Calculating π,

mit FreeRTOS und EduBoard

im Studiengang
Elektrotechnik HF

vorgelegt von

Philipp Eppler

am 14. April 2019
an der höheren Fachschule Zürich

Prüfer: Martin Burger

Kurzfassung

Gegenstand der hier vorgestellten Arbeit ist die Berechnung von PI mittels FreeRTOS und EduBoard. Der Algorithmus soll dabei frei ausgewählt und der laufend angenäherte Wert von PI auf dem LCD ausgegeben werden.

Die Aufgabenstellung ist im Dokument ES_U_CalculatingPI.pdf zu finden.

Inhaltsverzeichnis

1 Algorithmus

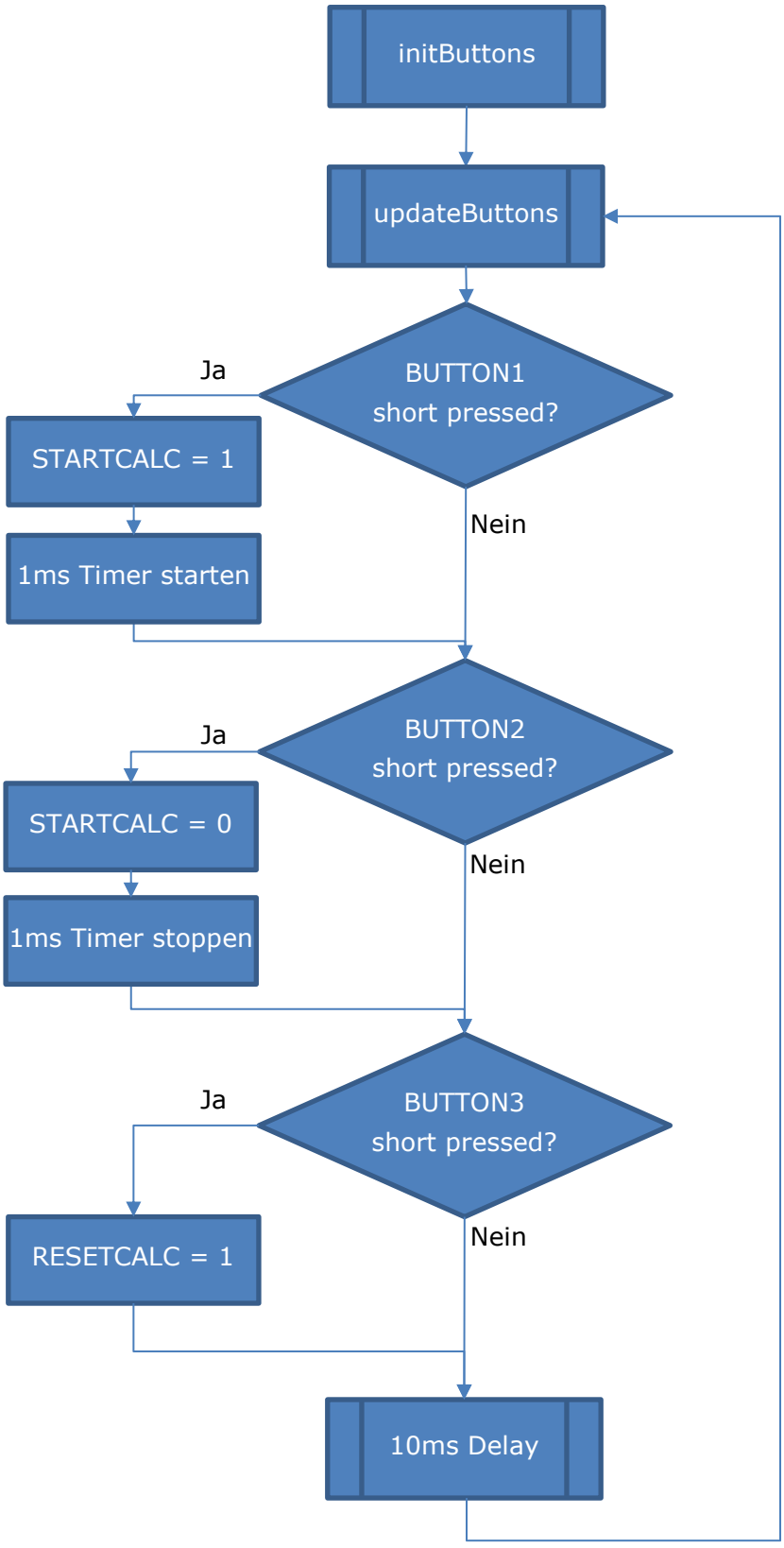
Für die Berechnung von PI wurde die Leibniz-Reihe eingesetzt. Diese ist zwar nicht sehr effizient, aber er ist sehr leicht einzusetzen und benötigt wenig C-Code.

