Martin-Niemöller Gesamtschule

Apfelstraße 210, 33611 Bielefeld

Name: Simon Krampe

Fach: Grundkurs Informatik

Lehrer: Herr Andy Notarnicola

Abgabe: 17.02.2022

Thema:

Internet of Things – Entwicklung eines DIY Taschenrechners mit dem Raspberry Pi Zero

Inhalt

1 Einleitung

Da ich mich seit einiger Zeit stärker für Programmieren und ähnliches interessiere, habe ich mir in Bezug auf viel Technik die Frage gestellt wie sie funktioniert und wie schwer es wäre etwas Ähnliches herzustellen, sowie zu programmieren. Da in der Schule beinahe täglich Taschenrechner benutzt werden kam ich auf die Idee, dass diese sich gut eignen um der Antwort auf diese Frage näher zu kommen. In dieser Facharbeit werde ich den Prozess und die Schwierigkeiten der Entwickelung eines Taschenrechners darstellen. Zusätzlich habe ich erfahren wo die Schwierigkeiten liegen und auch in welchen Aspekten es einfacher ist, als man anfangs erwarten würde.

> *"Programming isn't about what you know; it's about what you can figure out." – Chris Pine*

Um einen eigenen Taschenrechner selber zu bauen werde ich auf den Raspberry Pi Zero[1] zurückgreifen, was ich im nachfolgenden Kapitel weiter begründen werde. Das Programm wird in Java verfasst, da ich bereits Vorerfahrung mit dieser Sprache habe, aber meine Kenntnisse gerne erweitern würde. Außerdem wird im Laufe dieser Facharbeit ebenfalls ein speziell für diesen Zweck individualisiertes PCB[2] entwickelt, sowie gebraucht. Das bedeutet, dass ich eine Platine designen werde, die die Auslesung eines Tasten-Schaltkreises, sowie die Ansteuerung eines I2C[3] LCD[4]-Displays ermöglicht. Diese Platine wird zusammen mit dem Bildschirm und Tasten etwa aussehen wie in Abbildung 1 dargestellt.

2 Raspberry Pi Zero

Für dieses Projekt habe ich mich dazu entschieden einen Raspberry Pi Zero zu nutzen, da er für seine Größe und Preis eine sehr gute Leistung bietet. Ebenfalls ist er mit allen Komponenten ausgestattet die ein regulärer Computer ebenfalls besitzt. Der

---

[1] Meinen habe ich auf https://www.berrybase.de/raspberry-pi/raspberry-pi-computer/boards/raspberry-pi-zero-wh gekauft (shorturl.at/dmHMN)

[2] **P**rinted **C**ircuit **B**oard leitet Spannungen zwischen elektronischen Bauteilen und ist auch als Leiterplatte oder Platine bekannt

[3] **I**nter-**I**ntegrated **C**ircuit ist ein Kommunikationsprotokoll mit geringer Bandbreite

[4] **L**iquid **C**ristal **D**isplay

Raspberry Pi Zero ist dank der 1 GHz[1] starken Single-Core CPU[2], seiner GPU[3] und den 512MB RAM[4] in der ein reguläres Betriebssystem, wie Linux, auszuführen welches er in HD über den Mini-HDMI Anschluss anzeigen kann. Jedoch sind die GPIO[5]s das wichtigste Element des Raspberry Pi, da sie ermöglichen externe Geräte, wie Tasten und simple Bildschirme, anzusteuern.

Insgesamt hat der Raspberry Pi 40 Pins[6], wovon 26 GPIO Pins sind. Über Programme ist es möglich diese GPIO Pins zu steuern. Es gibt einige Pins die immer 3.3V[7] oder 5V ausgeben. Es gibt ebenfalls Ground-Pins[8] die als Sicherheitsmaßnahme genutzt werden und Kurzschlüsse vermeiden sollen. Sie „saugen" also praktisch die Elektrizität auf um Komponenten zu schützen. Die wahrscheinlich wichtigsten Pins sind die GPIOs, da sie in der Lage sind Spannung auszugeben, sowie zu Spannung zu lesen.

2.1 Nutzung der GPIOs mit Button Matrix

In meinem Projekt benutze ich zwei verschiedene Anwendungsmöglichkeiten der GPIOs, da ich die Tasten auslese und den Bildschirm ansteuere. Zuerst werde ich mich hier auf das Auslesen der Tasten fokussieren, da dies ein sehr wichtiger Aspekt dieses Projektes ist.

Alle Tasten sind über Spalte[9] und Zeile[10] definiert, wie in Abbildung 2 dargestellt. Taste ‚1' entspricht also Spalte 1, Zeile 1. Sobald eine Taste gedrückt wird, werden die Leitungen der Spalten und Zeilen verbunden.

---

[1] Anzahl von Vorgängen pro Sekunde

[2] **C**entral **P**rocessing **U**nit – Der Prozessor eines Computers

[3] **G**raphics **P**rocessing **U**nit – Die Grafikkarte eines Computers

[4] **R**andom-**A**ccess **M**emory – Sehr schneller Speicher

[5] **G**eneral **P**urpose **I**nput/**O**utput ist eine Allzweckein- und ausgabe

[6] Ein elektrischer Kontakt

[7] Volt ist eine elektrische Spannung

[8] Pins, welche elektrische Spannung ableiten

[9] In der Abbildung als gelb dargestellt

[10] In der Abbildung als rot dargestellt

Damit das Programm erkennen kann welche Taste gedrückt wurde, müssen die jeweiligen GPIOs entsprechend konfiguriert werden. Die GPIOs die an die Zeilen angeschlossen sind, werden als Input Pins mit „Pull Up“ Resistenz initialisiert und können deshalb eine Änderung der Spannung erkennen, während die GPIOs für die Spalten als Output Pin initialisiert werden und ohne Einwirkung vom Programm konstant eine hohe Spannung ausgeben. „Pull Up“ Resistenz bedeutet, dass erkannt wird, wenn die anliegende Spannung niedrig wird und da die Spalten eine hohe Spannung ausgeben wird noch nicht erkannt, ob ein Knopf gedrückt wurde. Deshalb werden die Spalten nacheinander auf eine niedrige Spannung gesetzt und alle Zeilen überprüft, ob eine niedrige Spannung anliegt. Sollte bei keiner der Zeilen eine niedrige Spannung anliegen, wird die Spalte wieder auf eine hohe Spannung gesetzt, während die nächste Spalte auf eine niedrige gesetzt wird, worauf wieder alle Zeilen auf eine niedrige Spannung überprüft werden. Sobald eine Zeile eine niedrige Spannung erkennt, weiß das Programm welche Taste gedrückt wurde, da es weiß welche Zeile die niedrige Spannung erkannt hat und welche Spalte eine niedrige Spannung ausgibt. (Stern, kein Datum)

2.2 Nutzung der GPIOs mit I2C LCD-Display

In diesem Projekt wird das LCD-Display mit vier Pins verbunden, wobei zwei davon für die Stromversorgung und zwei für den Datentransfer genutzt werden. Die für die Stromversorgung zuständigen Pins sind GND[1], sowie 3.3 VDC[2] der konstant 3.3 Volt ausgibt. Diese beiden Pins sind für das Programm nicht von Interesse, da es nicht möglich ist die Ausgabe von diesen Pins zu verändern.

