Zyklische Ablaufsteuerung (minimales Betriebssystem) mit Timern

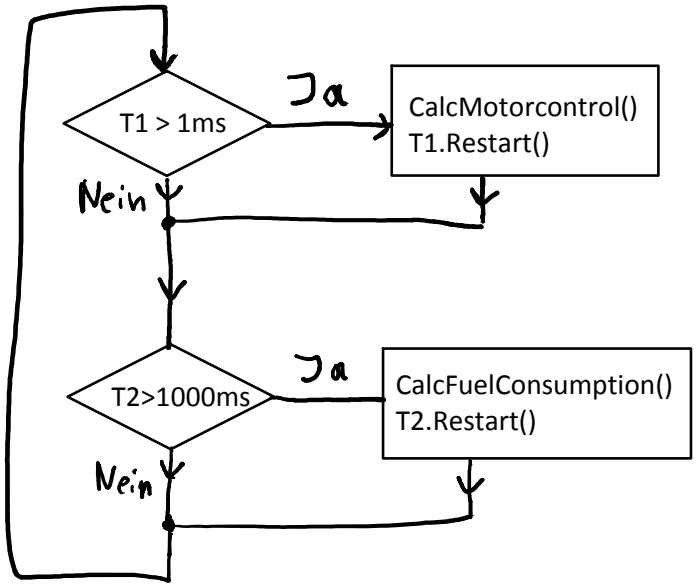
In embedded und Echtzeitsystemen ist es sehr of notwendig verschiedene Aufgaben immer wieder zyklisch mit unterschiedlichen Zykluszeiten (Frequenzen) auszuführen.

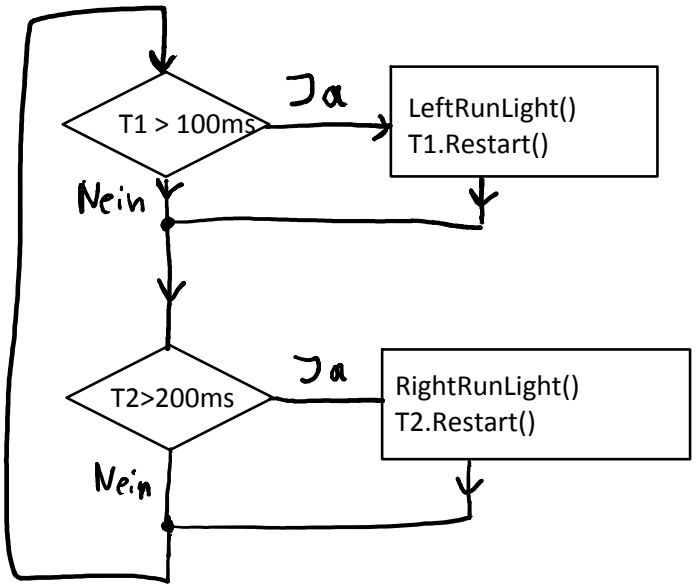
Nehmen wir als Beispiel die Motorkontrol-Einheit eines Autos:

- Die Motorkontrolle (Zündzeitpunkt, Eispritzmenge . . .) muß mit einer Zykluszeit von 1MSec aufgerufen werden.
- Für die Berechnung des Spritverbrauchs reicht ein Aufruf mit einer Zykluszeit von 1000MSec.