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

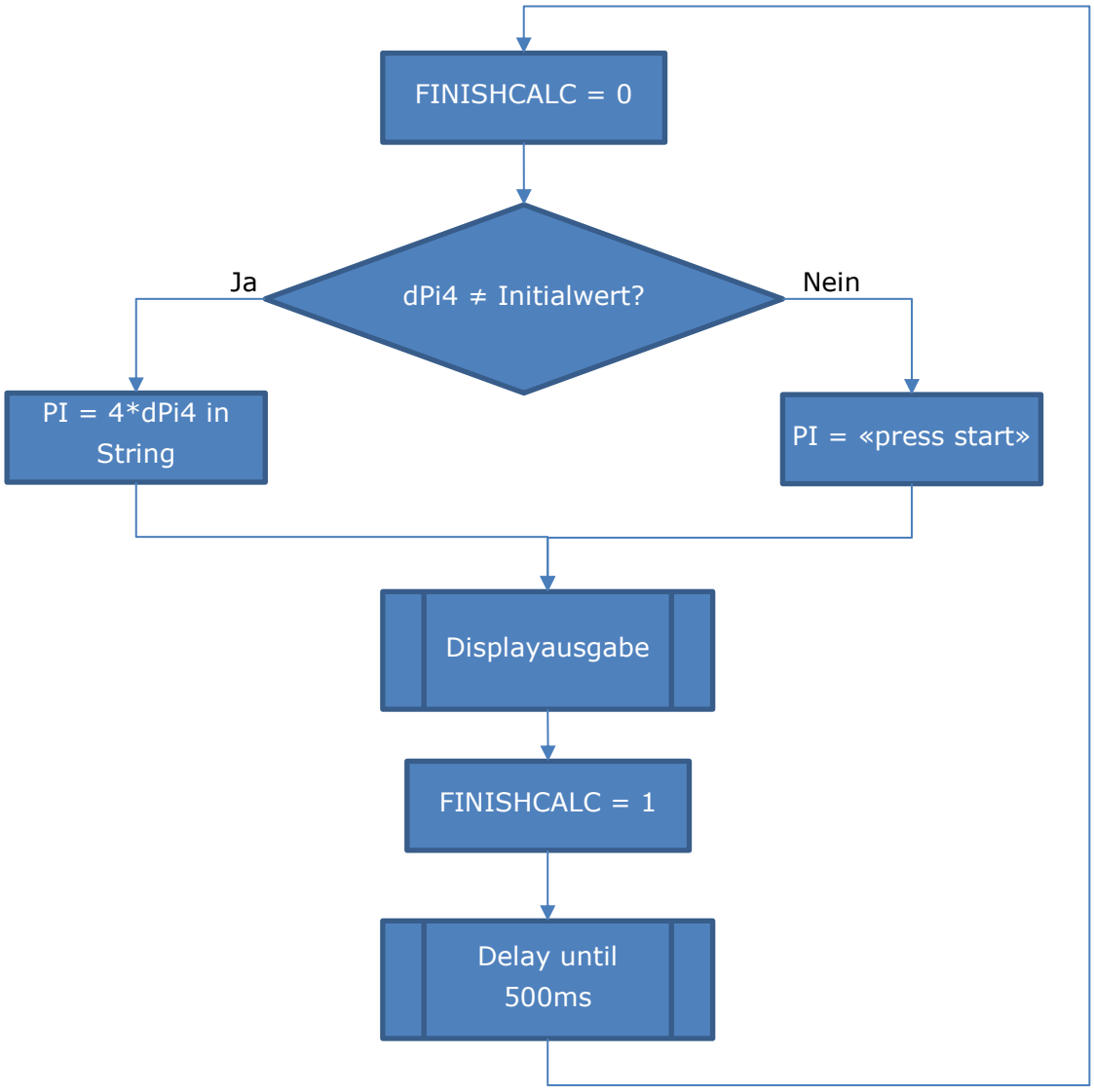
Quelle: https://de.wikipedia.org/wiki/Leibniz-Reihe

