Juventus Technikerschule HF
www.technikerschule.ch

## Semester Arbeit Embedded-Systems

# Pi Berechnung freeRTOS

**Andreas Sprecher**

**Juventus Technikerschule HF**

**19. April 2022**

# Inhaltsverzeichnis:

**Abbildungsverzeichnis ..... 3**
**Einleitung zur Semesterarbeit Pi Berechnungen ..... 4**
**1 Messbericht freeRTOS Pi Berechnungen ..... 5**
1.1 Kapitel Wahl und Beschreibung der Algorithmen ..... 5
1.2 Kapitel Aufstellung und Umfang ..... 7
1.3 Kapitel, Code, Aufbau und Struktur ..... 8
1.4 Kapitel Diskussion und Konklusion des Codes sowie Ausführung ..... 9
**Persönliches Fazit ..... 11**
**Literaturverzeichnis ..... 12**
**Anhang ..... 13**

# Abbildungsverzeichnis

Abbildung 1 Leibniz Reihe Summenformel ..... 5
Abbildung 2 Euler Formel ..... 6

# Einleitung zur Semesterarbeit Pi Berechnungen

Im Fach Embedded-Systems, welches von Martin Burger betreut wird, soll als Fach Semesterarbeit mittel freeRTOS ein Projekt erstellt werden. Die genaueren Details können der beigelegten Aufgabenstellung entnommen werden. Nachfolgend werden kurz die wichtigsten Eckpunkte erläutert.

Mittels dem bereits ausgegebenen EduBoard soll mittels freeRTOS und dem Programm Microchip Studio©[1] ein Code erstellt werden, welcher zwei unterschiedliche Algorithmen verwendet, mittels denen man die Kreiszahl Pi berechnen kann.

Die beiden Algorithmen müssen mittels  Ein und Ausgabemöglichkeiten des EduBoard dargestellt und gesteuert werden können.

Das bedeutet konkret, mit dem ersten Knopf soll der Algorithmus gestartet werden, mit dem zweiten gestoppt, mit dem dritten zurückgesetzt und mit dem vierten Knopf soll zwischen den Berechnungsmethoden gewechselt werden. Weitere Implementationen mittels der Knöpfe stehen zu freien Wahl.

Die aktuelle Berechnung soll im Intervall von einer halben Sekunden immer angezeigt werden.

Des Weiteren soll der Code eine Implementierung enthalten, mittels derer man die zwei unterschiedlichen Algorithmen zeitlich miteinander vergleichen kann. Das heisst konkret, es soll die Zeit gemessen werden, wie lange es geht bis der eine oder andere Algorithmus die 5 Stelle nach dem Komma berechnet hat und die Zeit anzeigen. Die Berechnung läuft dabei im Hintergrund weiter.

Das Ganze wird abgerundet durch eine korrekte, übersichtliche und nachvollziehbare Versionsverwaltung des Codes unter GitHub.

Der Bericht wurde in deutscher Sprache ausgeführt, etwelche in der Programmier-Umgebung verwendeten Anglizismen werden so belassen und werden nicht eingedeutscht.

Zur Verständlichkeit dieses Berichtes, sowie des erarbeiteten Codes, werden grundlegende Kenntnisse der Materie vorausgesetzt.

[1] Microchip Studio ist eine eingetragene Marke der Microchip Technology Inc.

# 1 Messbericht freeRTOS Pi Berechnungen

## 1.1 Kapitel Wahl und Beschreibung der Algorithmen

Es sollen zwei voneinander unabhängig arbeitende Algorithmen erstellt werden. Die Aufgabenstellung definiert den einen bereits fix, der andere muss frei gewählt werden.

Beide Algorithmen müssen implementiert werden. Nachfolgend nun die Beschreibung der beiden Methoden.

Die Aufgabenstellung definiert als die erste Methode, die Berechnung und nach der Leibniz Reihe[2].

Die Leibniz Reihe ist bekannt nach seinem Erfinder Gottfried Willhelm Leibniz. Leibniz postulierten dies Berechnungsmethode im Jahr 1682 in der Zeitschrift „Acta Eritudorum".

Die Methode war aber bereits zu früheren Zeiten anderen Wissenschaftlern bekannt, Leibniz entdeckte diese sozusagen lediglich neu für das kontinentale Europa.

Die Summenformel der Reihe lautet:

$$\sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \cdots = \frac{\pi}{4}.$$

**Abbildung 1 Leibniz Reihe Summenformel**

Mithilfe dieser Formel konnten bereits im 17.ten und 18.ten Jahrhundert Mathematiker die Pi-Zahl auf bis zu 100 Stellen nach dem Komma genau bestimmen[3].

