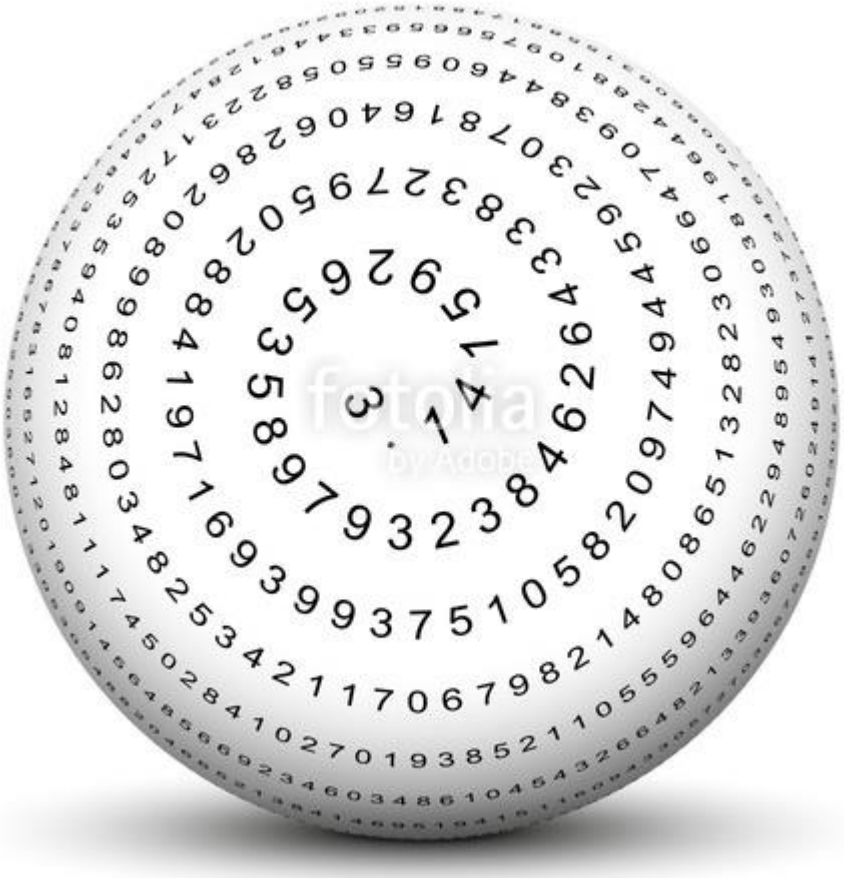
Pi berechnen



Eine Programmierarbeit von:

Cédric Häuptli

Eingereicht am 16.04.2019 bei:

Martin Burger
Juventus Schule
Lagerstrasse 102
8021 Zürich

Inhaltsverzeichnis

1 Einleitung

Das Ziel dieser Arbeit ist es Pi mithilfe eines Algorithmus zu berechnen. Dies sollte möglichst schnell auf 5 Stellen nach dem Koma genau sein, bis dies Erreicht ist, soll die Zeit gemessen werden.

2 Algorithmus

Ich habe mich dafür entschieden, zwei verschiedene Algorithmen zu verwenden. Da es mich Interessiert, welcher dieser Algorithmen schneller ist und wie viel schneller.

2.1 Leibniz-Reihe

Ich habe die Leibniz-Reihe als Vergleichswert gewählt. Da diese sehr einfach aufgebaut ist, kann diese ohne grossen Aufwand realisiert werden. Der Nachteil von der Leibniz-Reihe ist, dass sie viele Iterationen benötigt, bis sie genau wird.