Die Pins, welche für den Datentransfer genutzt werden sind die speziellen GPIOs Nummer 8[3] und Nummer 9[4]. Diese beiden GPIOs werden speziell für das „Inter-Integrated Circuit“ Kommunikations-Protokoll verwendet, wobei GPIO 8 der „SDA“[5], und GPIO 9 der „SCL“[6] Pin ist. Der SDA Pin ist für das senden, sowie

[1] Ground Pin, der Pin Nummer 6 entspricht

[2] Power Pin, der Pin Nummer 1 entspricht

[3] Pin Nummer 3

[4] Pin Nummer 5

[5] Serial Data

[6] Serial Clock

empfangen von Daten zuständig, während der SCL Pin ein „Takt-Signal“ sendet welches angibt wann Daten gesendet werden.
Ein klassischer Datentransfer über I2C läuft ab wie in Abbildung 3 dargestellt.

Die Start-Bedingung ist, dass der SDA-Anschluss von einer hohen Spannung zu einer niedrigen wechselt, während der SCL-Anschluss eine hohe Spannung ausgibt. Darauf wird die Adresse des angesteuerten Displays übertragen, damit definiert ist welcher Bildschirm angesteuert wird, wenn mehrere angeschlossen sind. In diesem Szenario wird zwar nur ein Bildschirm verwendet, allerdings wird die Adresse trotzdem übertragen. Das folgende Bit gibt an ob etwas auf den Bildschirm geschrieben werden soll, in welchem Fall der SDA-Anschluss eine niedrige Spannung ausgibt, oder ob Daten von dem Bildschirm gelesen werden sollen, in welchem Fall eine hohe Spannung ausgegeben wird.
Im Anschluss sendet der Bildschirm einen ACK[1]- oder NACK[2] Bit zurück um zu bestätigen, dass die Daten empfangen wurden oder nicht. Das ACK Bit steht dafür, dass die Daten erfolgreich empfangen wurden, während das NACK Bit dafür steht, dass der Datentransfer fehlgeschlagen ist und das senden erneut versucht, oder abgebrochen wird. Dieses Bit wird vom Empfänger nach der Übertragung von Daten zurückgesendet, was in dem Szenario des Taschenrechners immer der Bildschirm ist, da keine Daten vom Bildschirm abgefragt werden müssen.
Sobald der ACK empfangen wurde, wird ein Kontroll-Byte gesendet. Wenn das erste Bit eine hohe Spannung hat bedeutet dies, dass nicht nur ein Datenpaket gesendet wird sondern noch ein weiteres danach folgt. Hat das erste Bit jedoch eine niedrige Spannung weiß der Empfänger, dass das folgende Datenpaket das letzte ist. Nach dem Kontroll-Byte folgt wieder ein ACK- oder NACK Bit.
Wenn der ACK empfangen wurde, werden die Daten übertragen. Der Datentransfer erfolgt bei dem I2C-Protokoll immer in 8-Bit. Sobald 8 Bits übertragen wurden erfolgt ein ACK- oder NACK Bit von dem Empfänger.
In dem Fall, dass das erste Bit des Kontroll-Bytes hohe Spannung hatte werden ein weiterer Kontroll-Byte, der auch einen weiteren Datentransfer ankündigen kann, und ein weiterer Daten-Byte (8-Bits) übertragen. Sobald das erste Bit eines Kontroll-Bytes eine niedrige Spannung hat weiß der Empfänger, dass der nächste Daten-Byte

[1] Das **Ack**nowledgement-Bit ist die Empfangsbestätigung

[2] Das **N**o **Ack**nowledgement-Bit zeigt an dass nicht alles Empfangen wurde

der letzte ist.
Zum Schluss folgt die Stopp-Bedingung, die daraus besteht, dass der SDA-Anschluss von einer niedrigen Spannung auf eine hohe wechselt, während der SCL-Anschluss eine hohe Spannung ausgibt. Damit wird die Datenübertragung beendet. (Waveshare, 2021)

2.3 Implementation der Button Matrix

Das Programm speichert das Layout der Tasten in einem zweidimensionalen-Char-Array, welches direkt mit allen Zeichen initialisiert wird.

Für das Auslesen und Ansteuern der GPIOs benutze ich die Pi4J[1] Library. Sie ermöglicht einen Zugriff auf die Pins des Raspberry mithilfe von Java, was sehr hilfreich ist, da mein Taschenrechner in Java geschrieben ist. Im „klassischen“ Fall werden die GPIOs über Python gesteuert, aber da Java meiner Meinung nach für komplexe Programme, aufgrund höherer Übersichtlichkeit, besser geeignet ist, wird Java für den Großteil des Projektes genutzt. Um zu erkennen welche Taste gedrückt wurde muss, wie in Kapitel 2.1 bereits erläutert, durch die Spalten und Zeilen iteriert werden. Dafür gibt es die „Listener“-Klasse, welche in Abbildung 4 dargestellt ist.

Die Spalten werden mit einer for()-Schleife iteriert, wobei die „äußere“ Schleife die Variable „i“ hochzählt. Sobald die Spalte auf niedrig gesetzt wurde, wird für jede Zeile überprüft, ob diese niedrig ist. Dabei wird die Variable „j“ hochgezählt. Sobald also erkannt wird, dass eine Taste gedrückt wurde weiß das Programm welche Taste in welcher Spalte welcher Zeile gedrückt wurde. Das entsprechende Zeichen wird darauf aus dem Char-Array abgerufen.

Mit diesem Zeichen wird darauf die „add()“-Methode der „StringComposer“-Klasse aufgerufen, welche in Abbildung 5 dargestellt ist. Diese Methode speichert den String[2], der ausgerechnet werden soll, ruft die Rechenmethode auf und zeigt die Eingaben, sowie das Ergebnis auf dem Bildschirm an.

[1] Pi4J steht für „Pi für Java“

[2] Ein Datentyp, der mehrere Chars speichert

Wenn das Eingegebene Zeichen ein „#“ ist, dann wird das Ergebnis mit der Rechenmethode berechnet und auf dem Bildschirm angezeigt. Darauf werden alle Variablen zurückgesetzt, da die Methode und die Variablen „static“[1] sind und sich die Werte nach dem berechnen nicht automatisch zurücksetzten. Sobald die Variablen zurückgesetzt sind wird die Methode abgebrochen, da das Ergebnis bereits berechnet wurde und die Rechenanfrage nicht weiter modifiziert werden muss.