2 Tasks

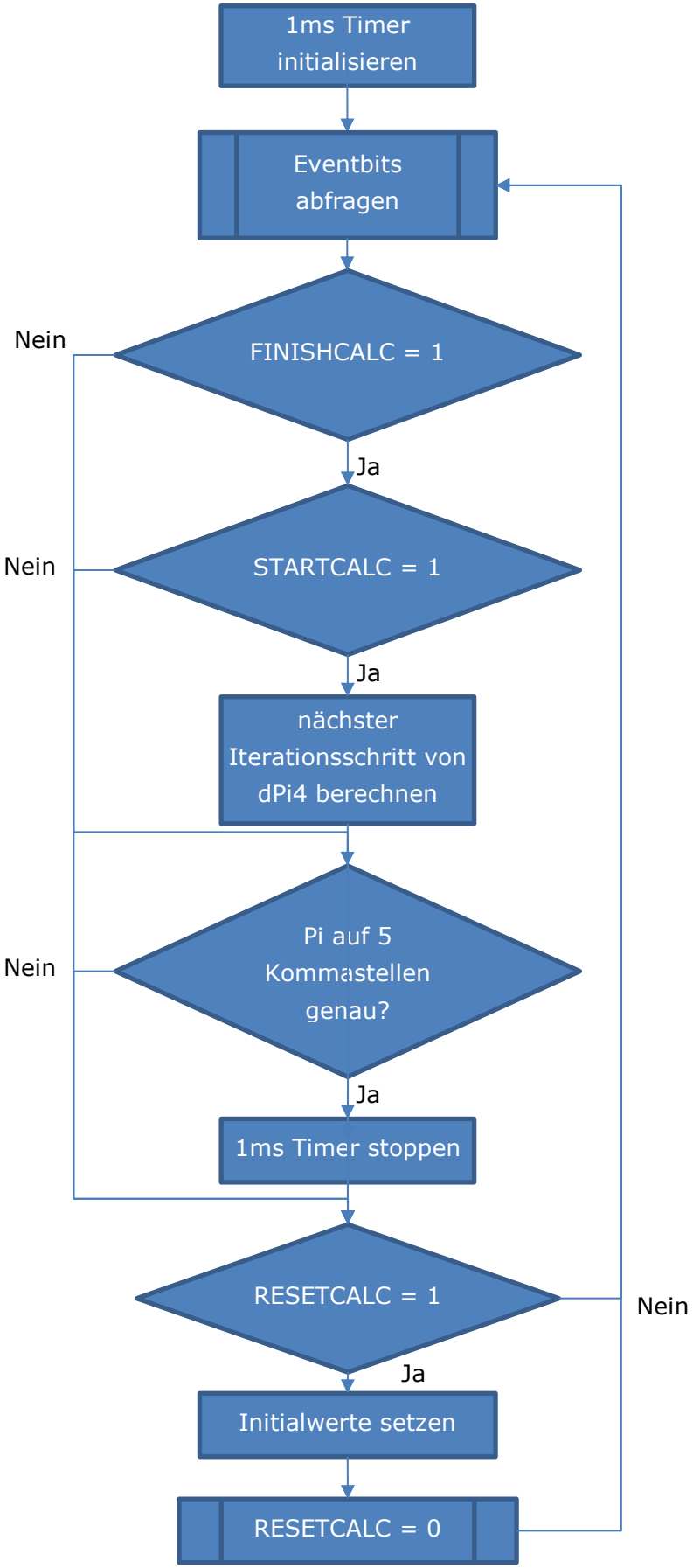
2.1 vButton



2.2 vUI


FINISHCALC = 0
dPi4 ≠ Initialwert?
Ja
Nein
PI = 4*dPi4 in String
PI = «press start»
Displayausgabe
FINISHCALC = 1
Delay until 500ms