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

Abbildung 1: Formel Leibniz-Reihe

2.2 Gauss-Legendre

Um Pi möglichst schnell zu berechnen, habe ich mich für den Gauss-Legendre Algorithmus entschieden. Dieser ist nicht schwer zu realisieren. Er wird in nur sehr wenigen Iterationen sehr genau. Bereits bei 25 Iterationen wird Pi auf 45 Millionen Stellen genau. Die Nachkomastellen werden exponentiell genauer, bei der ersten Iteration ist es noch sehr ungenau jedoch bereits bei der zweiten ist es auf 8 Stellen genau.

$$3.140\ldots$$
$$3.14159264\ldots$$
$$3.1415926535897932382\ldots$$

Abbildung 2: Die ersten drei Iterationen von Gauss-Legendre

$$a_0 = 1,\ b_0 = \frac{1}{\sqrt{2}},\ s_0 = \frac{1}{4}$$

$$a_{n+1} = \frac{a_n + b_n}{2}$$
$$b_{n+1} = \sqrt{a_n b_n}$$
$$s_{n+1} = s_n - 2^n(a_n - a_{n+1})^2$$

$$\text{Dann gilt } \lim_{n\to\infty} \frac{a_{n+1}^2}{s_n} = \pi.$$

Abbildung 3: Formel Gauss-Legendre

3 Tasks

Da das wechseln von Tasks bereits Ressourcen benötigt, habe ich mich entschieden möglichst wenige Tasks zu verwenden, so dass es trotzdem übersichtlich bleibt.

3.1 Algorithmus Task

Der Algorithmus Task ist nur dafür zuständig, den Algorithmus zu berechnen. Es gibt drei Algorithmus Tasks, es wird dennoch jeweils immer nur einer von ihnen aktiv sein, die Anderen werden suspendiert. Zwei davon benutzten den gleichen Algorithmus, einer von diesen beiden benützt die Float 64 bit Library. Dieser Task ist dazu da, das Pi auf mehr Nachkomastellen dargestellt wird. Dies verursacht dass der Task verlangsamt wird, welches ich messen möchte.

3.2 Ausgabe Task

Der Ausgabe Task ist zuständig für die Ein- und Ausgabe. Er wird ca. alle 10ms aufgerufen, um die Tasten einzulesen. Die Ausgabe am Display wird durch eine Timervariable eingestellt, durch diese updatet sich das Display ca. alle 500ms. Für die Übersicht habe ich ein Struktogramm erstellt. Dieses Struktogramm dient bloss als Übersicht und ist nicht eine 1:1 Kopie vom Code.

Den Reset habe ich nicht über die Eventbits gelöst, aus dem einfachen Grund, dass sich im Algorithmus Task nur die Berechnung befindet und nicht noch eine weitere Abfrage, ob die Variablen gelöscht werden sollten oder nicht. Damit keine

Korrupten Daten entstehen, wird der Ausgabe Task, sobald ein Reset ausgelöst wird, den Algorithmus Task pausieren. Der Ausgabe Task wartet dann 10ms ab, damit der Algorithmus Task alle Berechnungen abschliessen kann und erst danach werden die Variablen gelöscht bzw. auf Anfangswerte zurückgesetzt.

