durch die Trennung von Frontend und Backend eine robuste und effiziente Architektur bietet. Das Backend übernimmt die vollständige Kontrolle über die Hardware und die Zahlungslogik, während das Frontend eine benutzerfreundliche und optisch ansprechende Interaktion ermöglicht. Die Verwendung von Electron im Frontend und Python im Backend stellt sicher, dass der ATM sowohl lokal effizient als auch technisch flexibel agieren kann.

Der Quellcode der entwickelten Software kann zudem für besseres Verständis auf Github eingesehen werden [50].

5 Demonstration des Lightning-ATM Konzepts

Nach erfolgreichem Zusammenbau des Lightning-ATMs sowie der Implementierung und erfolgreichen Testung der Software und der Konnektivität zwischen den einzelnen Hardwarekomponenten, unterstützt eine durch Bilder gestützte Demonstration die Veranschaulichung des Konzepts. Dabei wird die vollständige Prozesskette aus der Sicht eines Nutzers, der Bitcoin erwerben möchte, Schritt für Schritt beschrieben.

Damit ein potenzieller Kunde den ATM fehlerfrei nutzen kann, müssen einige grundlegende Voraussetzungen erfüllt sein. Zu diesen zählen insbesondere:

- **Ausreichende Liquidität:** Die hinterlegte LnBits Lightning Wallet muss über eine ausreichende Menge an Satoshis verfügen. Die Wallet selbst kann bei Bedarf in der Konfigurationsdatei ‘config.js‘ angepasst werden.
- **WLAN-Internetverbindung:** Der ATM muss über eine stabile WLAN-Verbindung verfügen, um die Transaktionen erfolgreich abzuwickeln.
- **Stromversorgung:** Es ist entweder ein 240V oder ein 110V Stromanschluss erforderlich, da das Netzteil beide Spannungsarten unterstützt.

Sind diese Voraussetzungen erfüllt, steht einer erfolgreichen Nutzung des Lightning-ATMs nichts mehr im Wege.

Die folgenden Prozessketten beschreiben den gesamten Funktionsumfang des ATMs und werden durch das untenstehende Prozessdiagramm illustriert:

Standard-Prozess

Der Standardprozess beschreibt die reguläre Abwicklung eines Kaufs von Satoshis über den ATM.