Sollte das Zeichen also kein „#“ sein wird das neue Zeichen an die Rechenanfrage gehängt, sowie zusammen mit den restlichen Zeichen seit dem letzten Operator[2] in der ersten Zeile angezeigt. Sobald ein Operator eingegeben wird, wird die Aktuelle erste Zeile in die zweite bewegt und die erste Zeile wird wieder geleert. (Krampe, SimonKrampe/Taschenrechner, 2022)

2.4 Implementation des I2C LCD-Displays

Für die Implementation des Displays benutzte ich die „RGB1602.py“-Library[3] Library des Displayherstellers[4]. Diese Klasse schickt die Bits an das Display und mein selbstgeschriebenes Python Programm „controlDiplay.py“ empfängt die Argumente, mit denen es von meinem Java Programm gestartet wird (DragonLord999, 2018), und führt die „printout()“-Methode der Library aus. (Krampe, SimonKrampe/Taschenrechner, 2022)

3 Printed Circuit Boards

Printed Circuit Boards werden oft auch als Leiterplatte oder Platine bezeichnet und bestehen aus Kupferfolie, sowie FR4, was eine Verbindung von Epoxidharz und Glasfasergewebe ist. (Leiterplatten, Multi-CB, kein Datum) Diese Platinen werden in elektronischen Geräten benutzt um zu verhindern, dass alles mit Kabeln verbunden werden muss, was sehr unübersichtlich wäre. PCBs werden in einigen schritten hergestellt.

[1] Statische Variablen sind über alle Instanzen eines Objektes synchronisiert und statische Methoden können aufgerufen werden ohne vorher ein Objekt der Klasse zu erstellen.

[2] Eine Rechenoperation

[3] https://www.waveshare.com/wiki/File:LCD1602-RGB-Module-demo.zip (shorturl.at/rATX0)

[4] https://www.waveshare.com/wiki/LCD1602_RGB_Module?Amazon (shorturl.at/jyzN1)

Im ersten Schritt dieser Herstellung wird eine Kupferfolie auf eine FR4-Platte laminiert. Darauf werden die Bohrungen für die einzelnen Komponenten gemacht und das Kupfer wird an den Stellen die keine Leiter werden sollen weggeätzt. Dieser Prozess wird „etching“ genannt. Sobald der Prozess abgeschlossen wurde wird das Board auf Fehler überprüft und es wird Lötstopplack aufgetragen, der das PCB vor Staub und Oxidation schützt, sowie das Zusammenlöten von mehreren Leitungen verhindert. Das PCB, welches ich in meinem Projekt verwende, besteht aus zwei Schichten. Dabei ist eine Schicht oben und eine unten. PCBs können auch mit noch mehr Schichten hergestellt werden, wobei der Prozess allerdings deutlich komplizierter wird. Dabei sieht ein Durchschnitt eines PCB etwa aus wie in Abbildung 6 gezeigt.

Bohrungen in PCBs werden oft als PTH[1] gemacht, was in Abbildung 7 dargestellt ist. Das heißt, dass die Kontakte auf der oberen und der unteren Seite verbunden sind. Dies ermöglicht einen einfacheren Anschluss von Komponenten. Bei meinem Projekt habe ich dies ebenfalls genutzt, da ich die Tasten, welche mit Leitungen auf der Oberseite des PCB verbunden sind, von unten verlötet habe.

3.1 Design des PCB

Für das Design meines PCB habe ich das Programm Eagle von Autodesk[2] benutzt, da die kostenlose Version das Design einer Platine mit zwei Schichten und 80cm^2 (10cm * 8cm) bietet. Ein PCB dieser Größe mit zwei Schichten reicht für einen Taschenrechner aus, weil die Komplexität eines Taschenrechners gut mit zwei Schichten zu bewältigen ist. Um die korrekten Pins vom Raspberry zu finden wurde eine Grafik benutzt. (Howard, 2021)

Dabei muss zuerst ein Schaltplan[3] erstellt werden. Am besten startet man damit, alle Komponenten einzufügen und sie ordentlich zu platzieren. Sobald alles eingefügt ist kann dazu übergegangen werden alles zu verbinden. Der Schaltplan für meinen Taschenrechner sieht aus, wie in Abbildung 8 gezeigt.

---

[1] **P**lated **T**hrough **H**ole

[2] https://www.autodesk.de/products/eagle/free-download (shorturl.at/zAK39)

[3] Die grafische Darstellung einer Schaltung

Wenn der Schaltplan ist kann man damit anfangen das Board zu erstellen. Alle Komponenten und Verbindungen, die auf dem Schaltplan gemacht wurden, werden ebenfalls in diesem Arbeitsschritt angezeigt, um den Prozess zu erleichtern.
Mein fertiges Board[1] sieht aus wie Abbildung 9 zeigt, dabei befinden sich die blauen Leitungen auf der Unterseite, und die roten Leitungen auf der Oberseite:

4 Rechenalgorithmus

Der Rechenalgorithmus, welchen ich für meinen Taschenrechner geschrieben habe, ist dazu fähig eine Rechnung mit einer beliebigen Anzahl an Operatoren auszuführen, Multiplikation und Division vor Addition und Subtraktion durchzuführen, sowie mit negativen Zahlen zu rechnen. Die Entwicklung des Algorithmus habe ich in zwei Teile aufgeteilt. Der erste bestand daraus den Algorithmus in der Theorie zu entwickeln und der zweite war die Implementierung in Java. Einen eigenen Algorithmus zu entwerfen, anstatt eine bereits geschriebene Library zu nutzen, hat mir geholfen ein besseres technisches Verständnis zu entwickeln und hat dieses Projekt ebenfalls deutlich persönlicher gemacht.

4.1 Das Grundprinzip des Algorithmus

Der Algorithmus ist in drei wichtige Teile aufgeteilt. Der erste Teil überprüft welche Operatoren die Eingabe enthält und ruft dementsprechend die richtigen Rechenoperationen auf. Wenn die Eingabe Multiplikation und Division enthält, wird zuerst der zweite Teil, die Methode zu der Berechnung von Multiplikation und Division, aufgerufen. Wenn die Eingabe zusätzlich Addition oder Subtraktion enthält, oder nur daraus besteht, wird der dritte Teil, die Berechnung von Addition und Division, aufgerufen.
Der erste und zweite Teil benutzen denselben Algorithmus zum Erkennen der Operatoren und Operanden[2]. Die Eingabe wird durchiteriert, bis ein gesuchter Operator gefunden wurde. Beim ersten Teil sind dies Multiplikation und Division, während es bei dem zweiten Teil Addition und Subtraktion sind. Sobald ein Operator gefunden wurde, werden die Stelle des Operators und die Eingabe an eine Methode übergeben, welche die Operanden definiert. Sobald die Operanden definiert wurden

[1] Bestellt auf https://jlcpcb.com/

[2] Es wird unter „normalen“ umständen mit zwei Operanden und einem Operator eine Berechnung durchgeführt

werden diese zurückgegeben und berechnet. Nach dem Berechnen wird das soeben berechnete Ergebnis durch die berechneten Operatoren und Operanden ausgetauscht, wie Abbildung 10 zeigt, und es wird weiter nach Operatoren gesucht.