2.3 vCalc


1ms Timer initialisieren
Eventbits abfragen
FINISHCALC = 1
Nein
Ja
STARTCALC = 1
Nein
Ja
nächster Iterationsschritt von dPi4 berechnen
Pi auf 5 Kommastellen genau?
Nein
Ja
1ms Timer stoppen
RESETCALC = 1
Nein
Ja
Initialwerte setzen
RESETCALC = 0

3 Eventbits

3.1 STARTCALC

Das Start-Eventbit erfüllt gleich 2 Aufgaben:

1. Starten der Berechnung
2. Stoppen der Berechnung

Das in der Aufgabenstellung verlangte Stopp-Eventbit wurde in das Start-Eventbit integriert und wegoptimiert.

Die Berechnung von Pi läuft, wenn STARTCALC = 1 und wird angehalten, wenn STARTCALC = 0

3.2 RESETCALC

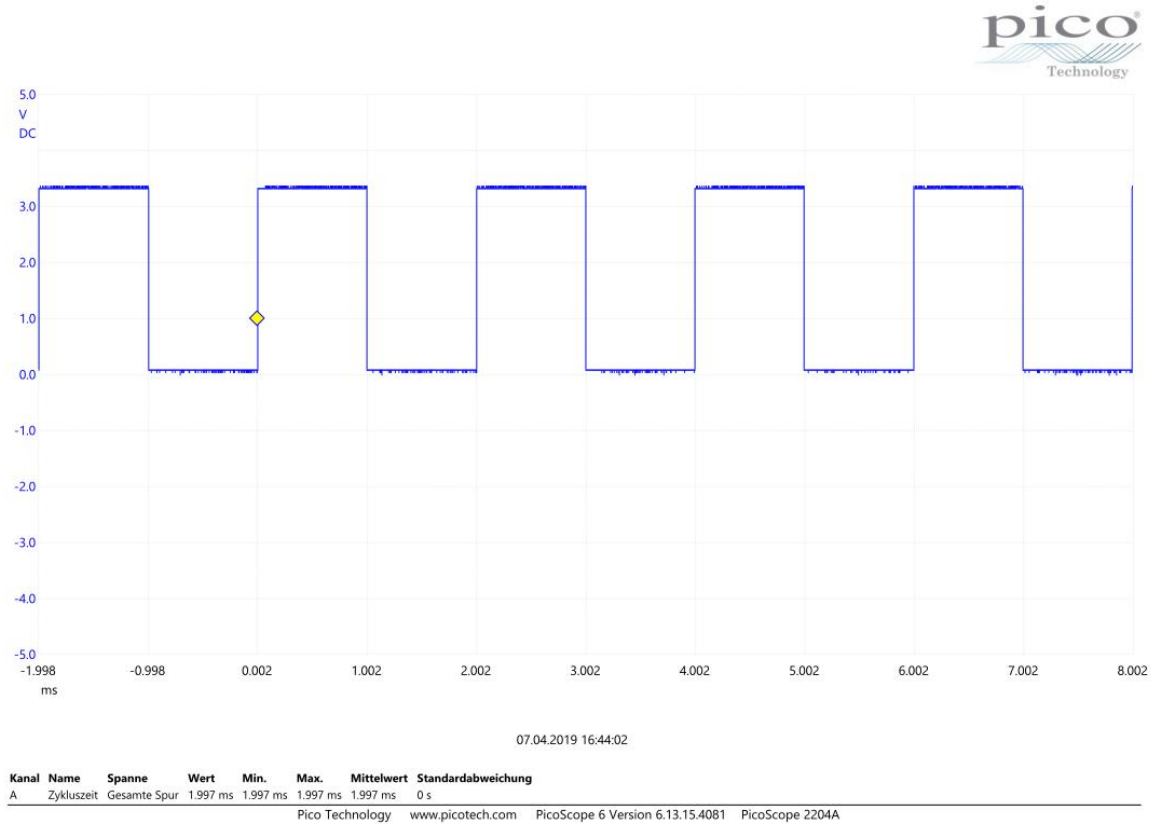
Das Reset-Eventbit bewirkt eine Rückstellung der Berechnungsvariablen auf den Initialwert. Die komplette Berechnung und wird als geresettet.