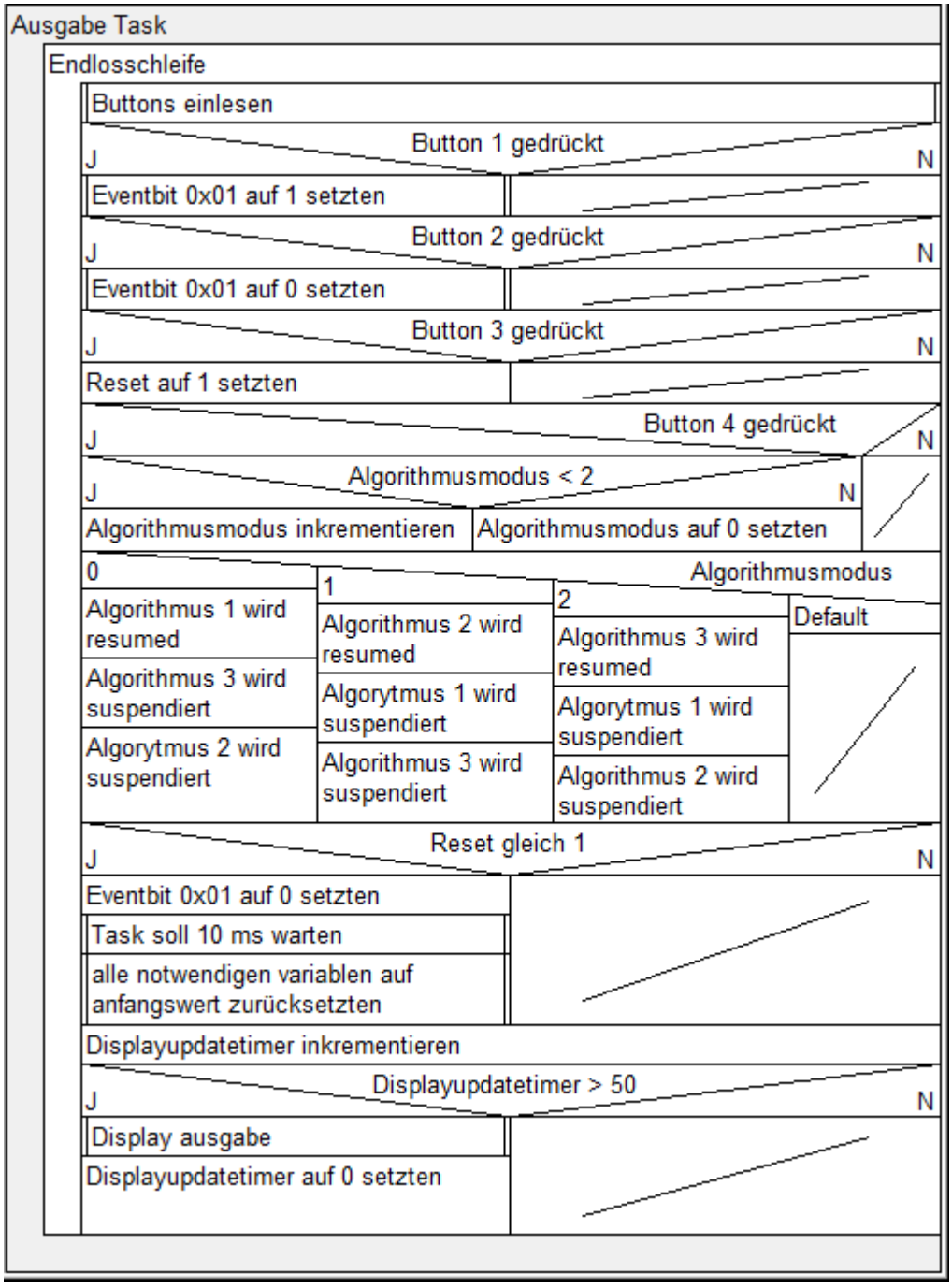


Abbildung 4: Struktogramm Ausgabe Task

4 Event Bits

Ich habe nur einen Event Bit benutzt. Dieser ist nur dafür zuständig, dass der Algorithmus rechnet oder pausiert ist.

5 Zeitmessung

Der Leibniz Algorithmus braucht 10,5 Sekunden, bis er auf 5 Stellen nach dem Koma genau ist, dies sind ca. 130‘000 Iterationen. Der Gauss-Legendre Algorithmus braucht weniger als 1ms, bis er auf 5 Stellen nach dem Koma genau ist. Dies sind zwei Iterationen. Da meine Zeitmessung eine Toleranz von +- 1ms hat, kann nicht die exakte Zeit gemessen werden. Beim Gauss-Legendre Algorithmus, mit der Float 64 bit Library, braucht es für zwei Iterationen bereits 16ms.

6 Abbildungsverzeichniss