Darauf wird das Ergebnis, wenn alle Operatoren berechnet wurden, von der Methode zurückgegeben. Sollte die Eingabe Multiplikation oder Division, sowie Addition oder Subtraktion enthalten hat wird das Ergebnis aus der Multiplikation und Division- Methode an die Addition und Subtraktion- Methode gegeben damit dies ebenfalls ausgerechnet wird.

4.2 Implementation in Java

Die Methode, welche zuerst aufgerufen wird überprüft die Eingabe auf verschiedene Rechenarten und ruft die dementsprechenden Methoden auf. Diese Methode ist in Abbildung 11 dargestellt.

Die Methoden zum Multiplizieren und Dividieren, welche in Abbildung 12 dargestellt ist, sowie zum Addieren und Subtrahieren funktionieren Grundsätzlich nach dem gleichen Prinzip und ähneln sich stark, da sie beide von den Methoden einigen Methoden Gebrauch machen.

Für die Methode zum Addieren und Subtrahieren, die in Abbildung 13 dargestellt ist, müssen allerdings noch einige Befehle geändert werden.

Die Änderungen bestehen daraus, dass abfragen gemacht werden müssen, damit keine Fehler entstehen. Wenn beispielsweise eine negative Zahl an vorderster Stelle steht, wird dies als Operator erkannt. Deshalb muss ebenfalls überprüft werden, dass der Operator nicht an erster Stelle ist. Bei dem Addieren und Subtrahieren muss ebenfalls extra eingefügt werden, dass der zweite Operand negiert wird, falls der erste Operand negativ ist, um das korrekte Ergebnis zu erhalten. Da dieselbe Methode zum Definieren der Operanden benutzt wird, wie bei der Methode für Multiplikation und Division, werden negative Zahlen als erster Operator nicht erkannt, weil dies für Multiplikation und Division nicht wichtig ist.

Wenn jedoch bei Multiplikation und Division der zweite Operand eine negative Zahl ist kommt ebenfalls eine negative Zahl beim Berechnen heraus. Wenn allerdings der

erste Operand ebenfalls negativ ist bleibt das „Minus“ vorne einfach stehen. Das doppelte „Minus“, was nach der Berechnung vorne steht, wird erkannt und durch ein „Plus“ ersetzt, was in Abbildung 14 dargestellt ist.

Die Methode die benutzt wird um die Operanden zu definieren wird von beiden Berechnungsmethoden aufgerufen und ist in Abbildung 15 dargestellt.

Mit dieser Methode werden die „Grenzen“ der Operanden definiert. Dies geschieht indem die Grenze verschoben wird bis der nächste Operator, das Ende oder der Anfang erreicht sind. Sobald der Anfang und das Ende des Operanden erreicht sind wird alles zwischen diesen Grenzen als String gespeichert und zu einem Double[1] konvertiert. Anschließend werden alle wichtigen Werte mit einer Wrapper-Klasse[2] verpackt und zurückgegeben. Hier wird ein Wrapper-Objekt verwendet, da es vereinfacht mehrere Werte zurückzugeben.

Eine ebenfalls wichtige Methode, die von beiden Berechnungs-Methoden aufgerufen wird, ist updateInput(). Sie fügt das soeben berechnete Ergebnis (TemporäresErgebnis) in die Eingabe ein und überprüft sie auf doppelte Operatoren. In Abbildung 10, sowie Abbildung 14 sind Beispiele abgebildet. (Krampe, SimonKrampe/Taschenrechner, 2022)

4.3 Laufzeitanalyse

Die Laufzeitberechnung ist ein wichtiger Faktor um die Effizienz eines Programmes zu testen. Bei meinem Programm habe ich ebenfalls eine Möglichkeit zur Berechnung der Laufzeit entwickelt. Ich werde hier nicht weiter auf die Implementation der Laufzeitberechnung eingehen, da nur die Zeiten des Algorithmus gemessen, sowie gespeichert werden. (W3Schools, kein Datum) Wie die Laufzeit berechnet wird ist nicht weiter wichtig für das Projekt, da der Fokus auf dem Rechenalgorithmus, der Eingabe mithilfe von dem Raspberry Pi Zero und der Ausgabe durch den Raspberry Pi Zero auf einen Bildschirm liegt.

[1] Ein Datentyp, der eine Zahl mit Nachkommastellen speichert

[2] Eine Klasse, die benutzt wird um mehrere Variablen zu „verpacken“ und gebündelt zu speichern oder zu übergeben

Einige Parameter können bei der Laufzeitberechnung eingestellt werden um die einzelnen Bestandteile des Algorithmus zu testen (Siehe Abbildung 16). Der Operand wird dabei immer wieder hinten an den bestehenden String angehängt.

Die von der Laufzeitberechnung ausgegebenen Werte sind in Abbildung 17 graphisch dargestellt. Es lässt sich erkennen, dass der Algorithmus etwa mit einer Laufzeit von $O(n^2)$[1] Berechnungen ausführt. O wird dabei nicht zur Berechnung der tatsächlichen Laufzeit genutzt, sondern soll nur angeben wie der Algorithmus sich auf längere Zeit verhält.

5 Schluss und Ausblick

Einen Taschenrechner selbst zu bauen, sowie zu Programmieren hat mein technisches Verständnis um einiges erweitert. Sehr interessant an diesem Prozess fand ich ein Programm zur algorithmischen Berechnung von Beliebig vielen Zahlen zu entwickeln, sowie eine Platine zu designen. Interessant würde ich ebenfalls noch finden eine bessere Laufzeit zu erreichen, da eine quadratische Laufzeit nicht wünschenswert ist. Mit dem was ich in diesem Projekt in Bezug auf PCBs gelernt habe, möchte ich zukünftig eine eigene Tastatur für den alltäglichen Gebrauch designen. Einen Taschenrechner mit dem Raspberry Pi Zero zu erstellen hat ebenfalls gewisse Schwierigkeiten mit sich gebracht, da die Pins von Raspberrys gewöhnlich per Python ausgelesen und angesteuert werden. Für die Tasten habe ich, wie in Kapitel 2.3 erwähnt, die Pi4J Bibliothek gefunden die das Arbeiten mit GPIOs aus Java ermöglicht. Anfangs wollte ich diese Bibliothek ebenfalls für den Bildschirm benutzen, allerdings sind bei Tests mit einem anderen Bildschirm Probleme aufgetreten, weshalb ich bei dem neuen Bildschirm auf die Python-Bibliothek vom Hersteller zurückgreife. Dies hat mir ebenfalls einen kleinen Einblick in Python verschafft. Da Eingabe, sowie Ausgabe vollständig über das PCB erfolgen und der Raspberry über Batterien laufen kann lässt sich der Taschenrechner auch vollständig kabellos benutzten, was einen Taschenrechner ausmacht, da heute jeder Computer bereits einen eingebauten Taschenrechner hat. Zusammenfassend lässt sich also sagen, dass einen Taschenrechner selbst zu programmieren, sowie zu bauen, einige Schwierigkeiten, speziell Löten, das Arbeiten mit GPIOs und I2C

[1] Ein mithilfe der O-Notation angenäherter Wert wie der Algorithmus sich auf unbestimmte Länge verhält

Bildschirmen hat, aber auch in manchen Aspekten, wie dem Programmieren des Algorithmus, einfacher ist als anfangs erwartet.

