Pi Calculator

Embeded System Project

Christian Nydegger

2023

Contents

[1. Aufgabenstellung 2](#_Toc149594703)

[1.1. Aufgabe 2](#_Toc149594704)

[1.2. Rahmenbedingungen 2](#_Toc149594705)

[2. Berechnungsmethoden 3](#_Toc149594706)

[2.1. Leibniz 3](#_Toc149594707)

[2.2. Vieta 3](#_Toc149594708)

[3. Code 4](#_Toc149594709)

[3.1. State Event Diagram 4](#_Toc149594710)

[3.2. Tasks 5](#_Toc149594711)

[3.2.1. Time Measurement Task 5](#_Toc149594712)

[3.2.2. Leibniz Task 5](#_Toc149594713)

[3.2.3. Vieta Task 6](#_Toc149594714)

[3.2.4. Compare Task 6](#_Toc149594715)

[3.2.5. Display Task 7](#_Toc149594716)

[3.2.6. Control Task 8](#_Toc149594717)

[4. Fazit 9](#_Toc149594718)

[4.1. Geschwindigkeit der Algorithmen 9](#_Toc149594719)

[4.2. Rechenleistung 9](#_Toc149594720)

# Aufgabenstellung

Die folgende Aufgabenstellung ist 1:1 aus dem erhaltenen Auftrag übernommen

## Aufgabe

* Realisiere die Leibniz-Reihen-Berechnung in einem Task
* Wähle einen weiteren Algorithmus aus dem Internet
* Realisiere den Algorithmus in einem weiteren Task.
* Schreibe einen Steuertask, der die zwei erstellten Tasks kontrolliert.

## Rahmenbedingungen

* Der aktuelle Wert soll stets gezeigt werden. Update alle 500ms
* Der Algorithmus wird mit einem Tastendruck gestartet und mit einem anderen Tastendruck gestoppt.
* Mit einer dritten Taste kann der Algorithmus zurückgesetzt werden.
* Mit der vierten Taste kann der Algorithmus umgestellt werden. (zwischen Leibniz und dem zweiten Algorithmus)
* Die Kommunikation zwischen den Tasks kann entweder mit EventBits oder über TaskNotifications stattfinden.
* Folgende Event-Bits könnte man beispielsweise verwenden:
  + EventBit zum Starten des Algorithmus
  + EventBit zum Stoppen des Algorithmus
  + EventBit zum Zurücksetzen des Algorithmus
  + EventBit für den Zustand des Kalkulationstask als Mitteilung für den Anzeige-Task
* Mindestens drei Tasks müssen existieren.
* Interface-Task für Buttonhandling und Display-Beschreiben
* Kalkulations-Task für Berechnung von PI mit Leibniz Reihe
* Kalkulations-Task für Berechnung von PI mit anderer Methode
* Erweitere das Programm mit einer Zeitmess-Funktion (verwende
* xTaskGetTickCount) und messe die Zeit, bis PI auf 5 Stellen hinter dem Komma stimmt. (Zeit auf dem Display mitlaufen lassen und beim Erreichen der Genauigkeit die Zeit berechnen. Die Berechnung von PI soll weitergehen.)[1]

# Berechnungsmethoden

## Leibniz

Die Leibniz Reihe, auch Madhava-Leibniz Reihe genannt, ist eine Methode zur Annäherung Pis.

A number and plus one on a white background

Description automatically generated with medium confidence

Dazu werden von allen Ungeraden Zahlen der Kehrwert genommen, und abwechselnd von 1 abgezogen, beziehungsweise dazugezählt. Die Leibniz. Dadurch entsteht langsam, aber stetig, Pi-Viertel. Eine Annäherung an Pi auf 5 Stellen genau, erfolgt nach ungefähr einer Million Iterationen.

## Vieta

Die Vieta Methode ist eine der ältesten Varianten, um Pi anzunähern. Die Formel ist eine abgeleitete Version der Eulerschen Pi Formel, Die Mathematik wie die nun aber zusammenhängt, überlasse ich anderen[2].   
Nachfolgend sind die benutzten Formeln, es handelt sich um ein auf Multiplizieren immer weiter verschachtelten wurzeln von 2. Dies ist mathematisch aufwändig, in Code Form aber glücklicherweise eher simpel umsetzbar.

A mathematical equation with numbers and symbols

Description automatically generated

Das Produkt aller ai/2 für alle Integer ergibt 2 über Pi

A math equations with numbers and symbols

Description automatically generated with medium confidence

In einer ersten Iteration wird einfach Wurzel 2 verwendet. In jeder weiteren Iteration wird der gesamte Wurzelausdruck der letzten Iteration dazugezählt. Dadurch entsteht ein immer weiter «verwurzelter» Ausdruck

Quelle der Abbildungen: Wikipedia

# Code

## State Event Diagram

A drawing of a person with text

Description automatically generated with medium confidence

## Tasks

## Time Measurement Task

A computer screen with colorful text

Description automatically generated

Dieser Task ist zuständig, für das Messen der Zeit, sowie dem Reset State. Hier wird die Momentane Zeit ausgelesen. Im Falle eines Reset, wird zuerst gewartet, bis die beiden Pi Tasks melden, dass sie zurückgesetzt sind. Danach wird der Timer zurückgesetzt sowie alle Eventgroupbits.