3.3 FINISHCALC

Geändert wird das Finish-Eventbit nur vom vUI-Task. Dieser pausiert sozusagen die weitere Berechnung von PI während der Ausgabe auf das Display. Diese wird nur durchgeführt, wenn FINISHCALC = 1

4 Zeitmessung

4.1 1ms Tick



Für die Zeitmessung wurde das LED1 bei jedem Tick invertiert. Mittels PicoScope wurde dann diese Zeit ausgemessen.

Gemäss der Messung dauern 2 Ticks 1.997ms. Dies ergibt einen Fehler von 0.15% und ist für unsere Aufgabe völlig ausreichend.

4.2 Messung

Für eine Zeitmessung muss die Berechnung zuerst gestoppt (Button 2) und danach zurückgesetzt (Button 3) werden. Nachdem «press start» auf dem Display erscheint, kann die Berechnung mit korrekter Zeitmessung gestartet werden (Button1).

Messung	**Zeit**
#1	10267ms
#2	10267ms
#3	10267ms
#4	10200ms
#5	10245ms
Mittelwert	10249ms

Die Berechnung von Pi dauert mit dem finalen Code durchschnittlich 10249ms.

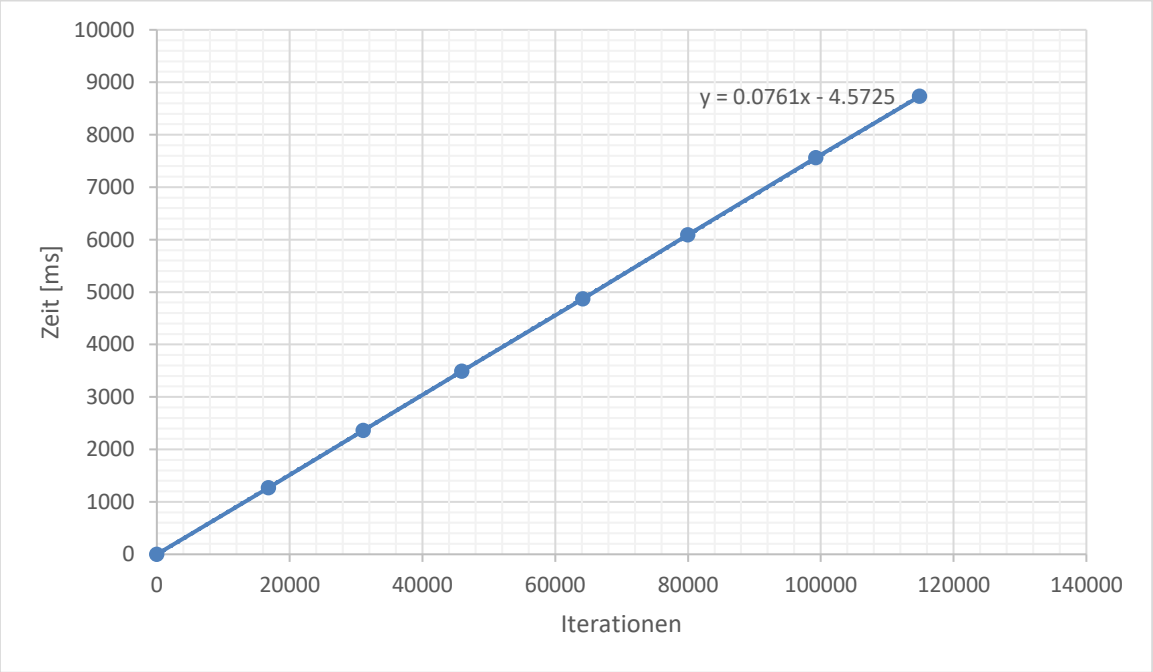
Da die Berechnung von Pi und die Ausgabe der Iterationen bei Erreichen der gewünschten Genauigkeit von Pi auf 5 Kommastellen nicht angehalten wird, wurde mittels Brakepoint die Messreihe wiederholt:

Messung	**Iterationen**	**Zeit**
#1	131852	9841ms
#2	130009	9711ms
#3	133107	9951ms
#4	128167	9603ms
#5	130285	9714ms
Mittelwert	130684	9764ms

4.3 Linearität Iterationen – Zeit

Hier soll untersucht werden, ob die Anzahl Iterationen und die gemessene Zeit linear verlaufen. Dazu wird eine Berechnung von Pi mehrmals gestoppt und die Iterationen und die Zeit in ms notiert

Iterationen	**Zeit [ms]**
0	0
16792	1270
31081	2360
45946	3491
64145	4870
79983	6090
99238	7560
114871	8730



Im Diagramm ist zu sehen, dass die Zeit mit der Anzahl Iterationen linear ansteigt. Dies beweist, dass der uC für jede Iteration der Leibniz-Reihe die gleiche Anzahl Zyklen benötigt.

4.4 Rückschluss auf Prozessorleistung

Gemäss der zweiten Messreihe waren durchschnittlich 130684 Iterationen und 9764ms nötig, um Pi auf 5 Kommastellen genau zu berechnen.

Der uC läuft mit einer Taktfrequenz von 32MHz, was einer Zykluszeit von 31.25ns entspricht.

$$\frac{9764ms}{130684\,Iterationen} = 0.0747\,\frac{ms}{Iteration} = 74.7\,\frac{us}{Iteration} = 2391\,\frac{Zyklen}{Iteration}$$

Die berechnete Zyklenzahl von 2391 pro Iteration werden für den kompletten Durchlauf der Schleife im vCalc benötigt. Dazu kommen noch Overhead vom FreeRTOS und den anderen Tasks (vUI und vButton).

Diese werden im Verhältnis aber sehr wenig aufgerufen und werden hier somit vernachlässigt.

In der vCalc-Schleife werden im Berechnungsbetrieb 3 if-Abfragen betätigt.

5 Fazit

Die Vorgaben aus der Aufgabenstellung wurden alle erfüllt.

Auf die Verwendung des Stopp-Eventbits wurde verzichtet, da dieses nur ein Soll-Kriterium war und ohne dieses das Programm schlanker gestaltet werden konnte. Es wurde also wegoptimiert.

Die Implementierung der vom Dozenten vorgeschlagenen Library zur Berechnung von float64-Zahlen wurde nicht gemacht, da dies die Aufgabenstellung nicht gefordert hatte und dies die Berechnung von Pi mit der Leibniz-Reihe massiv verlangsamt hätte.

6 Reflexion

Die Aufgabe war lehrreich, spannend und äusserst interessant. Meist sind solche Übungsaufgaben sehr weit hergeholt und motivieren daher nicht allzu sehr, Freizeit dafür zu opfern. Anders war die Berechnung von Pi mit EduBoard und FreeRTOS. Ich habe sehr viel Zeit und Interesse in dieses Projekt gesteckt - was man hoffentlich am Umfang der Dokumentation bemerkt.

Ich hätte mir gewünscht, dass diese Arbeit noch 1-2 Ziele mehr gehabt hätte, und dafür auch benotet werden würde. Bei motivierenden Arbeiten wie diese bleibt viel mehr hängen, als wenn man für eine Prüfung lernt. Da wird nach der Abgabe der Prüfung der Kopf wieder für neues frei gemacht und relativ schnell hat man alles wieder vergessen. Bei einer über längere Zeit dauernde Arbeit lohnt sich die investierte Zeit viel mehr und man erarbeitet sich die neuen Erkenntnisse so, dass diese einen viel grösseren Lerneffekt bewirken.

Ich hätte auch gerne noch einen zweiten und dritten Algorithmus implementiert und das Programm weiter ausgebaut. Im Grossen und Ganzen bin ich aber zufrieden mit meiner Arbeit.