6 Literaturverzeichnis (Primärliteratur)

Krampe, S. (15. Februar 2022). *SimonKrampe/Facharbeit-Informatik-Taschenrechner*. (GitHub) Abgerufen am 15. Februar 2022 von SimonKrampe/Facharbeit-Informatik-Taschenrechner: https://github.com/SimonKrampe/Facharbeit-Informatik-Taschenrechner

Krampe, S. (10. Februar 2022). *SimonKrampe/Taschenrechner*. (GitHub) Abgerufen am 10. Februar 2022 von SimonKrampe/Taschenrechner: https://github.com/SimonKrampe/Taschenrechner

Leiterplatten, Multi-CB. (kein Datum). *Flex- und Starrflex Leiterplatten*. Abgerufen am 18. Januar 2022 von https://www.multi-circuit-boards.eu/leiterplatten-design-hilfe/flex-starrflex.html (shorturl.at/vMV36)

Stern, F. (kein Datum). *Raspberry Pi Keypad anschließen – Codeschloss*. Abgerufen am 5. Januar 2022 von https://tutorials-raspberrypi.de/raspberry-pi-keypad-tastatur/ (shorturl.at/myHQW)

Waveshare. (2. April 2021). *LCD1602 RGB Module.* Abgerufen am 20. Januar 2022 von LCD1602 RGB Module: https://www.waveshare.com/wiki/File:LCD1602_RGB_Module.pdf (shorturl.at/lwU79)

7 Literaturverzeichnis (Sekundärliteratur)

DragonLord999. (27. März 2018). *How to execute a python file with few arguments in java*. Abgerufen am 25. Januar 2022 von https://www.edureka.co/community/358/how-to-execute-a-python-file-with-few-arguments-in-java#:~:text=You%20can%20use%20Java%20Runtime,and%20then%20set%20it%20executable (shorturl.at/yCES6)

GeeksforGeeks. (24. November 2020). *String class in Java | Set 1*. Abgerufen am 28. Dezember 2021 von https://www.geeksforgeeks.org/string-class-in-java/ (shorturl.at/zDJTW)

Howard, P. (27. Oktober 2021). *Raspberry Pi GPIO Pinout*. Abgerufen am 5. Januar 2022 von https://pinout.xyz/

W3Schools. (kein Datum). *Java Create and Write To Files*. Abgerufen am 4. Januar 2022 von https://www.w3schools.com/java/java_files_create.asp (shorturl.at/cxGX0)

8 Anhang

Alle Abbildungen wurden von mir mit PowerPoint erstellt, außer anders in einer Grafik angemerkt. Die Grafiken lassen sich in Quelle (Krampe, SimonKrampe/Facharbeit-Informatik-Taschenrechner, 2022) wiederfinden.

