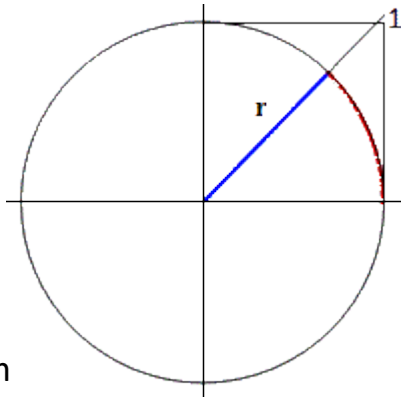
Übung: Calculating PI

Algorithmen:

Es gibt diverse Algorithmen, wie man PI berechnen kann. Einige sind schneller als andere.



Ein sehr simpler, jedoch auch langsamer Ansatz ist folgender:
Wenn man in einem Quadrat mit der Seitenlänge 1 zwei zufällige Punkte wählt und deren Distanz zur linken unteren Ecke berechnet, dann bekommt man entweder einen Wert über oder unter 1.

Nun besagt der Ansatz, dass das Verhältnis der Punkte im Viertel-Kreis drin im Vergleich zum ganzen Quadrat einem Viertel PI entspricht. Diese Tatsache kann man sich zum Vorteil nehmen und Punkte in diesem Bereich berechnen.

Einen anderen Weg zu PI stellen Reihen dar. Eine einfache Reihe für diesen Zweck ist die **Leibniz-Reihe**. Diese konvergiert, je weiter man sie berechnet, immer mehr gegen PI/4.

$$1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \cdots = \frac{\pi}{4}$$

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.

Dabei soll folgendes stets gegeben sein:

- 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.)

Dokumentationsarbeiten:

Es soll ein kurzer Bericht zum Projekt verfasst werden. (ca. 8+ Seiten)
Darin enthalten sein soll folgendes:

- Erklärung des zusätzlich gewählten Algorithmus.
- Beschreibung der Tasks. Der Ablauf der Tasks und deren Funktionsweise muss ersichtlich sein. (z.B. Finite State Machine)
- Die EventBits/TaskNotification und deren Aufgabe/Verwendung soll erklärt werden.
- Die Resultate der Zeitmessung sollen aufgeführt und erklärt werden.
- Vergleiche die Geschwindigkeit von Leibniz gegenüber der anderen Methode
- Wage einen Rückschluss bezüglich der gemessenen Rechenleistung zur tatsächlichen Prozessorleistung zu machen.
- Die Softwareverwaltung GIT ist anzuwenden und online in einem Repository zu sichern.
  Die Aufgabe muss bis zum 24.10.2023 erledigt sein und abgegeben werden. (Link auf Repository auf Github reicht. Bericht ebenfalls im Repository)

Die Übung wird benotet.

Gewertet werden folgende Punkte:

- Vollständigkeit gemäss Aufgabenstellung
- Einhalten der Vorgaben gemäss Aufgabenstellung
- Realisierung der zwei Algorithmen
- Realisierung des Steuertasks
- Realisierung des Zeit-Mess-Funktion
- Vollständigkeit der Dokumentation
- Korrekte Benutzung von Git

Folgende Einstellungen müssen in den Projektoptionen vorgenommen werden:

AVR/GNU Linker

- ➔ *General: «Use vprintf library(-WL,-u,vfprintf)» aktivieren*
- ➔ *Miscellaneous: Other Linker Flags: «-lprintf_flt»*

Damit wird die Unterstützung von Float und Double in sprintf aktiviert.

Die FLoat-Ausgabe des Display-Treibers reicht nicht um genügend Stellen nach dem Komma auszugeben. Es muss sprintf aus der Library <stdio.h> verwendet werden.

Es ist ein bereits aufgesetztes Projekt (inklusive Buttonhandler und Float-Einstellungen) auf Github zu finden unter https://github.com/Juventus-Technikerschule-HF/U_PiCalc_HS2023_Template

Atmel bietet nur eine Unterstützung von 32bit floating Point Zahlen. (gemäss https://de.wikipedia.org/wiki/IEEE_754)

Wenn 64bit Floating benötigt wird gibt es eine Library dazu. Das macht die Sache zwar etwas komplizierter, ist dafür aber halt genauer.

Dazu gibt es ein Demo-Projekt für unser EDUBoard
https://github.com/Juventus-Technikerschule-HF/Float64Demo