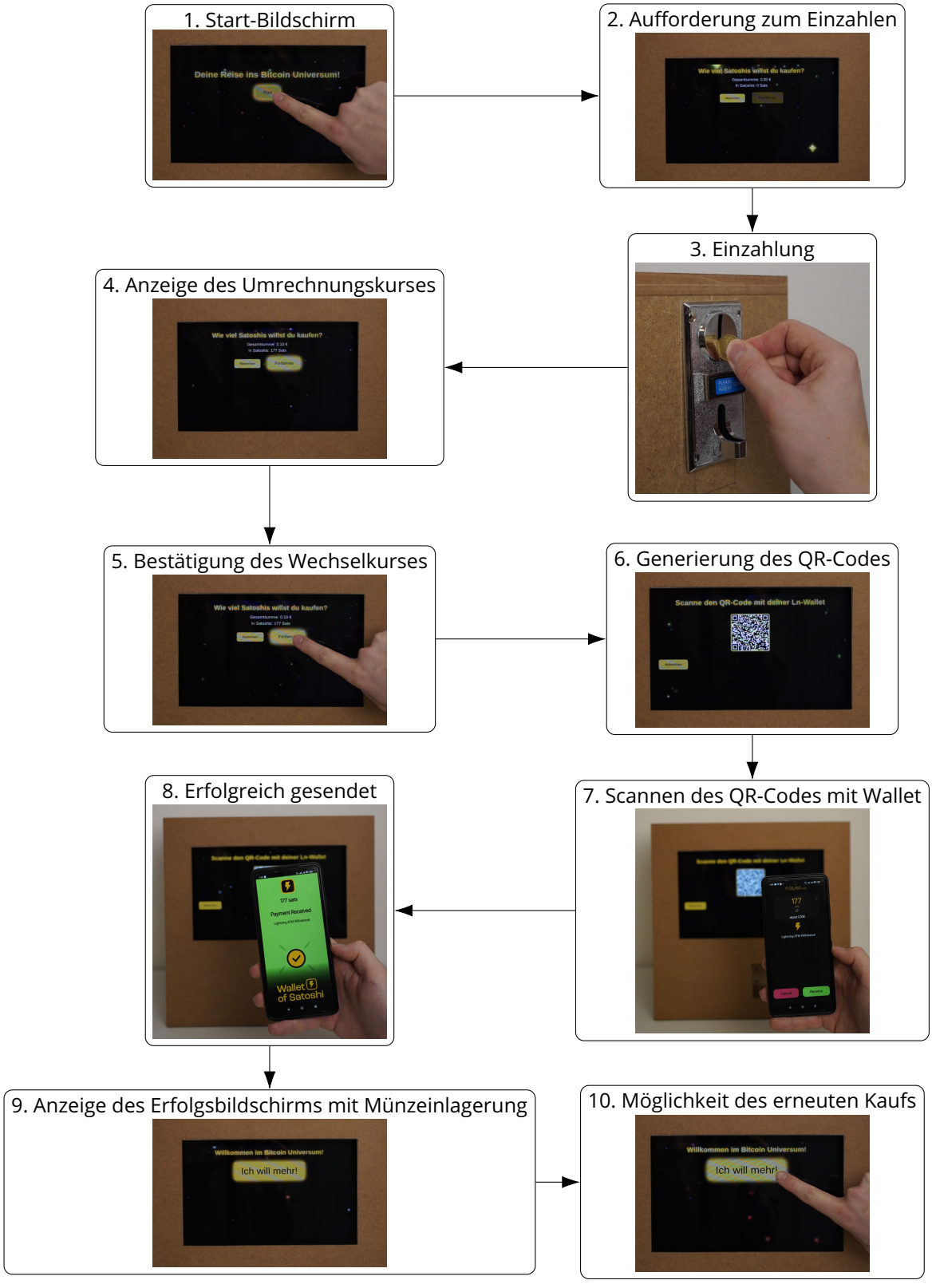


Abbildung 5.1: Standard-Prozess des Lightning-ATMs

Prozessabbruch vor oder während der Einzahlung

Der Prozessabbruch ist ein entscheidender Bestandteil, da Benutzer aus unterschiedlichen Gründen die Zahlung nicht fortsetzen könnten (z. B. unerwartete Verzögerungen oder technische Probleme). Um eine optimale Benutzerfreundlichkeit zu gewährleisten, wurde die Münzflussmechanik integriert, die sicherstellt, dass bei einem Abbruch die bereits eingezahlten Münzen automatisch zurückgegeben werden und der ATM in den Ausgangszustand versetzt wird.

Die Prozesskette umfasst 5 Schritte:

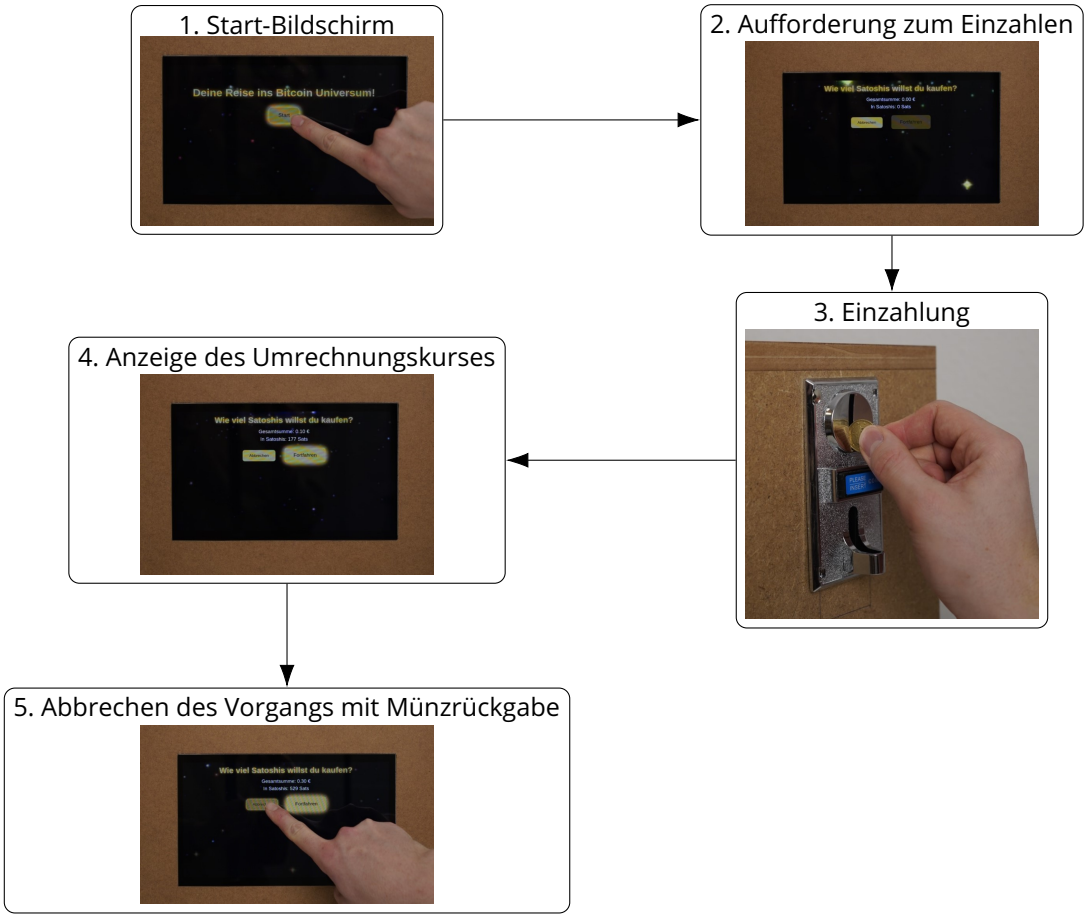


Abbildung 5.2: Prozessabbruch vor oder während der Einzahlung

Prozessabbruch nach Einzahlung und QR-Code Generierung

Dieser Prozess beschreibt den Fall, in dem der Nutzer den Vorgang abbricht, nachdem die Einzahlung erfolgt ist und ein QR-Code generiert wurde. Der Prozess umfasst insgesamt 7 Schritte:

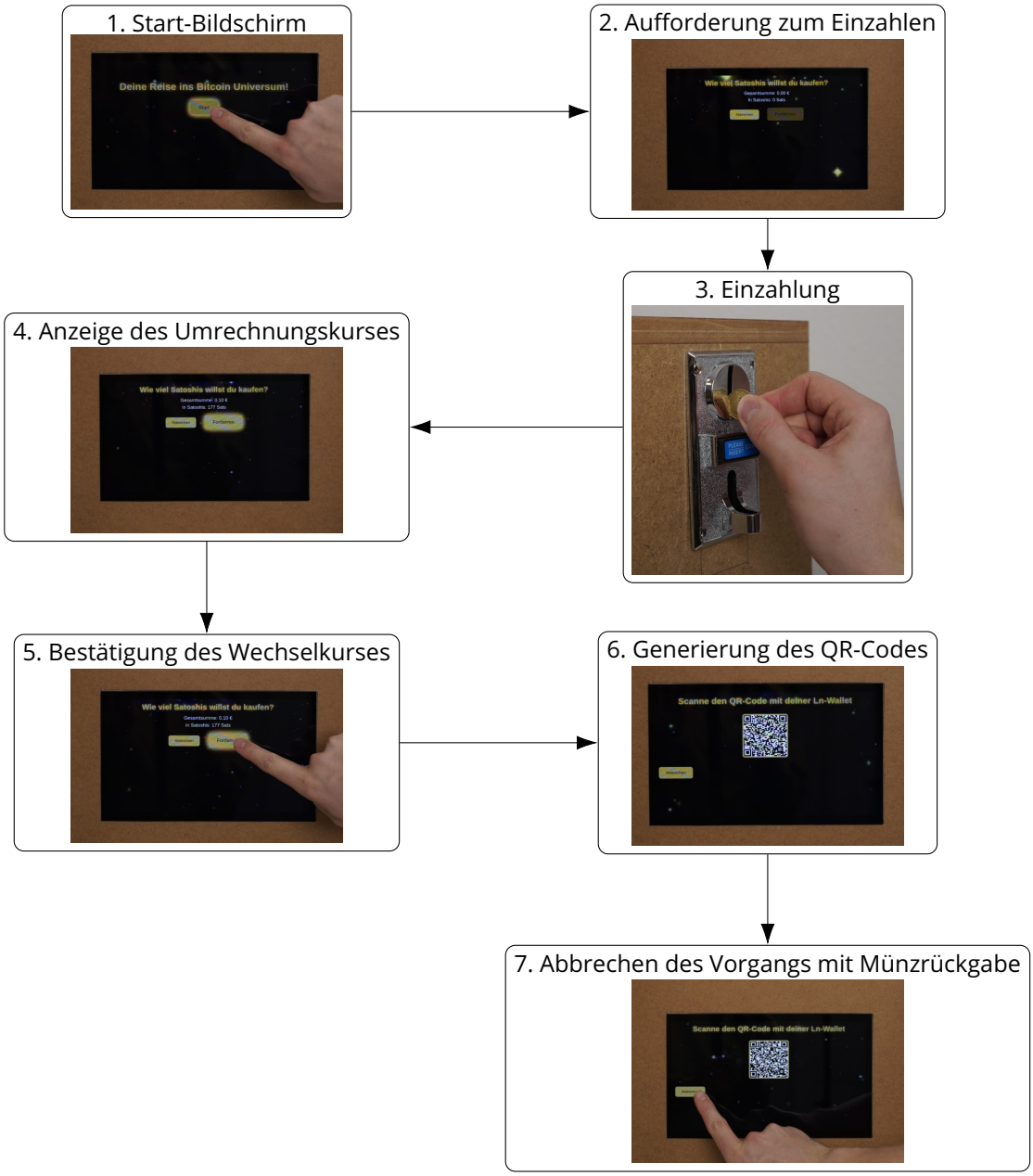


Abbildung 5.3: Prozessabbruch nach Einzahlung und QR-Code Generierung

Die Demonstration des Lightning-ATM-Konzepts zeigt drei zentrale Prozessszenarien: den Standard-Prozess, den Prozessabbruch vor oder während der Einzahlung und den Prozessabbruch nach Einzahlung und QR-Code Generierung. Falls Probleme im Backend bzgl. Liquidität oder Internetprobleme Auftreten, können die Abbrechen-Buttons trotzdem ausgeführt werden. Ein Fehlschlagen der Invoices durch Liquiditöt wird dem Benutzer beim scannen der LNURLw mitgeteilt. Bei fehlendem Internet wird kein QR-Code generiert.

Im Standard-Prozess wird der Nutzer schrittweise von der Auswahl der Einzahlungssumme bis zur Auszahlung in Satoshis geführt. Abbruchszenarien stellen sicher, dass Münzen jederzeit korrekt zurückgegeben und der ATM in einen sicheren Ausgangszustand versetzt wird. Besonders wichtig ist die Rückerstattung nach der QR-Code-Generierung, um Missverständnisse und Verluste zu vermeiden.