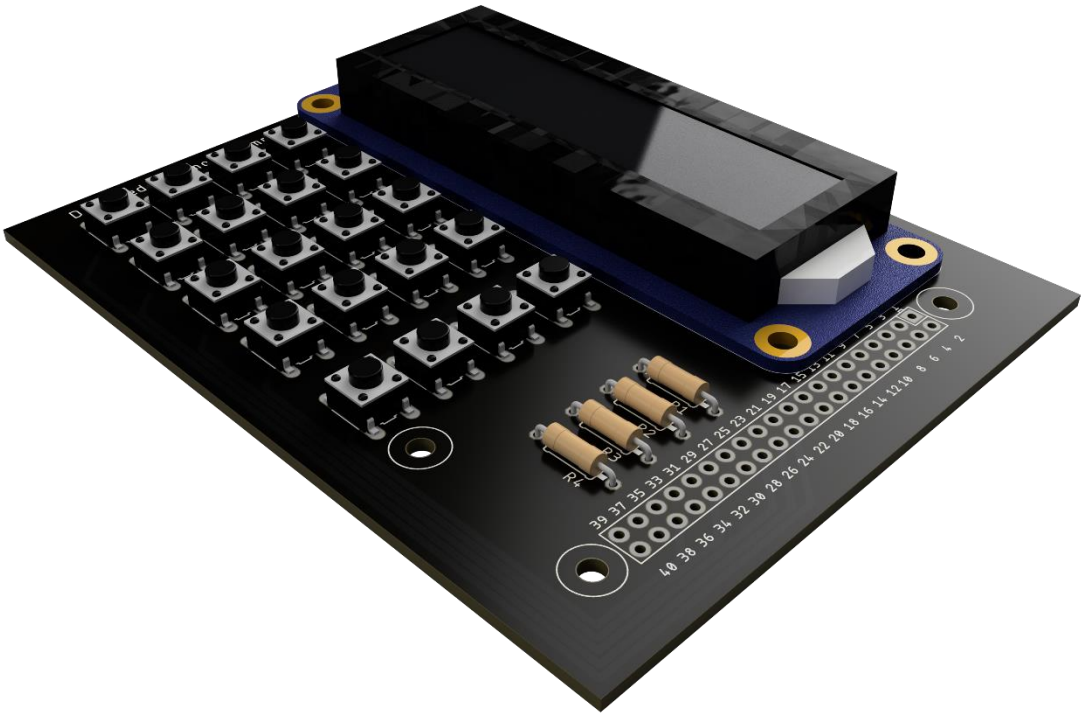

Abbildung 1 *Erstellt mit Eagle*

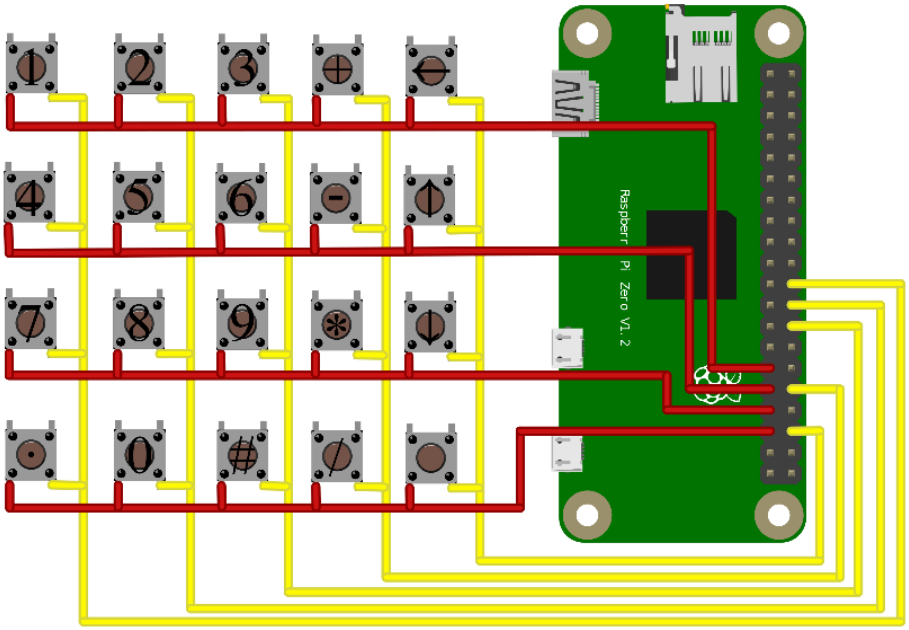

Abbildung 2 *Erstellt mit Fritzing*

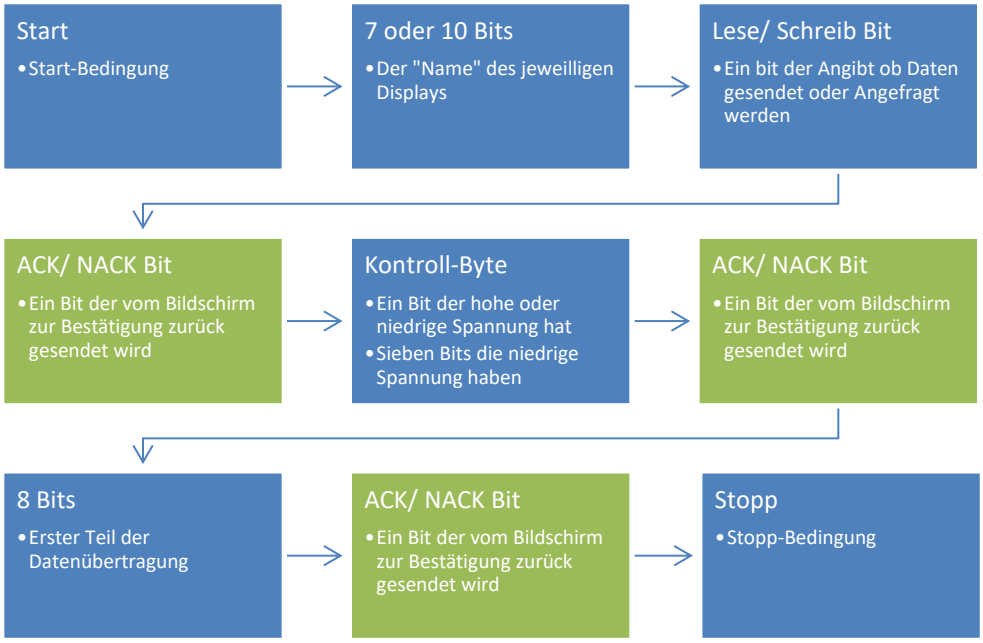

Abbildung 3

```
public class Listener
    public static void main()
        Wiederhole
            Wiederhole für jede Spalte          //i
                Setzte aktuelle Spalte auf niedrig
                Wiederhole für jede Zeile       //j
                    Wenn die aktuelle Zeile niedrig ist
                        Rufe das Zeichen für die aktuelle Zeile und Spalte ab
                        StringComposer.add(Zeichen)
                Setzte aktuelle Spalte auf hoch
```

Abbildung 4

```
public class StringComposer
    public static void add(Zeichen)
        Wenn das Zeichen ‚#‘ entspricht
            Berechne das Ergebnis mit Rechenanfrage
            Zeige das Ergebnis auf dem Bildschirm an
            Setze alle Variablen zurück
            return;
        Rechenanfrage = Rechenanfrage + Zeichen
        zeile1Temporär = zeile1Temporär + Zeichen
        Wenn das Zeichen ‚+‘ oder ‚-‘ oder ‚*‘ oder ‚/‘ entspricht
            zeile2Temporär = zeile1Temporär
            Setze zeile1Tempörär zurück
        Zeige zeile1Temporär und zeile2Temporär auf dem Bildschirm
```