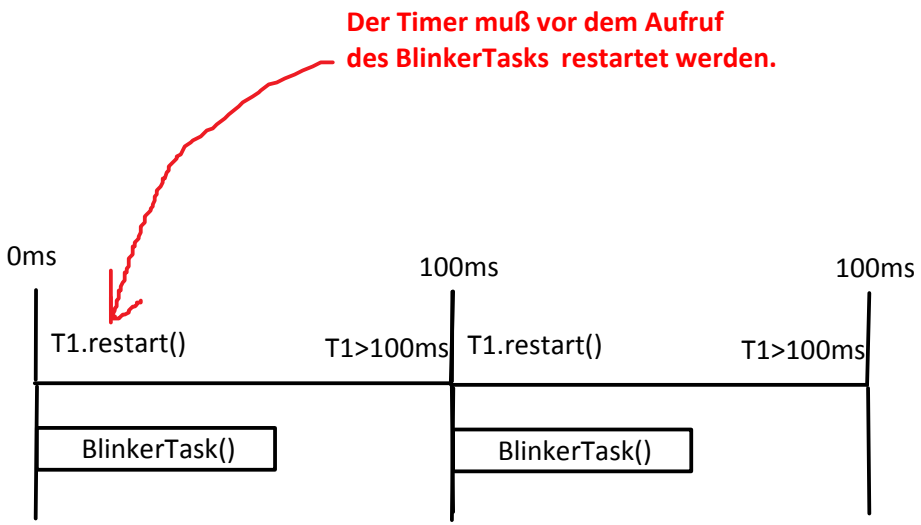
Es gibt keinen Grund die Spritverbrauchs -Berechnung öfter als 1x pro Sec. aufzurufen und dadurch Rechenzeit zu verschwenden.

Wir werden diese als **zeitliches Scheduling** bezeichnete Technik verwenden, um um Lauf und Blinklichter mit unterschiedlichen Frequenzen laufen und blinken zu lassen.

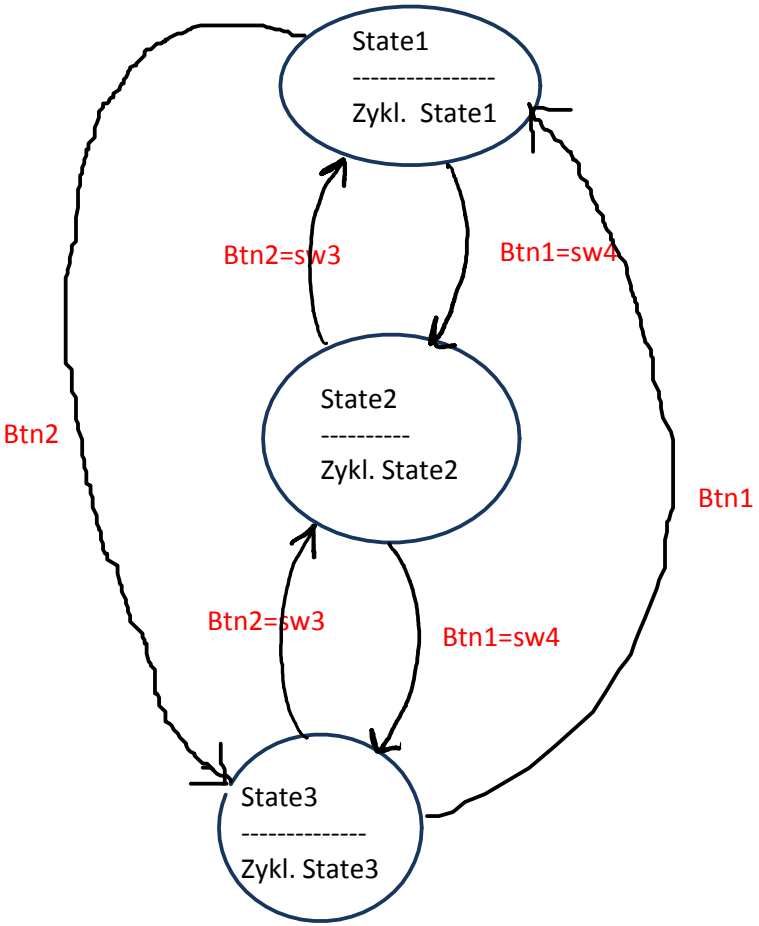




Zyklische Ablaufsteuerung (minimales Betriebssystem) mit Timern



State-Diagramm für die 3-Zustände einer Fahrradleuchte



Der Zustand der Fahrradleuchte soll auch noch mit 2 Leds dargestellt werden
01 10 11

