Calculating Pi

Let’s load our CPU

Philipp Frase

2019

Inhaltsverzeichnis

[Abbildungsverzeichnis 1](#_Toc6135907)

[Revisionshistorie 1](#_Toc6135908)

[Quellenverzeichnis 1](#_Toc6135909)

[Aufgabenstellung 2](#_Toc6135910)

[Ausführung 4](#_Toc6135911)

[Algorithmus 4](#_Toc6135912)

[vInterface 4](#_Toc6135913)

[vButtonHandler 4](#_Toc6135914)

[vCalculation 4](#_Toc6135915)

[Codeverwaltung mit Github 5](#_Toc6135916)

[Technisches Fazit 6](#_Toc6135917)

[Persönliche Reflexion 6](#_Toc6135918)

# Abbildungsverzeichnis

[Abbildung 1 Leibniz-Reihe 4](file:///C:\Users\frase\Google%20Drive\Juventus_Technikerschule\5.Semester_ED\Embedded_Systems\Übungen\Calculating_Pi_Bericht.docx#_Toc6142717)

[Abbildung 2 Github Logo 5](file:///C:\Users\frase\Google%20Drive\Juventus_Technikerschule\5.Semester_ED\Embedded_Systems\Übungen\Calculating_Pi_Bericht.docx#_Toc6142718)