Abbildung 5

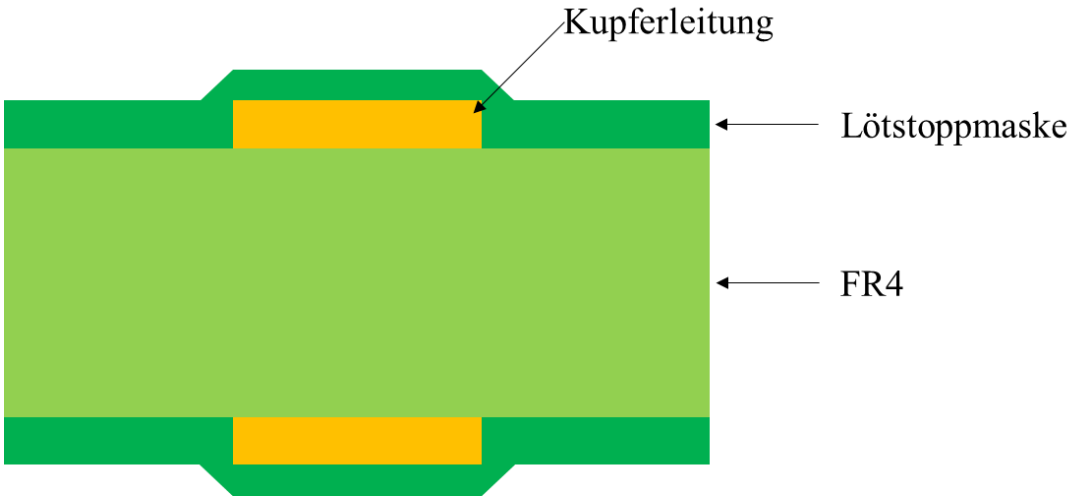

Abbildung 6

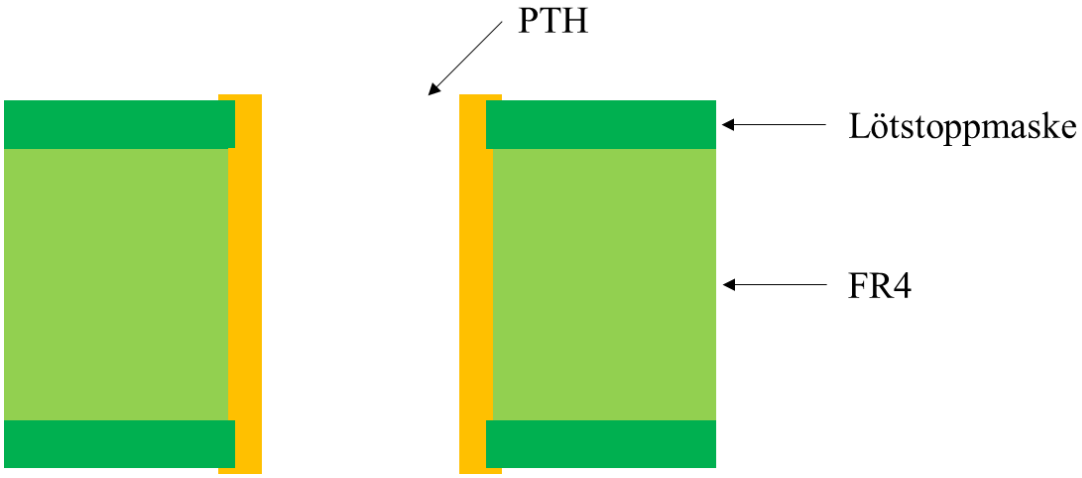

Abbildung 7

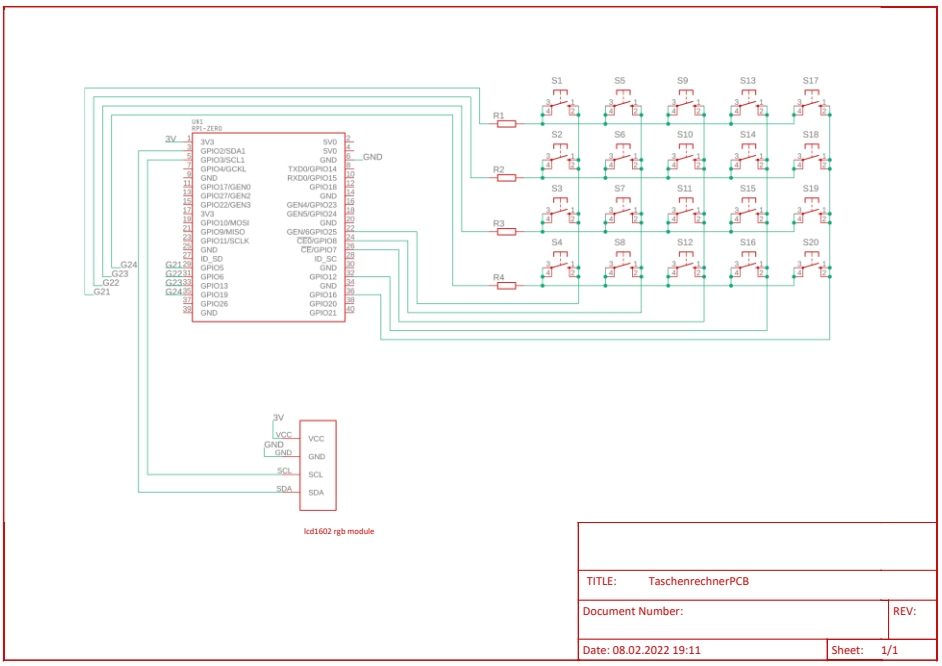

Abbildung 8 *Erstellt mit Eagle*

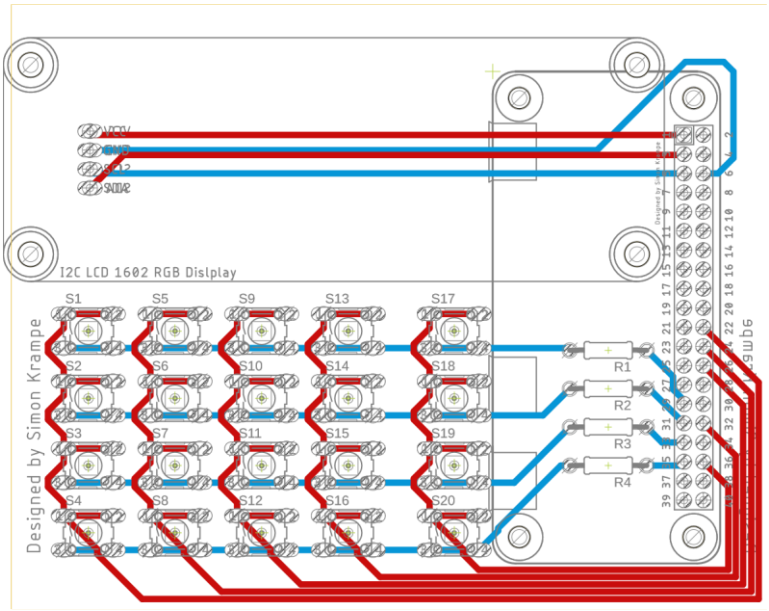

Abbildung 9 *Erstellt mit Eagle*

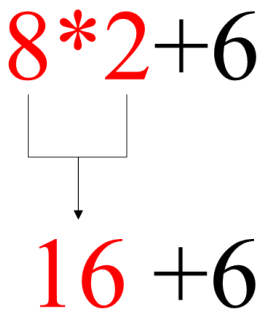

Abbildung 10

```
public class Rechner
    public static String berechnen(Eingabe)
        Wiederhole für jedes Zeichen der Eingabe
            Wenn das Zeichen ‚*‘ oder ‚/‘ entspricht
                Setze hatMultiplikationOderDivision auf „wahr“
            Wenn das Zeichen ‚+‘ oder ‚-‘ entspricht
                Setze hatAdditionOderSubtraktion auf „wahr“
            Wenn beide Variablen „wahr“ sind
                break;
        Wenn beide Variablen „wahr“ sind
            Eingabe = multiplizierenOderDividieren(Eingabe)
            Ergebnis = addierenOderSubtrahieren(Eingabe)
        Sonst wenn nur hatMultiplikationOderDivision „wahr“ ist
            Ergebnis = multiplizierenOderDividieren(Eingabe)
        Sonst wenn nur hatAdditionOderSubtraktion „wahr“ ist
            Ergebnis = addierenOderSubtrahieren(Eingabe)
        return Ergebnis;
```

