Juventus
Technikerschule HF
www.technikerschule.ch

Embedded Systems

Bericht Semesterarbeit Pi

der **Juventus Technikerschule HF**

Inhaltsverzeichnis:

1 Einleitung

Die Aufgabe besteht darum mit dem EDU Board zwei Algorithmen zu programmieren die Pi näherungsweise berechne. Dies auf fünf Stellen nach dem Komma. Dazu wurde uns der Leibnitz Algorithmus vorgegeben und der zweite ist frei zu wählen. Als zweiter Algorithmus wurde hier der von Euler verwendet dazu später mehr.

Eine weiter Vorgabe ist das Verwenden von FREE RTOS und das Erstellen der einzelnen Task. Es müssen min. drei Task verwendet werden. Zudem soll der aktuelle Wert der berechnung am Display angezeigt werden. Die Funktionen sollen über die Knöpfe gesteuert werden.

1.1 Euler Algorithmus

Beim zweiten Algorithmus wird ein Verfahren von Euler angewendet. Es Handelt sich dabei um das Basler Problem, das der Frage nach geht, was das Ergebnis der Summe der reziproken Quadratzahlen ist. Den Namen Basler Problem kommt davon das sich erst Jakob Bernoulli und nachher Leonard Euler an dem Problem versucht haben. Beides bekannte Basler Mathematiker. Euler konnte das Problem schliesslich 1735 lösen und kam auf Folgendes Ergebnis:

$$\frac{\pi^2}{6} = \sum_{n=1}^{\infty} \frac{1}{n^2}$$

Abbildung 1 Pi nach Euler

Die Lösung der Summe strebt nach einem Bruchteil von Pi ($\frac{\pi^2}{6} = 1.644943$) und kann deshalb zur berechnung von Pi verwendet werden. Bleibt nur die Frage wie schnell das ganze geht, um auf die Genauigkeit von fünf Stellen nach dem Komma Zu kommen.

2 Aufbau

2.1 Tasks

Es werden 4 Tasks erstellt:

```
void controllerTask(void* pvParameters);
void calculatLeibniz(void* pvParameters);
void calculatEuler(void* pvParameters);
void ButtonTask(void *pvParameters);
```

Abbildung 2 Task Übersicht

2.1.1 ControllerTask

Hier werden alle Tasks gesteuert und mit Hilfe von Eventbits und einer Statemachine überwacht. Die Eventbits und die Statemachine werden separat erklärt. Im Task werden die einzelnen Eventbits verarbeitet und so gesteuert welcher Task grad läuft für die Pi berechnung. Die Display Ausgabe wird im Controllertask ausgeführt so wie das sichere Einlesen vom Aktuellen Wert von Pi. Dieser Task hat die Priorität 2. So kommt er bevorzugt vor den Berechnungstask dran ist aber hinter dem ButtonTask.

2.1.2 CalculatLeibniz

In diesem Task wird Pi mit Hilfe der Leibnitz Reihe berechnet. Der Task schreibt den berechneten Wert nach jedem Rechnung Schritt in eine globale Variabel für Pi. Zudem überprüft er die Genauigkeit für Pi nach jedem 20 Schritt. Falls die gewünschten 5 Stellen nach dem Komma erreicht sind, meldet er das via Event Bit. Hier nochmals die Formel der Leibnitz Reihe: Dieser Task hat die letzte Priorität (1) so kann er immer rechnen wir aber gestoppt, wenn Task mit höherer Priorität bedarf, anmelden.

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

Abbildung 3 Leibnitz Reihe für Pi

2.1.3 CalculatEuler

In diesem Task wird Pi mit Hilfe der Euler Reihe berechnet. Der Task schreibt den berechneten Wert nach jedem Rechnung Schritt in eine globale Variabel für Pi. Zudem überprüft er die Genauigkeit für Pi nach jedem 20 Schritt. Falls die gewünschten 5 Stellen nach dem Komma erreicht sind, meldet er das via Event Bit. Dieser Task hat die letzte Priorität (1) so kann er immer rechnen wir aber gestoppt, wenn Task mit höherer Priorität bedarf, anmelden.

2.1.4 ButtonTask

Im ButtonTask werden die Taster (Knöpfe) eingelesen und einzelne Event Bits gesetzt. Dieser Task wurde extra vom Controller Task getrennt, um die Übersichtlichkeit zu verbessern. So kann zudem garantiert werden das die Tasten sicher eingelesen werden da dieser Task eine höhere Priorität bekommt wie der Controllertask.