Insgesamt gewährleisten die klar strukturierten Prozesse eine einfache Bedienung, hohe Sicherheit und Flexibilität für den Nutzer. Die Abläufe sind so konzipiert, dass sie potenzielle Fehler minimieren und eine reibungslose Interaktion ermöglichen.

Ein eigens für die Demonstration erstelltes Video ist auf Youtube zu sehen [51].

6 Abschließende Konzeptbewertung

Insgesamt kann das Konzept des entwickelten Lightning-ATMs als ein gelungener Prototyp zur praktischen Umsetzung und Demonstration eines Bitcoin-Zugangspunkts bewertet werden. Die Arbeit zeigt, dass es möglich ist, einen voll funktionsfähigen Automaten mit grundlegenden Mitteln und geringem Budget zu realisieren, der das Lightning-Netzwerk als schnelles und kostengünstiges Transaktionsmedium nutzt.

Die Entwicklung des ATMs zeigt eine prototypische Konzeption und Implementierung der Hardware- und Softwarekomponenten. Die Nutzung des Lightning-Netzwerks ermöglicht schnelle Abwicklungen, die besonders für kleine Transaktionen und den täglichen Gebrauch von Bedeutung sind. Ein besonderer Erfolg ist der modulare Aufbau, der eine einfache Anpassung und Erweiterung des Systems zulässt. Darüber hinaus bietet der ATM eine benutzerfreundliche Schnittstelle und einfache Bedienung, die auch technisch weniger versierten Nutzern den Zugang zum Bitcoin-Ökosystem erleichtert.

Die Umsetzung brachte jedoch auch einige Herausforderungen mit sich, die während des Projekts bewältigt werden mussten. Insbesondere traten Probleme bei der Software-Entwicklung auf, beispielsweise in Bezug auf das Frontend-Design und die fehlerhafte Anzeige von Benutzerinformationen. Diese wurden durch iteratives Testen und Anpassungen in der Software gelöst, aber zeigten den Bedarf an zusätzlichen Optimierungen und stabileren Schnittstellen.

Auch auf der Hardware-Seite gab es unerwartete Schwierigkeiten. Zu starke Federn im Münzmechanismus führten zu unregelmäßigem Verhalten beim Coin-Handling, und sehr kleine Lötstellen erhöhten das Risiko von Kurzschlüssen. Diese Probleme konnten durch detaillierte Nachbesserungen und manuelle Anpassungen behoben werden, erforderten jedoch erheblichen Zeitaufwand. Trotz der erfolgreichen Realisierung bleibt in einigen Bereichen Potenzial zur weiteren Optimierung. Der Prototyp wurde mit einfachen Werkzeugen und ohne den Einsatz professioneller Werkstattausrüstung gebaut, was sich in der Verarbeitungsqualität, insbesondere bei der Holzverarbeitung und den Gehäusekanten, bemerkbar macht. Eine Verbesserung der Verarbeitungspräzision, beispielsweise durch die Nutzung von professionellem 3D-Druck oder maschineller Holzbearbeitung, könnte die Stabilität und Langlebigkeit des Automaten signifikant steigern. Auch der Coin-Mechanismus könnte durch den Einsatz robusterer Materialien oder präziserer Fertigungsverfahren für den Dauerbetrieb ausgelegt werden.

Zusammenfassend erfüllt der entwickelte Lightning-ATM die Erwartungen als Prototyp und Demonstrationsobjekt und bietet eine stabile Grundlage für zukünftige Weiterentwicklungen. Er zeigt, dass ein kostengünstiger und zugänglicher Einstiegspunkt in das Bitcoin-Ökosystem möglich ist, der das Potenzial besitzt, den Zugang zu dezentralisierten Finanzsystemen wie Bitcoin weiter zu erleichtern. Trotz einiger technischer Hürden und Nachbesserungen bietet das Konzept wertvolle Erkenntnisse und Lernerfahrungen in den Bereichen Elektrotechnik, Software-Integration und Hardware-Design. Der Prototyp kann somit als Ausgangspunkt für weiterführende Projekte dienen und bietet Potenzial für eine professionelle Serienentwicklung, die das Konzept auf ein höheres Qualitätsniveau heben kann.