Abbildung 11

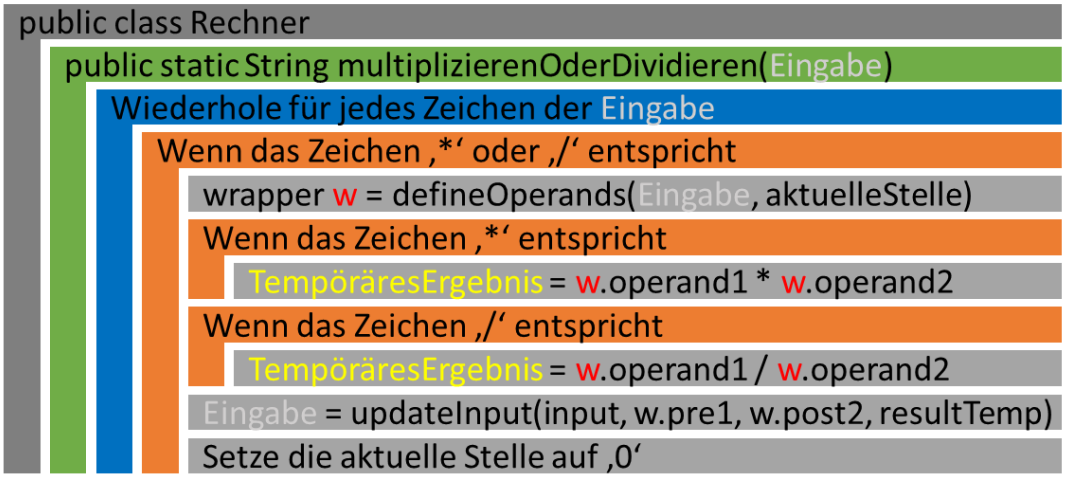

Abbildung 12

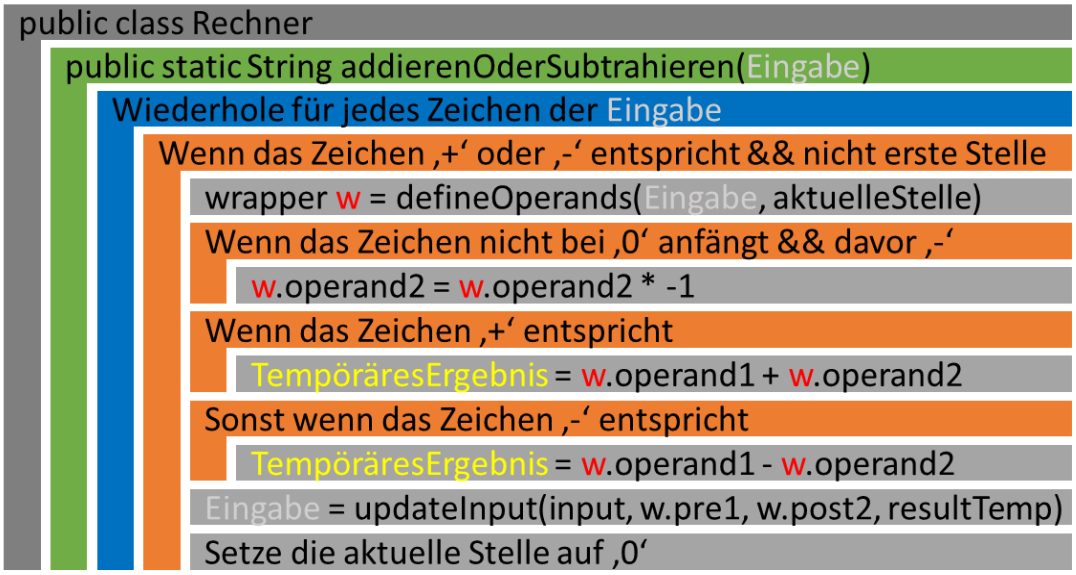

Abbildung 13

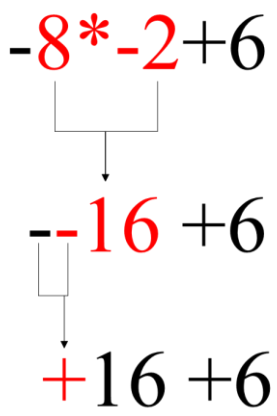

Abbildung 14

```
public class Rechner
    public static wrapper operandenDefinieren(Eingabe, aktuelleStelle)
        String pre1 = aktuelleStelle – 1;
        Wenn pre1 nicht ‚0‘ entspricht
            Solange das Zeichen an der Stelle vor pre1 nicht ‚+‘ oder ‚-‘
                pre1--;
                Wenn pre1 ‚0‘ entspricht
                    break;
        String post1 = aktuelleStelle – 1;
        String post2 = post1;
        Wenn post2 nicht der Länge der Eingabe – 1 entspricht
            Wenn das Zeichen an der Stelle post2 ‚+‘ oder ‚-‘ entspricht
                post2++;
            Solange das Zeichen an der Stelle vor post2 kein Operator ist
                post2++;
                Wenn post2 der Länge der Eingabe – 1 entspricht
                    post2++;
                    break;
        String pre entspricht allem von pre1 bis zu aktuelleStelle
        String post entspricht allem von post1 bis zu post2
        Int operand1 entspricht dem wert von pre
        Int operand2 entspricht dem wert von post
        wrapper w = new wrapper(pre1, post2, operand1, operand2)
        return w;
```

Abbildung 15

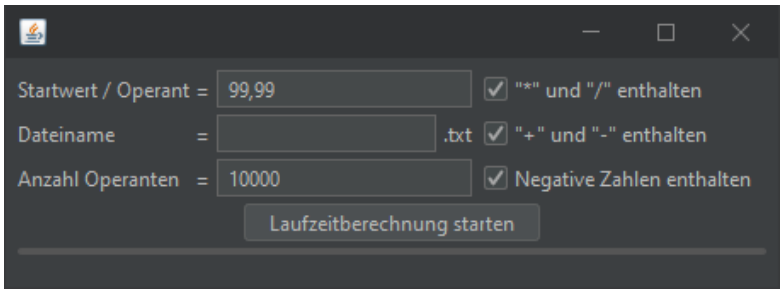

Abbildung 16 *Bildschirmfoto meines Programms*

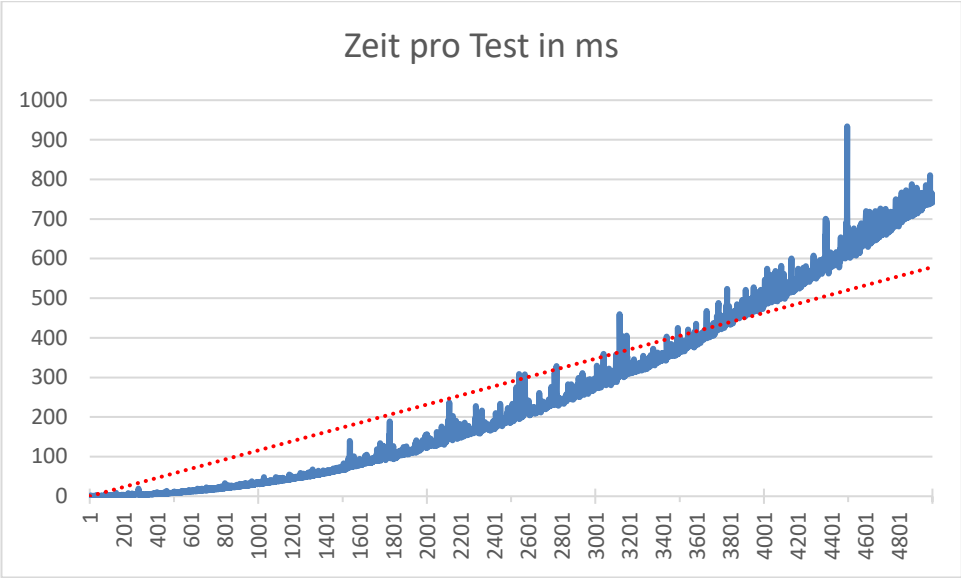

Abbildung 17

9 Erklärung

„Hiermit erkläre ich, dass ich die vorliegende Facharbeit selbständig angefertigt, keine anderen als die angegebenen Hilfsmittel benutzt und die Stellen der Facharbeit, die im Wortlaut oder im wesentlichen Inhalt aus anderen Quellen entnommen wurden, mit genauer Quellenangabe kenntlich gemacht habe. Dies gilt auch für Zeichnungen, Skizzen, bildliche Darstellungen sowie für Quellen aus dem Internet.“

_____ _____

Ort, Datum *Unterschrift*