2.2 Statemachine

Die Statemachine hat drei Zustände. Gesteuert werden diese durch einen internen Timer der die Tasks alle 200ms bzw. 500ms startet. Diese gehen automatisch zurück und den Idle Zustand, sobald sie ihre Aufgabe erledigt haben. Somit kann garantiert werden das im Pi Übernehme Zustand die berechnung von Pi gestoppt werden kann. Sobald das geschehen ist wird Pi von der globalen Variabel in die lokale Variable für Pi geschrieben. Danach wird der Task wieder gestartet. Das Ganze wird in einem Intervall von 500ms durchgeführt. Die lokale Variabel wird jeweils alle 200ms am Display aktualisiert. Im Display wird jeweils angezeigt mit welchem Algorithmus gerade gerechnet wird und wie lange das schon passiert. Befinden wir uns im Idel Zustand werden die ganzen Eventbits verarbeitet bzw. gesetzt. Zudem läuft da auch der Timer der die anderen Zustände aufruft. Das führt zwar zu einer Verzögerung da der Timer nur im Idle Zustand läuft. Das kann aber vernachlässigt werden, da die Zeitliche Abweichung nicht gross genug ist das wir davon etwas merken als Bediener.

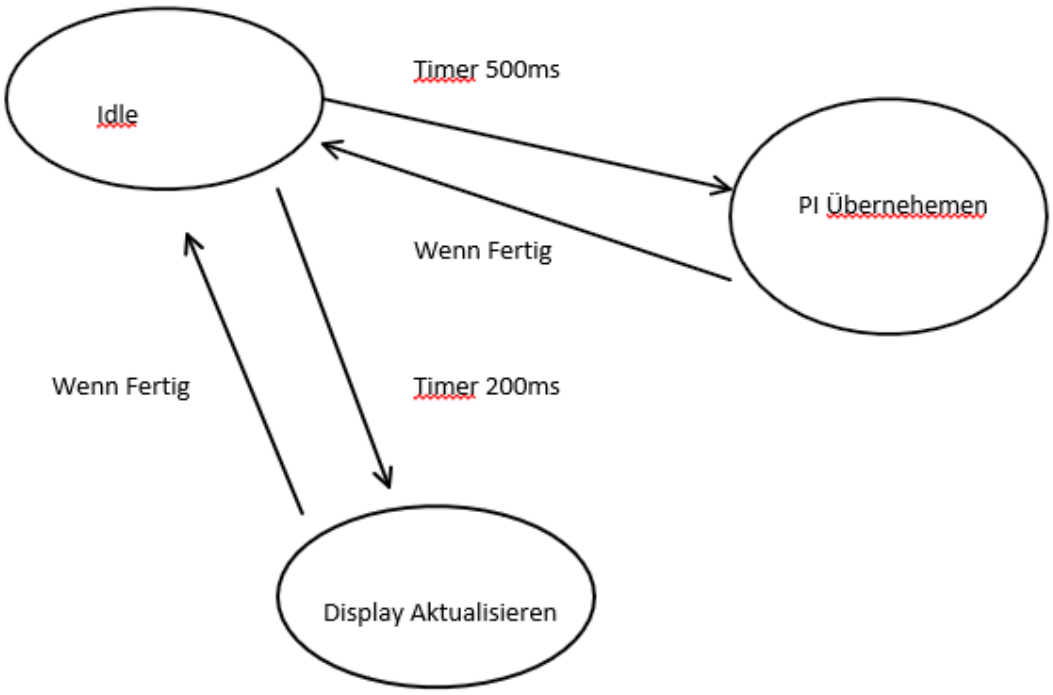

Abbildung 4 Statemachine

2.3 EventBits

Folgende Eventbist werden verwendet:

```
EventGroupHandle_t egButtonEvents = NULL;
#define BUTTON1_SHORT    0x01    // Startet den Algorithmus
#define BUTTON2_SHORT    0x02    // Stoppt den Algorithmus
#define BUTTON3_SHORT    0x04    // Setzt den Algorithmus zurück
#define BUTTON4_SHORT    0x08    // Wechseln vom Algorithmus
#define START_PI_1       0x10    // Start Stopp Bit für Leibnitz Pi Berechnung
#define START_PI_2       0x20    // Start Stopp Bit für Euler Pi berechnung
#define PI_READY         0x40    // Rückmeldung sobald Pi Task beendet und auf Stopp ist
#define PI_GENAU         0x80    // Pi Genauigkeit erreicht
#define BUTTON_ALL       0xFF    // Rücksetzten der Event Bit
```

Abbildung 5 Eventbit Liste

Es werden 9 Eventbits verwendet. Die ersten vier dienen für die Tasten Funktionen. Diese werden beim Tasten druck gesetzt und nach dem Verwenden wieder zurückgesetzt. Die vier weiteren dienen dem Steuern der Berechnung Tasks. Diese starten die Tasks bzw. pausieren sie und Melden sobald die Tasks gestoppt sind. Mit dem neunten können wiederum alle auf 0 gesetzt werden.