## Leibniz Task

A computer screen shot of a program code

Description automatically generated

In diesem Task, wird mithilfe der Leibniz Reihe, Pi berechnet. Wenn der Programm State RunLeibniz ist, wird in einer Zeile die Annäherung berechnet. Die Formel wird genauer im Kapitel 2.1. erläutert. Danach wird das Vorzeichen Bit getoggled, da dieses bei jeder Iteration wechseln muss. Zuletzt wird die Iteration hochgezählt, da diese in der Berechnung verwendet wird.

Im Falle eines Reset, werden alle Variablen zurückgesetzt und am Schluss das EventGroupBit gesetzt, welches mitteilt, dass der Reset durchgeführt wurde.

## Vieta Task

A screen shot of a computer program

Description automatically generated

In gleicher Manier wie im Leibniz Task, wird hier in einer anderen Form Pi berechnet. Der Reset Prozess verläuft gleich wie im Leibniz Task. Da die Vieta Methode mit Wurzeln funktioniert, wird hier die Funktion «sqrt()» aus der Math.h library verwendet. Diese gibt einfach einen Float Wert zurück, welche der Wurzel der mitgegebenen Zahl entspricht.

## Compare Task

A computer screen shot of a program code

Description automatically generated

Um herauszufinden, ob der momentan laufende Algorithmus Pi schon auf 5 Kommastellen genau erreicht hat, wird in diesem Task, der Berechnete Wert mit einer im Voraus definierten Referenz verglichen. Dazu werden die Double Variabeln, einfach um 10e5 erweitert und danach in einen integer gewandelt. Durch das werden alle noch folgenden Kommastellen einfach ausser Acht gelassen. Danach können die durch das gerundete Zahlen miteinander verglichen werden. Falls Pi genügend genau erreicht wurde, wird der Timer gestoppt.

## Display Task

A screen shot of a computer program

Description automatically generated

Das Ganze soll auch auf dem Display dargestellt werden. Diesen Job übernimmt der Display Task. Dieser füllt erst, je nach momentan aktiver Methode, Text und den momentan erreichten Pi Wert in 2 Char Arrays. Zudem wird die Zeit hier noch aufbereitet, um dann auch leserlich dargestellt werden zu können. Dazu wird zuerst die Zeitdifferenz in Ticks berechnet. Diese Differenz wird dann mithilfe von Modulo auf die richtige Ebene gebracht. Das Referenz Pi, sowie die verstrichene Zeit werden dann in 2 weitere Char Arrays geschrieben. Schlussendlich werden alle 4 zuvor aufbereiteten Strings aufs Display ausgegeben.

# Control Task

A computer screen shot of a program code

Description automatically generatedA screen shot of a computer program

Description automatically generated

Der Control Task dient nur dazu, zwischen den verschiedenen Programm States zu wechseln. Je nach Knopfdruck und vorhergehendem State, wird zum jeweils gewünschten State gewechselt. Zudem wird hier das Reset Bit gesetzt, und in spezifischen Situationen der Timer auf 0 gesetzt. Alle Funktionen von diesem Task, sind im State-Event Diagramm zu finden.

# Fazit

## Geschwindigkeit der Algorithmen

Leibniz: 2min 30s

Vieta: nicht messbar

Dies ist einfach erklärbar, da die Leibniz Reihe ungefähr 1000000 Iterationen benötigt, um auf die gewünschte Genauigkeit zu kommen. Vieta benötigt für dasselbe Resultat gerade einmal 11!

## Rechenleistung

Ich denke, dass beide Reihen trotz unterschiedlicher Lauf Dauer, eher Rechenaufwändig sein können. Bei der Leibniz Reihe steckt der Grossteil der Rechenleistung darin, dass immer grösser werdende umkehrwerte ermittelt werden müssen. Dies bedeutet, dass das Programm Ständig mit Kommazahlen rechnen muss.

Bei der Vieta Variante wird dies noch um Faktoren schlimmer, dort müssen nämlich anstelle von Divisionen, ständig Wurzeln gezogen werden. Dies ist aufwändig, da sich das Wurzel ziehen nicht gerade simpel erweist.   
Im Grossen und Ganzen sind aber beide Varianten nur im Bereich unserer Arbeit aufwändig. Wenn wir betrachten, was wir heute an Rechenleistungen haben, und bedenken, dass beide Methoden auf unserer eher langsamen Hardware funktionieren.

# Quellen

[1] Aufgabenstellung «ES\_U\_CalculatingPI\_V5»

[2] Variorum de Rebus Mathematics Reponsorum Liber VII