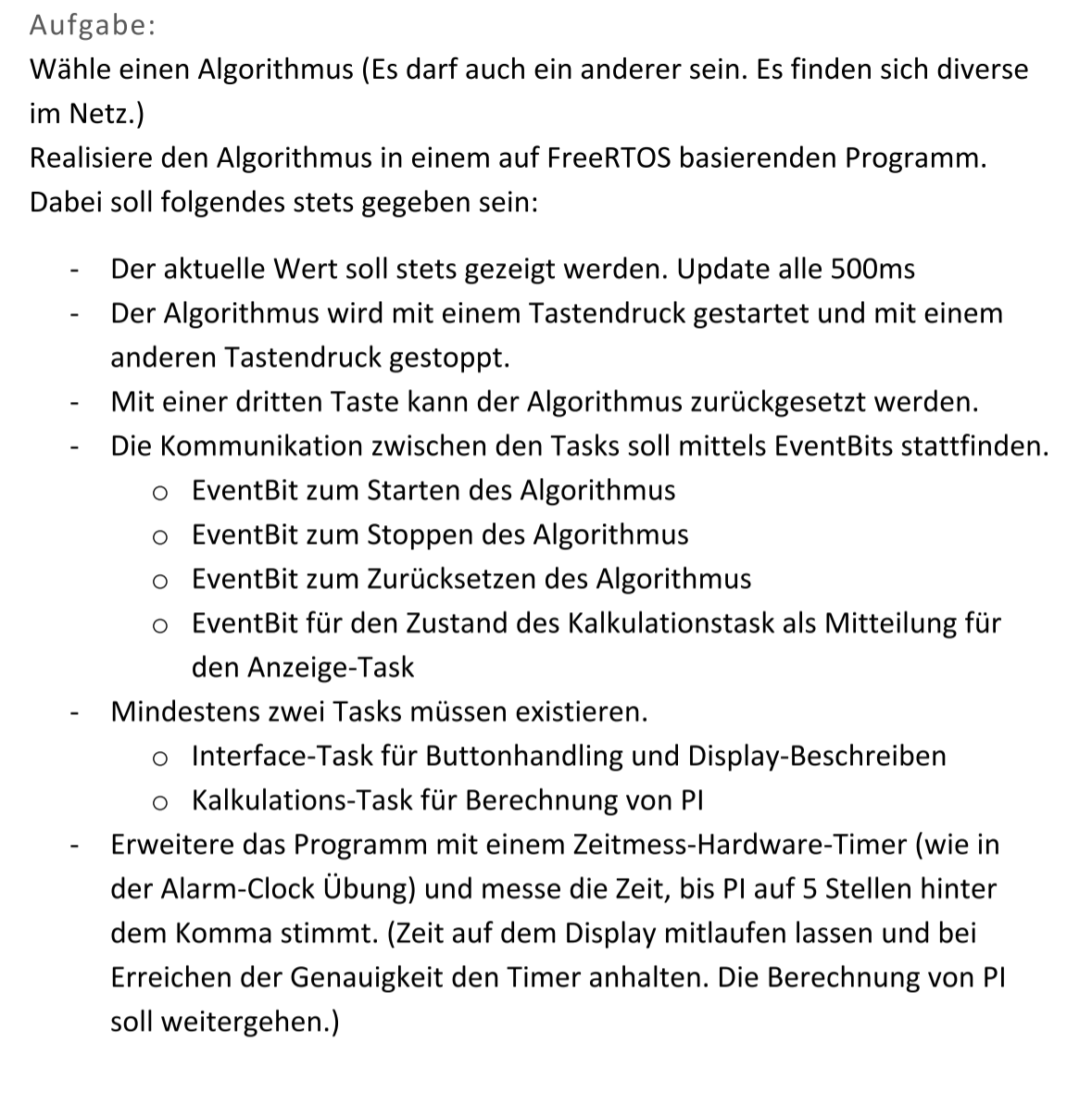
# Revisionshistorie

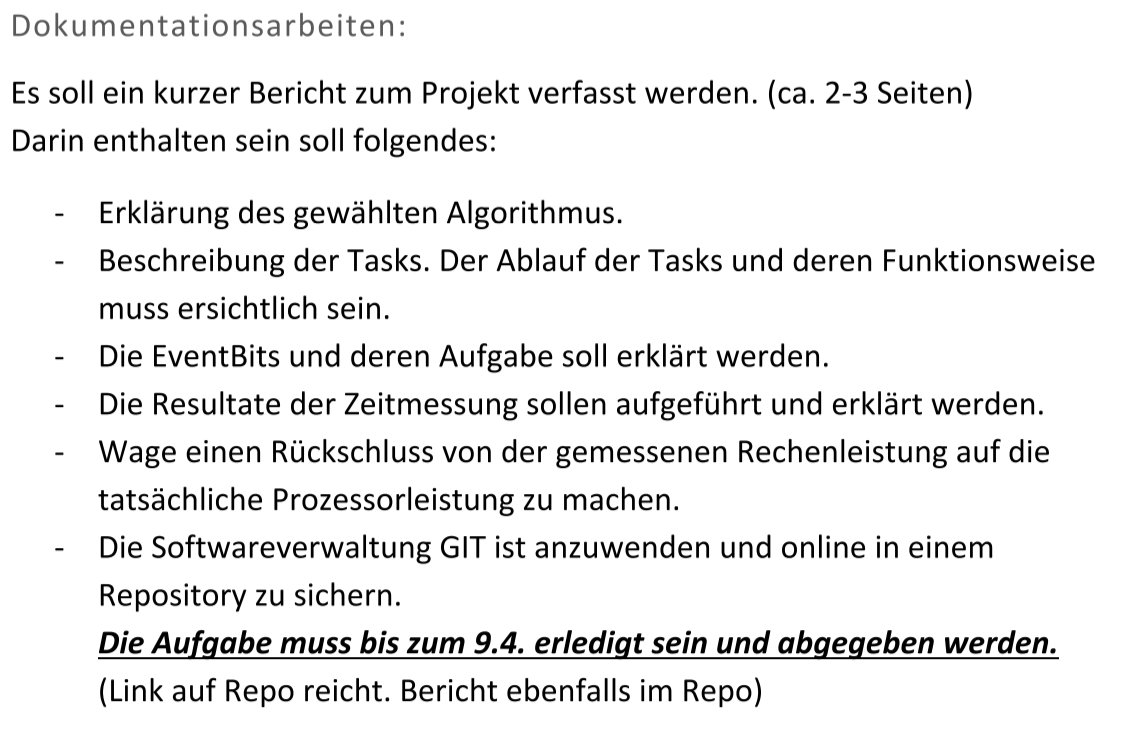
|  |  |  |  |
| --- | --- | --- | --- |
| Rev. nr. | Datum | Änderung durch | Kommentar |
| 1.0 | 14.04.2019 | Philipp Frase | Erstausgabe |
|  |  |  |  |

# Quellenverzeichnis

* Power Point Präsentationen, Martin Burger
* [www.freertos.org](http://www.freertos.org)
* AVR-XMEGA-Microcontroller, Günter Spanner, ISBN: 978-3-89576-288-8

# Aufgabenstellung





# Ausführung

## Algorithmus

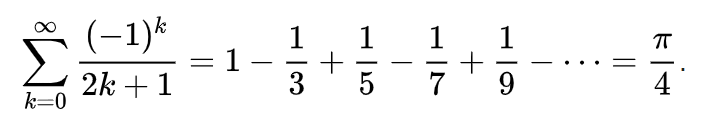
Für die Berechnung von Pi wird dir Leibniz-Reihe (Abb. 1) gewählt.