Der Abschluss des Projekts markiert nicht nur den erfolgreichen Bau eines innovativen Prototyps, sondern liefert auch entscheidende Einblicke und praktische Erfahrungen, die für künftige Entwicklungen im Bereich des Bitcoin- und Lightning-Ökosystems wertvoll sind.

Literaturverzeichnis

[1] Bitpanda, *Bitcoin-ETFs: ein Überblick*, https://www.bitpanda.com/academy/de/lektionen/bitcoin-etfs-ein-uberblick/, Accessed: [23.10.2024].

[2] M. Leder und B. Stiller, *Design, Implementation and Usability Evaluation of the AfriBit Bitcoin Lightning wallet*, https://files.ifi.uzh.ch/CSG/staff/vonderassen/extern/theses/BA_Leder.pdf, Accessed: [23.10.2024].

[3] A. M. Antonopoulos und O. Osuntokun, *Mastering the Lightning Network*. O'Reilly Media, 2021, S. 145–167, ISBN: 978-1492054863.

[4] S. Nakamoto, *Bitcoin: A Peer-to-Peer Electronic Cash System*, https://bitcoin.org/bitcoin.pdf, Accessed: [20.09.2024], 2008.

[5] Statista, *Gesamtzahl aller Bitcoin-Transaktionen weltweit von Februar 2017 bis September 2024*, https://de.statista.com/statistik/daten/studie/315084/umfrage/gesamtzahl-aller-bitcoin-transaktionen-weltweit/, Accessed: [14.10.2024].

[6] A. M. Antonopoulos, *Mastering Bitcoin: Unlocking Digital Cryptocurrencies*, 2. Aufl. O'Reilly Media, 2017, S. 63–102, ISBN: 978-1491954386.

[7] A. Narayanan, J. Bonneau, E. Felten, A. Miller und S. Goldfeder, *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*. Princeton University Press, 2016, S. 47–74, ISBN: 978-0691171692.

[8] Bitcoin.org, *The Parts Of A Transaction*, https://developer.bitcoin.org/devguide/transactions.html#the-parts-of-a-transaction, Accessed: [25.09.2024].

[9] J. Poon und T. Dryja, *The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments*, https://lightning.network/lightning-network-paper.pdf, Accessed: [20.09.2024], 2016.

[10] A. M. Antonopoulos, *Mastering Bitcoin: Unlocking Digital Cryptocurrencies*, 2. Aufl. O'Reilly Media, 2017, S. 153–172, ISBN: 978-1491954386.

[11] K. Croman, C. Decker, I. Eyal u. a., „On Scaling Decentralized Blockchains (A Position Paper)", in *Financial Cryptography and Data Security: FC 2016 International Workshops*, Springer, 2016, S. 106–125. DOI: 10.1007/978-3-662-53357-4_8.

[12] A. Narayanan, J. Bonneau, E. Felten, A. Miller und S. Goldfeder, *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*. Princeton University Press, 2016, S. 89–97, ISBN: 978-0691171692.

[13] BIP 125, *Enable Replace-By-Fee, Opt-In Only*, https://github.com/bitcoin/bips/blob/master/bip-0125.mediawiki, Accessed: [12.09.2024], 2015.

[14] A. M. Antonopoulos, *Mastering Bitcoin: Unlocking Digital Cryptocurrencies*, 2. Aufl. O'Reilly Media, 2017, S. 173–195, ISBN: 978-1491954386.