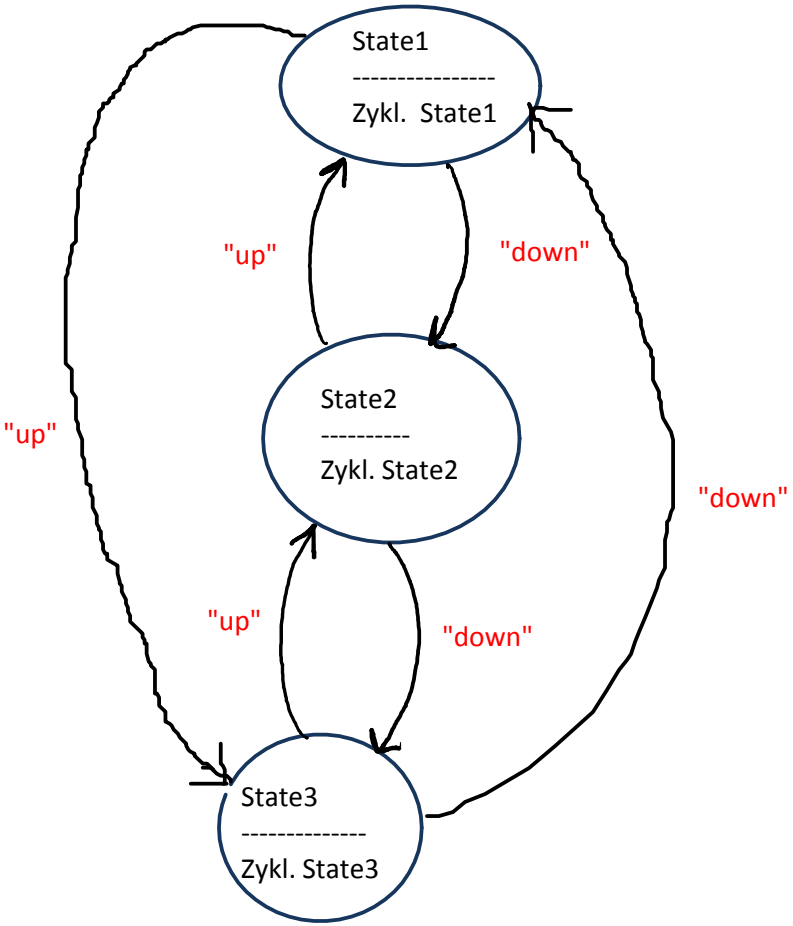
Zykl. State1: Blink 2Hz

Zykl. State2: RunLight-Left 5Hz

Zykl. State3: RunLight-Right 10Hz

Erweiterung State4:
Die Fahrradleuchte ist **aus**

Fahrradleuchte mit Steuerung und Statusanzeige über eine C#-App



Die Zustände werden durch eine blokierende String-Kommunikation
mit einer C#-App umgeschaltet und angezeigt.

Da die Zustandsfunktionen im Read() an der Seriellen blokiert sind
werden die zyklischen Aktionen von einer Timer-Interruptroutine ausgeführt.

Die Lösung mit blokierendem Read() brauchen wir nur wenn wir das **mbed::serial** verwenden.
Mit **Serial_HL.h** können wir Strings auch **nonblocking** lesen und die StateMachine so aufbauen wie wir es mit unserem standart Statemachine-Pattern gewohnt sind