Die Reihe sieht zwar nobel aus, konvergiert aber beim Berechnen relativ langsam und wird deswegen in der Praxis selten verwendet.

Die Leibniz Reihe wurde im Code mit einem eigenen Task verwirklicht.

[2] Leibniz Reihe, https://de.wikipedia.org/wiki/Leibniz-Reihe
[3] Abraham Sharp 167, 71 Stellen, https://matheguru.com/allgemein/die-kreiszahl-pi.html

Der zweite Algorithmus wurde von Leonhard Euler 1748 in Introductio in Analysin Infinitorum postuliert.[4] Diese basiert auf der ebenfalls von Euler bereits verwendeten Riemannsche Zeta Funktion[5].

Die Formel lautet folgendermassen:

$$\zeta(2) = \frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \cdots = \frac{\pi^2}{6}$$

**Abbildung 2 Euler Formel**

Euler gab in oben genannter Abschrift die Pi-Zahl bereits mit einer Genauigkeit von 148 Nachkommastellen an.

Euler war der eigentliche Entdecker dieser Art der Berechnung, welche auf der Logik der komplexwertigen mathematische Funktion beruht. Diese wird heute dem Teilgebiet der analytischen Zahlentheorie zugeordnet. Euler konnte die Logik dessen noch nicht ganz manifestieren, da ihm gewisse Werkzeuge noch nicht zur Verfügung standen. Erst Bernhard Riemann konnte fast 100 Jahre nach Euler dem Ganzen eine Logik verleihen, deshalb ist das Ganz auch nach Riemann benannt.

Die Euler Reihe bekommt ebenfalls einen eigenen Task

Der vorgängige, Augen merkliche Hauptunterschied der beiden Algorithmen liegt darin, dass bei der Euler Reihe eine Wurzel gezogen werden muss. Inwiefern sich dies dann in der Realität auf die Rechenleistung auswirken wird, wird sich dann in der Praxis zeigen und wird in den nachfolgenden Kapiteln beschrieben.

[4] Euler Funktion, https://www.biancahoegel.de/mathe/zahl/zahl_pi.html
[5] Riemannsche Zeta Funktion, https://www.biancahoegel.de/mathe/fkt/riemann_zeta_fkt.html

## 1.2 Kapitel Aufstellung und Umfang

Für diese Semesterarbeit werden Materialien gebraucht, welche den Studenten in den vorangegangenen Semestern bereits ausgegeben wurde. Dabei handelt es sich um folgende Hard und Software:

- EduBoard Version 10 der Juventus Technikerschulen
- Microchip Studio 7.0, Version 7.0.2542

Das EduBoard wurde bereits im vierten Semester verwendet, das EduBoard basiert auf einem ATXMega 128Ua3u, weitere Details sind der Arbeit beigefügten Dokumentation zu entnehmen.

Microchip Studio ist ein Freeware Programm, mittels derer die gesamte Erstellung, Implementierung und Verwaltung des zu erstellenden Codes erledigt wird.

Für die Ausführung des freeRTOS Programms wird eine Implementierung «FreeRTOS10-Template» vom Dozenten Martin Burger bereitgestellt. Diese Erweiterung wird dem Programm als ZIP Datei eingefügt. Auf Basis dieser Erweiterung wurde ein Projekt erstellt, in welchem der Code erstellt wird. In den Projektoptionen müssen zwingen noch folgende Einstellungen vorgenommen werden, damit double und float im sprintf aktiviert wird.

- General: «Use vprintf library(-WL,-u,vfprintf)»
- Miscellaneous: Other Linker Flags: «-lprintf_flt»

Auf detailspezifische Einstellungen im Microchip Studio wird an dieser Stelle nicht explizit eingegangen, da es sich um normale Einstellungen, respektive Erweiterungen handelt, welche jeweils Projektbezogen vorgenommen werden müssen. Für die Arbeit wichtige Aspekte werden Bezug nehmend erwähnt.

Die Verwaltung des Codes wurde mittels Git[6] über GitHub[7] gemacht. Der Benutzer verfügt dabei über ein eigenes Benutzerkonto bei GitHub und verwaltet das Projekt über diesen Account. Sämtliche für die Semesterarbeit relevanten Dokumente werden abschliessend ebenfalls darüber verwaltet und geteilt. Die URL des Benutzerkontos lautet:

https://github.com/SprecherHF

[6] Git ist ein Freeware Programm, Details im Literaturverzeichnis
[7] GitHub ist eine eingetragene Marke der Microsoft Corporation

Das Projekt firmiert unter Pi_calc

## 1.3 Kapitel, Code, Aufbau und Struktur

Sämtliche nachfolgend beschriebenen Ausführungen im Code beruhen wie eingangs bereits beschrieben auf freeRTOS. Dazu steht Online eine komplette Enzyklopädie zur Verfügung[8].