[15] A. Narayanan, J. Bonneau, E. Felten, A. Miller und S. Goldfeder, *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*. Princeton University Press, 2016, S. 149–176, ISBN: 978-0691171692.

[16] C. Decker und R. Wattenhofer, „Information Propagation in the Bitcoin Network", in *IEEE P2P 2013 Proceedings*, IEEE, 2013, S. 1–10. DOI: 10.1109/P2P.2013.6688704.

[17] mempool.space, *Bitcoin Mining Dashboard*, https://mempool.space/de/mining, Accessed: [25.09.2024].

[18] U.Today, *Here's How Much It Costs To 51% Attack Bitcoin (BTC)*, https://u.today/heres-how-much-it-costs-to-51-attack-bitcoin-btc, Accessed: [20.09.2024], 2022.

[19] Braiins, *How Much Would it Cost to 51% Attack Bitcoin?*, https://braiins.com/blog/how-much-would-it-cost-to-51-attack-bitcoin, Accessed: [20.09.2024], 2023.

[20] Coinmarketcap, *Transaktionen pro Sekunde (TPS)*, https://coinmarketcap.com/academy/de/glossary/transactions-per-second, Accessed: [14.10.2024].

[21] Decrypt, *Bitcoin's Biggest Upgrade Since 2017: Taproot Just Went Live*, https://decrypt.co/86027/bitcoins-biggest-upgrade-since-2017-taproot-just-went-live, Accessed: [20.09.2024], 2021.

[22] A. Narayanan, J. Bonneau, E. Felten, A. Miller und S. Goldfeder, *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*. Princeton University Press, 2016, S. 196–229, ISBN: 978-0691171692.

[23] Song, Yu and Xiong, Ao and Qiu, Xuesong and Guo, Shaoyong and Wang, Dong and Li, Da and Zhang, Xin and Kuang, Yue, „A Blockchain-Based Method for Optimizing the Routing of High-Frequency Carbon-Trading Payment Channels", *Electronics*, Jg. 12, S. 2586, Juni 2023. DOI: 10.3390/electronics12122586.

[24] J. Goro, *The Economics of Lightning Network Fees*, https://medium.com/social-evolution/the-economics-of-lightning-network-fees-76f0926da82, Accessed: [20.09.2024], 2018.

[25] A. M. Antonopoulos und O. Osuntokun, *Mastering the Lightning Network*. O'Reilly Media, 2021, S. 192–214, ISBN: 978-1492054863.

[26] R. Pickhardt und C. Decker, *Optimally Reliable & Cheap Payment Flows on the Lightning Network*, https://arxiv.org/abs/2107.05322, 2021.

[27] E. Tairi, P. A. Moreno-Sánchez und M. Maffei, „A2L: Anonymous Atomic Locks for Scalability in Payment Channel Hubs", in *IEEE Symposium on Security and Privacy*, Accessed: [20.09.2024], 2021. DOI: 10.1109/SP40001.2021.00111.

[28] G. Avarikioti, O. Litos und R. Wattenhofer, „Cerberus Channels: Incentivizing Watchtowers for Bitcoin", in *Financial Cryptography and Data Security*, Accessed: [20.09.2024], 2020. DOI: 10.1007/978-3-030-51280-4_19.

[29] A. Gangwal, H. R. Gangavalli und A. Thirupathi, *A Survey of Layer-Two Blockchain Protocols*, https://arxiv.org/pdf/2204.08032, Accessed: [20.09.2024], 2022.

[30] R. Pickhardt und S. Richter, *Optimally Reliable & Cheap Payment Flows on the Lightning Network*, https://ar5iv.org/pdf/2107.05322.pdf, Accessed: [20.09.2024], 2021.

[31] SWIFT, *Cross-border payments, transformed*, https://www.swift.com/our-solutions/corporates/network/cross-border-payments-transformed, Accessed: [20.09.2024], 2021.