3 Fazit

3.1 Problem

3.1.1 Statemachine Zustand 2

Das Programm startet alles Task und geht das erste Mal in den Zustand Display Aktualisieren. Danach geht es aber nicht zurück in den Zustand Idle. Somit lässt sich nicht mehr steuern. Der String für die Display Ausgabe war zu lang. Das hat zu einem Absturz geführt. So wurde die Rücksetzten vom Zustand auf Idel nie erreicht. Seit der String die maximale Grösse einhaltet, tritt das Problem nicht mehr auf.

3.1.2 Berechnungs Algorithmus

N wir jedes Mal wieder auf 3 zurückgesetzt. Variable N wurde jedes Mal wieder in der For Schleife mit 3 initialisiert. Deshalb die Initialisierung aus der For Schleife verlegt.

3.1.3 Pi wird nur mit 3 Stellen dargestellt

Pi wird auf dem Display nur mit 3 Nachkommastellen dargestellt. Das Float muss via String variable noch formatiert werden. Das ist in der vDisplay Funktion nicht implementiert. Mit der Formatierung %1.5f funktioniert es einwandfrei.

3.1.4 Vergleich der Genauigkeit

Der vergleich der Genauigkeit hat mit der Variante von String vergleich nicht funktioniert. Wahrscheinlich, weil da auch noch Zeichen die nicht definiert wurden drin standen und so keine exakte Gleichheit erreicht wurde. Das vergleichen über Int Variablen hat dann fehlerfrei funktioniert.

3.1.5 Berechnung geht nur einen schritt

Nach dem ersten Aktualisieren des Displays mit dem Wert für Pi bleibt das Programm hängen. Die genaue ursache konnte nicht ermittelt werden. Nach einer generellen Überarbeitung vom Berechnung Task ist der Fehler nicht mehr aufgetreten.

3.1.6 Fehler im Euler Algorithmus

Der Algorithmus kommt nach dem ersten Durchlauf direkt auf Unendlich. Der Fehler lag an dem Verwenden des falschen Zeichens für Hoch. Zuerst wurde das ausserhalb C übliche ^ verwendet. In Math.h wird aber pow(b,e) als hoch verwendet.

3.2 Auswertung

Mit dem Algorithmus von Euler kann innerhalb von 12s die Genauigkeit von Pi berechnet werden. Das funktioniert so weit einwandfrei. Bei der Methode von Euler sieht das ganze nicht so rosig aus. Der Algorithmus ist viel langsamer. Nach ca. 50'000 Durchläufen kommt er nicht mehr weiter. Er hat sich bis dahin auf 3.14139 angenähert und bleibt dort stecken. Auch mit über einer Million Durchläufen ändert sich nichts mehr. Das weil die Änderungen wohl so klein werden das sie nicht mehr in die float variabel passen. Abschliessen kann wohl gesagt werden das die Euler Methode nicht verwendet werden kann, um schnell viele Stellen von Pi zu berechnen.

Ich hatte einige mühen das Programm Initial zum Laufen zu bringen. Da C nicht ganz intuitiv ist für mich. Ich bin aber froh, dass der Leibniz Algorithmus, doch so läuft wie angedacht. Beim Euler war mir nicht klar, dass er so lange hat für Pi. Ich habe damit gerechnet das er länger hat da er doch sehr schnell sehr klein wird. Der Task läuft so lange bis die 32 Bit Variabel von n überläuft weiter.

4 Anhang

4.1 Abbildungsverzeichnis

4.2 Quellenverzeichnis

- Aufgabestellung: https://lernen.juvecampus.ch/auth/RepositoryEntry/113672720/CourseNode/96213599132445/path%3D~~%C3%9Cbungen/0
- Basler Formel: https://de.wikipedia.org/wiki/Basler_Problem
- Free RTOS: https://www.freertos.org/
- Projekt Repository: https://github.com/phschwenk/U_PiCalc_HS2022