Der Code wird in der Datei «Main.c» erstellt.

Der Aufbau beinhaltet die üblichen Bibliotheken. Alle spezifischen Berechnungen und Auswertungen werden mittels eines eigenen Tasks versehen. Dabei wurden folgende Tasks erstellt:

- Leibniz Task
- Euler Task
- Compare Task
- Button Task
- Controller Task

Dieses Kapitel stellt die theoretische Überlegung dar, wie der Code ausgeführt werden kann, wie im nachfolgenden Kapitel zu entnehmen, wurden diverse Ideen überarbeitet oder gar verworfen

Der Leibniz und der Euler Task sind für die Berechnungen der jeweiligen Algorithmen zuständig. Die Algorithmen laufen in einer Schleife und wiederholen sich immer wieder.

Der berechnete Wert wird mittels Semaphore / Mutex ausgelesen und im Controller Task ausgewertet und auf der Anzeige angezeigt. Dies geschieht in beiden Tasks unabhängig, da der Definition nach sowieso nur ein Task aktiv geschaltet ist.

Die Pi-Zahl Variable im Leibniz Task lautet «pileibniz»

Die Pi-Zahl Variable im Euler Task lautet «pieuler»

Der Compare Task ist für die Berechnung der Pi-Zahl auf die fünfte Stelle genau verantwortlich. Zudem soll hier ebenfalls via Semaphore / Mutex die vergangene Zeit bis zum Erreichen des Ziels auf der Anzeige angezeigt werden.

Im Button Task wird das Management mit den Knöpfen erstellt, sprich die Ansteuerung des Programms mittels den vier zur Verfügung stehenden Knöpfen. Die Buttons sind dabei fest mit den Eventbits versehen und verknüpft.

[8] freeRTOS, https://www.freertos.org/community/overview.html

Der Controller Task ist für die gesamte Kontrolle sowie für die Darstellung auf der Anzeige verantwortlich. Die Kommunikation, respektive die Ansteuerung der einzelnen Tasks erfolgt über Eventbits.

## 1.4 Kapitel Diskussion und Konklusion des Codes sowie Ausführung

Dieses Kapitel beschreibt im gewissen chronologischen Ablauf die Erstellung des Codes. Wie der Historie im GitHub zu entnehmen ist, unterlag die Entwicklung einem starken Lernprozess, daher musste der Aufbau häufig komplett wieder überarbeitet werden.

Der Dozent Martin Burger stellte ein bereits teilweise aufgesetztes Main.c zur Verfügung. Auf dieser Grundlage ist auch der vorliegende Code entstanden. Anfänglich wurden nur drei Tasks erstellt, zwei mit den entsprechenden Algorithmen und ein Steuertask. Dieser wurde später durch einen Button handleTask erweitert. Der Steuertask (Control Task) soll effektiv lediglich die Steuerung des Codes und die Ansteuerung der Anzeige beinhalten.

Die Berechnung der Algorithmen erfolgt in gesonderten Tasks, dass Printen der Resultate muss Thread-Safe geschehen, sprich, die Abfrage darf keine anderen Abläufe beeinflussen, respektive Stören (Race Condition). Dies wird erreicht, indem das Auslesen und Printen mittels Semaphore / Mutex bearbeitet wird. Da definitionsgemäss jeweils nur ein Task am Laufen ist, wurden nur einmalige Variablen-Namen ausgesucht. Der Ablauf innerhalb der zwei Algorithmen Tasks ist identisch. Dabei wird zuerst die Berechnung der Zahl in einer for Schleife ausgeführt. Anschliessend wird mittels Eventbits abgefragt, ob der Task zurückgesetzt werden soll (Reset). Danach wird via Semaphore die berechnete Variable, in einem Fall «pileibniz» im anderen Fall «pieuler» genannt, der «Übergabe Variable» zugeteilt, diese hat eben genau diesen Namen, der Einfachheit her. Die Übergabe variable wird anschliessen der Printzugeteilt, welche danach auf dem Display geprintet wird.

Im Verlauf der Zeit kam dann nach Überlegung noch den Compare Task hinzu, dieser soll die geforderte Zeit-Berechnungmit dem Erreichen der fünften Stelle nach dem Komma genau managen und regeln. Nach ein paar Überlegungen gelangte ich zum Schluss, dass es wohl besser wäre, den Vergleich direkt in den jeweiligen Berechnung-Tasks zu integrieren. Da es eigentlich kein Sinn ergibt einen einzelnen Task für diese Aufgabe zu kreieren.