[32] LnBits, *Fully responsive Web Wallet*, https://lnbits.com/, Accessed: [14.10.2024].

[33] Bleskomat, *Bleskomat Coins ATM*, https://shop.bleskomat.com/product/bleskomat-coins-atm/, Accessed: [20.09.2024].

[34] 21isenough, *LightningATM GitHub Repository*, https://github.com/21isenough/LightningATM, Accessed: [20.09.2024].

[35] Raspberry Pi, *Raspiberry Pi 4 Model B*, https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/, Accessed: [25.09.2024].

[36] HAMTYSAN, *10.1 Zoll HDMI Monitor*, https://www.amazon.de/dp/B0B9M5SCG4, Accessed: [25.09.2024].

[37] Heschen, *Elektromagnet HS-0530B DC12V 5N 10mm*, https://www.amazon.de/dp/B07MJJB12M, Accessed: [25.09.2024].

[38] Kungfu Mall, *DC 12V 35mm Long Stroke Push Pull Solenoid*, https://www.amazon.de/dp/B07SZCHYF7, Accessed: [25.09.2024].

[39] FTVOGUE, *Münzprüfer für Automaten*, https://www.amazon.de/dp/B07QCWPBG5, Accessed: [25.09.2024].

[40] Kingwen, *Schaltnetzteil 12V 20A 240VAC auf 12VDC*, https://www.amazon.de/dp/B08QDDY412, Accessed: [25.09.2024].

[41] Tiardey, *12V zu 5V Konverter DC-DC Step Down Modul*, https://www.amazon.de/dp/B09PFV3SWN, Accessed: [25.09.2024].

[42] TOpitec, *Steckplatine Breadboard, groß 55 x 165 mm*, https://www.opitec.ch/technisches-zubehoer/elektronische-bauteile/platinen/steckplatine-breadboard-gross-55-x-165-mm.html, Accessed: [25.09.2024].

[43] Bojack, *2N2222 NPN Allzweck-Transistoren*, https://www.amazon.de/gp/product/B07T51C9ZC, Accessed: [25.09.2024].

[44] VooGenzek, *12V 1-Kanal Relaismodul Low Level Trigger*, https://www.amazon.de/gp/product/B0CG2VWKVP, Accessed: [25.09.2024].

[45] Eiechip, *Metallschichtwiderstandssatz mit Box 10Ohm 1M*, https://de.aliexpress.com/item/1005006143425537.html, Accessed: [25.09.2024].

[46] dieelektronikerseite.de, *Der Transistor-Ein Tausendsassa*, https://www.dieelektronikerseite.de/Lections/Der%20Transistor%20-%20Ein%20Tausendsassa.htm, Accessed: [25.09.2024].

[47] EasyEDA, *Online PCB-Editor*, www.easyeda.com, Accessed: [17.10.2024].

[48] JLCPCB, *Easier One-Stop Solutions for PCB and PCBA Services*, www.jlcpcb.com, Accessed: [17.10.2024].

[49] Jan Bittner, *LNATMV1*, https://oshwlab.com/lnatmpcb/lnatm, Accessed: [17.10.2024].

[50] Jan Bittner, *LNATMV1-Code*, https://github.com/JB1tcoin/LNATM, Accessed: [21.10.2024].

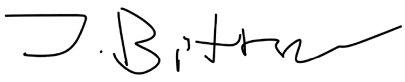
[51] Jan Bittner, *Demonstration LNATM*, https://youtu.be/QmhCz1x3pBQ, Accessed: [21.10.2024].

Eidesstattliche Erklärung

Hiermit versichere ich – Jan Leon Bittner – an Eides statt, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Sämtliche Stellen der Arbeit, die im Wortlaut oder dem Sinn nach Publikationen oder Vorträgen anderer Autoren entnommen sind, habe ich als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt oder anderweitig veröffentlicht.



Mittweida, 23. Oktober 2024	
Ort, Datum	Jan Leon Bittner