Abbildung Leibniz-Reihe

Umso öfter dieser Algorithmus berechnet wird, umso näher erreicht man Pi/4.  
Um anschliessend Pi auf dem Display zu sehen, muss man Pi/4 noch mit 4 multiplizieren, dies geschieht in der „printf“ Funktion.

## vInterface

In diesem Task wird die ganze Display Anzeige ausgeführt. Das Display zeigt die aktuelle Version der Firmware, welche gerade auf dem Controller läuft, die Anzahl Iterationen und der momentan berechnete Wert für Pi. Das Display wird alle 500ms aktualisiert.

## vButtonHandler

Der Button Handler Task sorgt dafür, dass jeder Button weiss, was er zu tun hat.  
In dieser Anwendung sind die Buttons wie folgt programmiert:

* Button 1: Startet die Berechnung
* Button 2: Hält die Berechnung an
* Button 3: Setzt die Berechnung fort
* Button 4: Setzt die Berechnung zurück

## vCalculation

Im Calculation Task wird der Algorithmus berechnet. Die Berechnungen finden in einer for-Schleife statt. Gesteurt wird diese Schleife mit der Laufvariable «i». Durch festlegen des Endwertes in der Durchlaufbedingung, kann man festlegen, wie oft die Berechnung durchgeführt werden soll. In diesem Fall sind die Durchläufe auf 10 Mio. mal limitiert.

# Codeverwaltung mit Github

Abbildung Github Logo

Um eine saubere Versionskontrolle zu haben, wurde zur Codeverwaltung die Plattform Github verwendet (Abb. 2).

Sobald ein neues Feature erfolgreich programmiert und getestet wird, „committed“ und „pusht“ man sein Projekt und hat somit seine Daten sowohl lokal wie auch auf den Github Servern.

Dies macht es sehr übersichtlich und man kann zu jeder Zeit nachvollziehen, was, wann und mit welchem Update etwas geändert wurde.

Ein weiterer Vorteil ist, wie vorhin angesprochen, dass die Daten auf den Github Servern gespeichert sind. Somit hat man im Falle eines Datenverlustes ein Backup.

Folgender Link führt zum Github Repositorie des Projekts:

* <https://github.com/pfra17/Calculating_PI>

# Technisches Fazit

Leider wurde ich nicht ganz fertig mit dem Programm, sodass sich doch noch einige Bugs finden.  
Zum Beispiel das zurücksetzen auf null und erneute Starten funktioniert nicht ganz wie ich es mir vorgestellt hatte.

Um mehr Übersicht zu erhalten, entschied ich mich während des Projekts dazu einen dritten, den Button Handler Task, einzuführen. Dies führte, wie von unserem Dozenten vorhergesagt, zu sichtbaren Performance Verlusten.   
Die Software schafft nun circa 324'000 Berechnungen pro Minute. Wobei es vor Einführung des dritten Tasks ungefähr 400'000 Berechnungen pro Minute waren. Das sind rund 76'000 Berechnungen weniger, was meines Erachtens schon erheblich langsamer ist.

Leider kam ich auch nicht mehr dazu den Timer zu programmieren, da ich zu beschäftigt war, die anderen Probleme in den Griff zu kriegen.

# Persönliche Reflexion

Das Projekt hat mir persönlich sehr viel spass gemacht. Ich konnte meine noch nicht so starken Programmierkenntnisse wieder etwas verbessern und habe einiges über das Thema FreeRTOS gelernt.

Ich kann mir gut vorstellen, dass man solch eine Arbeit für kommende Semester vielleicht sogar als Semesterarbeit auslegen kann.