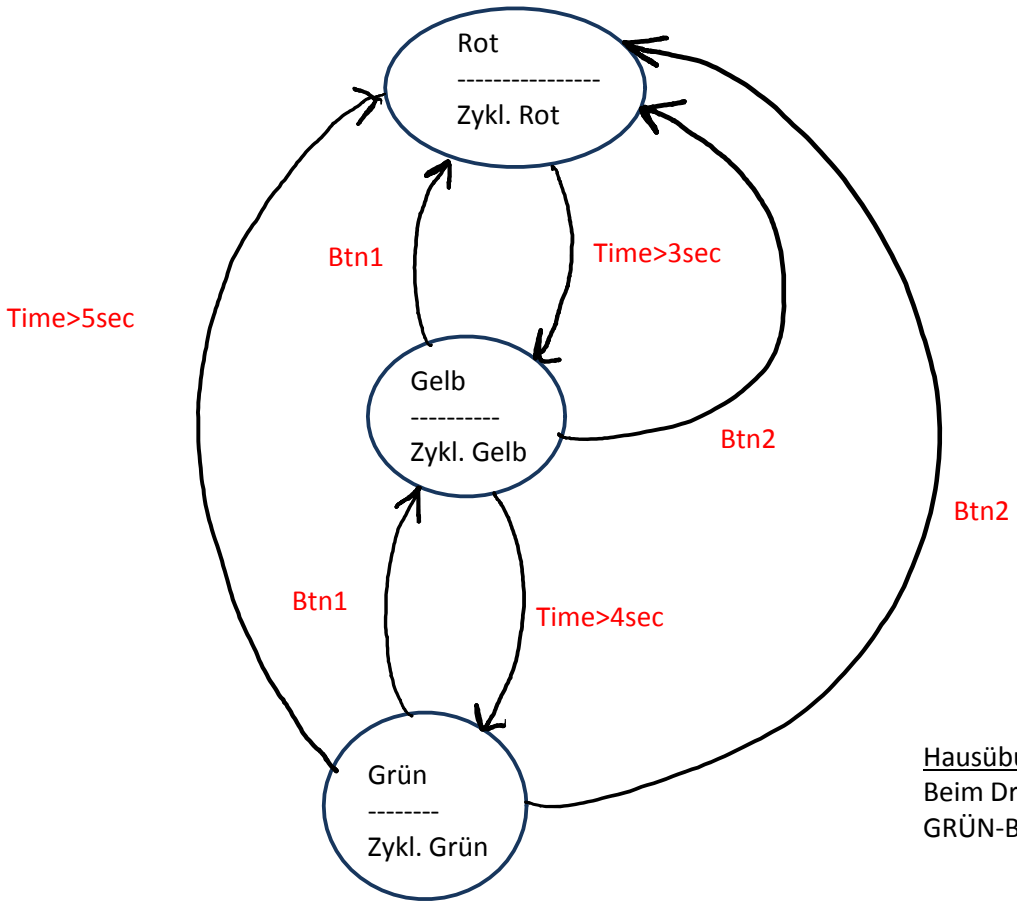
Zykl. State1: Blink 2Hz

Zykl. State2: RunLight-Left 5Hz

Zykl. State3: RunLight-Right 10Hz

Erweiterung State4:
Die Fahrradleuchte ist **aus**

State-Diagramm für eine Ampelsteuerung



Zykl. Rot: Blinken und Time anzeigen

Zykl. Gelb: Blinken und Time anzeigen

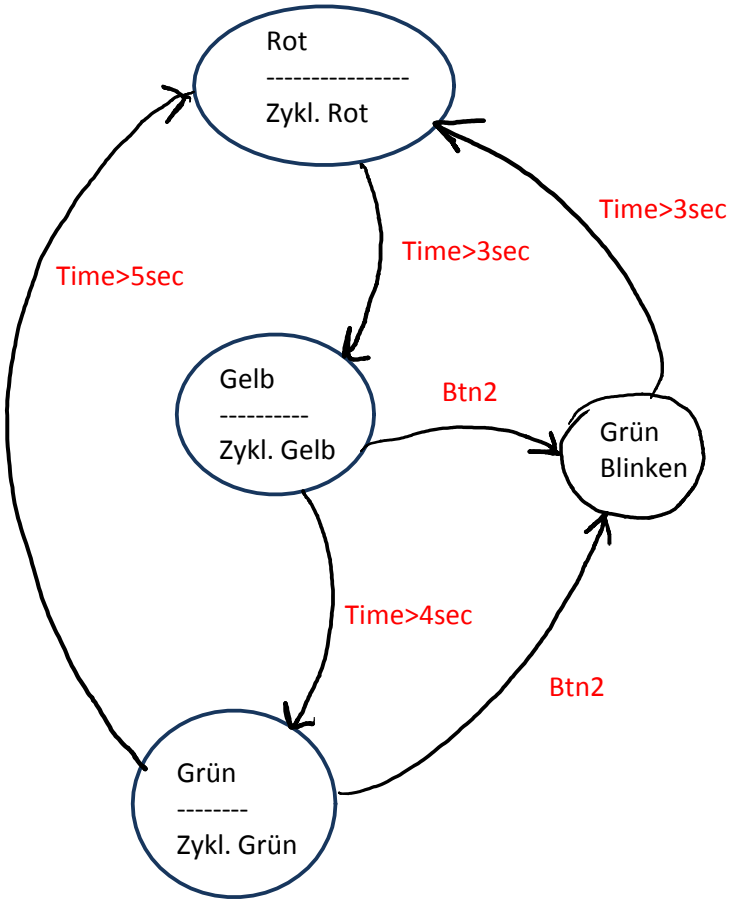
Zykl. Grün: Blinken und Time anzeigen

Mit dem Btn2 kommt man aus jedem Zustand wieder zurück in den Zustand Rot

In jedem Zustand soll auf der Textanzeige die Zeit angezeigt werden die in diesem Zustand schon verstrichen ist.

Hausübung
Beim Drücken von Btn2 geht die Ampel zuerst in den Zustand GRÜN-BLINKEN und dann erst nach ROT

State-Diagramm für eine Ampelsteuerung



Zykl. Rot: Blinken und Time anzeigen

Zykl. Gelb: Blinken und Time anzeigen

Zykl. Grün: Blinken und Time anzeigen

Mit dem Btn2 kommt man aus jedem Zustand wieder zurück in den Zustand Rot

In jedem Zustand soll auf der Textanzeige die Zeit angezeigt werden die in diesem Zustand schon verstrichen ist.

Hausübung
Beim Drücken von Btn2 geht die Ampel zuerst in den Zustand GRÜN-BLINKEN und dann erst nach ROT

Zyklische Ablaufsteuerung (minimales Betriebssystem) mit Timern aus dem Echtzeitbuch