Das heisst konkret, es wurde versucht mittels einer weiteren if Bedingung und der Funktion Tick Count und dem Einfügen von zwei weiteren Variablen, der Start und der Stopp Variable. Beide Knöpfe sind mit den Start und Stopp Knöpfen verknüpft. Einen Zähler zu integrieren. Den Wert in Ticks müsste man danach lediglich noch in einen realen Zeitwert umrechnen. Das Berechnen dieses Wertes kann man dann ohne Weiteres im Compare Task umsetzen.

Die Umsetzung der Knöpfe, Button Control Task, sind analog der Anforderung der Aufgabe umgesetzt. Mittels des ersten Knopfes wird ein Berechnungs-Task gestartet, mit dem zweiten gestoppt, mittels des Dritten wird der aktuelle Task zurückgesetzt. Der vierte Knopf dient zum.

Umstellen zwischen den beiden Tasks. Das Umstellen wird mit dem langen Drücken von Knopf drei umgesetzt.

Der Control Task wurde mit einer State Maschine umgesetzt. Diese dient der Steuerung der verschiedenen Möglichkeiten.

Aufgrund laufender Anpassungen seien dies Änderungen im Code, das Hinzufügen von neuen Funktionen, welche anschliessend wieder verworfen wurden, welche aber im GitHub chronologisch abgebildet sind, konnte dato Abschluss dieses Berichtes am Dienstag, 19. April um 16:00, kein funktionstüchtiger Code erstellt werden mittels dem man über die Anzeige auf dem EduBoard arbeiten kann. Dementsprechend ist auch die Compare Funktion nicht funktional und es können entgegen den Vorgaben keine Aussagen über die Zeit, welche die verscheidenden Berechnungs-Tasks in Anspruch nehmen. Auch eine Aussage über die effektive Rechenleistung relativ zur Prozessorleistungbleibt offen.

## Persönliches Fazit

Das persönliche Fazit stellt eine vom Rest der Arbeit abgekoppeltes Resümee dar. Der Autor dankt an dieser Stelle dem Dozenten Martin Burger für seine makellosen Präsentationen, seinen immer verständlich erklärten Ausführungen und Darstellungen der verschiedenen Sachverhalte und seine Geduld!

Diese Semesterarbeit war für mich teilweise sehr starkes «Neuland». Ich konnte diverse neue Einblicke gewinnen und erlernen. Trotzdem hatte und habe ich sehr stark Mühe, die geforderte Abstraktions-Ebene zu erreichen. Mir fehlt nach bisherigen Erkenntnisse schlicht eine jahrelange Basiserfahrung im komplexen Gebiet der Programmiersprachen. Zudem stehen indiskutable Versäumnisse seitens der Schule der vergangenen Semester plus die, für den Autor unzumutbaren fast komplett drei Semester andauernden Fernunterricht, im Weg. Ich gebe zu, ich bin ein Kind aus der analogen Welt und programmieren ist für mich leider eher etwas Notwendiges als etwas Gelebtes.

Ich werde weiter versuchen, die Aufgabe als gesamtes zu finalisieren und daran zu arbeiten. Die Aufgabe, welche wir im Fach von Peter Jost machen dürfen, wird mein Verständnis der Materie sicherlich steigern und positiv beeinflussen.

Ich bedanke mich für den bisherigen Unterricht und ebenfalls für das noch fortlaufende Semester.

Für Juventus Technikerschule

19. April 2022

Andreas Sprecher TS 2020 EL

# Literaturverzeichnis

1. **Diverse.** https://de.wikipedia.org. *https://de.wikipedia.org/wiki/Leibniz-Reihe.* [Online] [Zitat vom: 28. 03 2022.]

2. **Hemmerich, W.A.** https://matheguru.com/. *https://matheguru.com/allgemein/die-kreiszahl-pi.html.* [Online] [Zitat vom: 28. 03 2022.]

3. **Diverse.** https://www.biancahoegel.de. *https://www.biancahoegel.de/mathe/zahl/zahl_pi.html.* [Online] [Zitat vom: 28. 03 2022.]

4. **Diverse**. https://www.biancahoegel.de. *https://www.biancahoegel.de/mathe/fkt/riemann_zeta_fkt.html.* [Online] [Zitat vom: 28. 03 2022.]

5. **Diverse**. https://de.wikipedia.org. *https://de.wikipedia.org/wiki/Git.* [Online] [Zitat vom: 07. 04 2022.]

6. **Diverse**. https://www.freertos.org/community/overview.html. *https://www.freertos.org/community/overview.html.* [Online] [Zitat vom: 07. 04 2022.]

# Anhang

- Aufgabenstellung Semesterarbeit Embedded Systems
- Datenblatt EduBoard V10 als Zip im GitHub