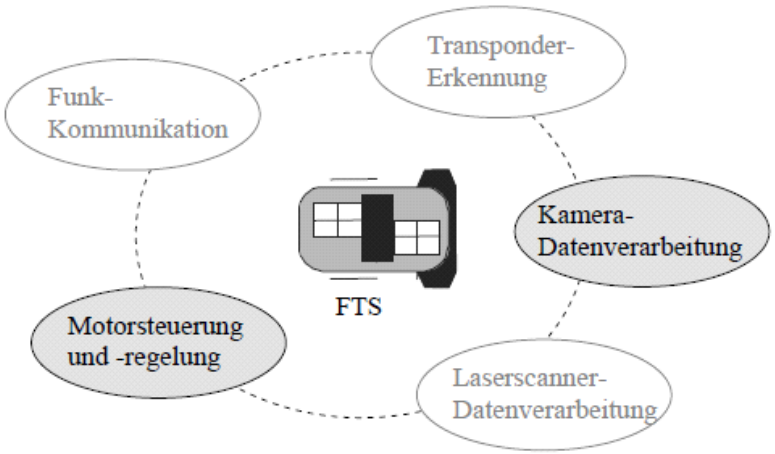


Abb. 5.9. Vereinfachtes FTS Beispiel mit zwei periodischen Aufgaben

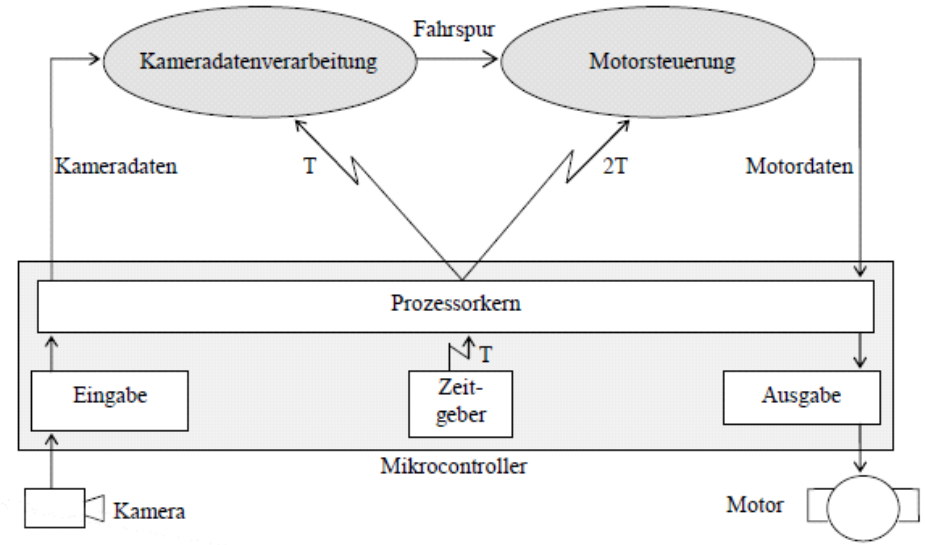


Abb. 5.10. Aufbau der FTS Steuerung mit einem Mikrocontroller

Ablaufplanung

Aufgabe	Periode
Kameradaten-verarbeitung	T
Motor-steuerung	2T

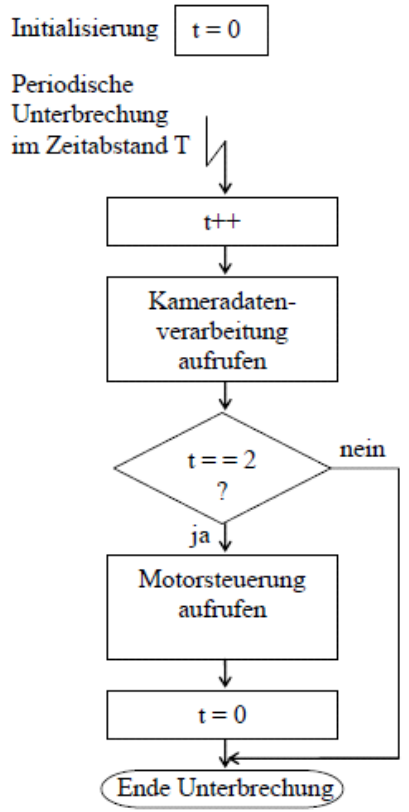


Abb. 5.11. Ablaufplanung und deren Realisierung für die einfache FTS-Steuerung

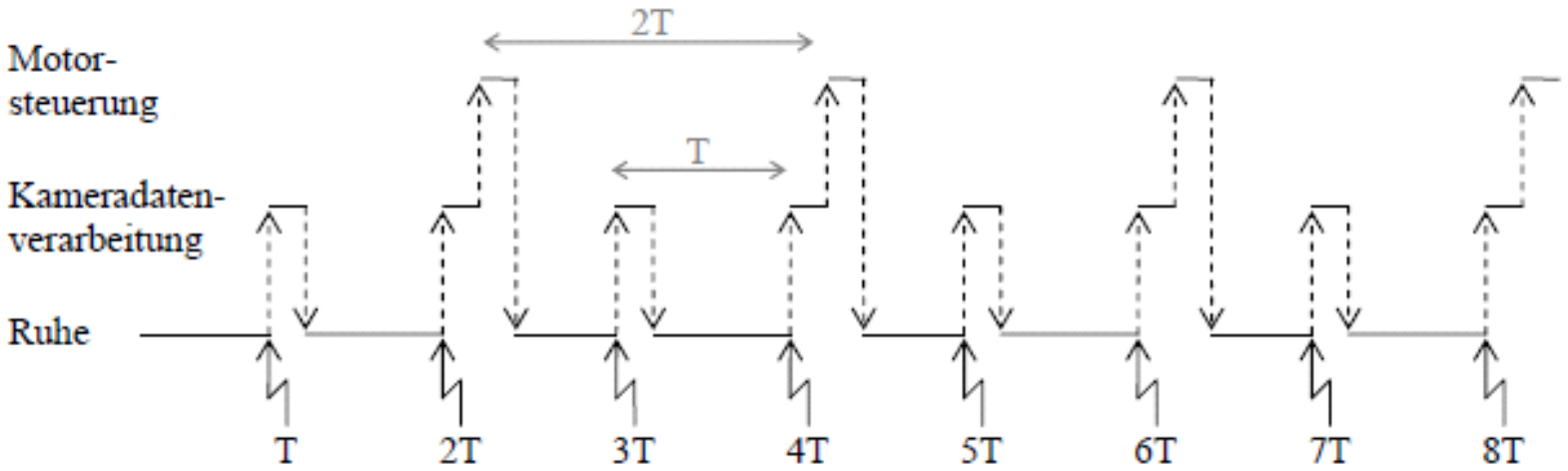
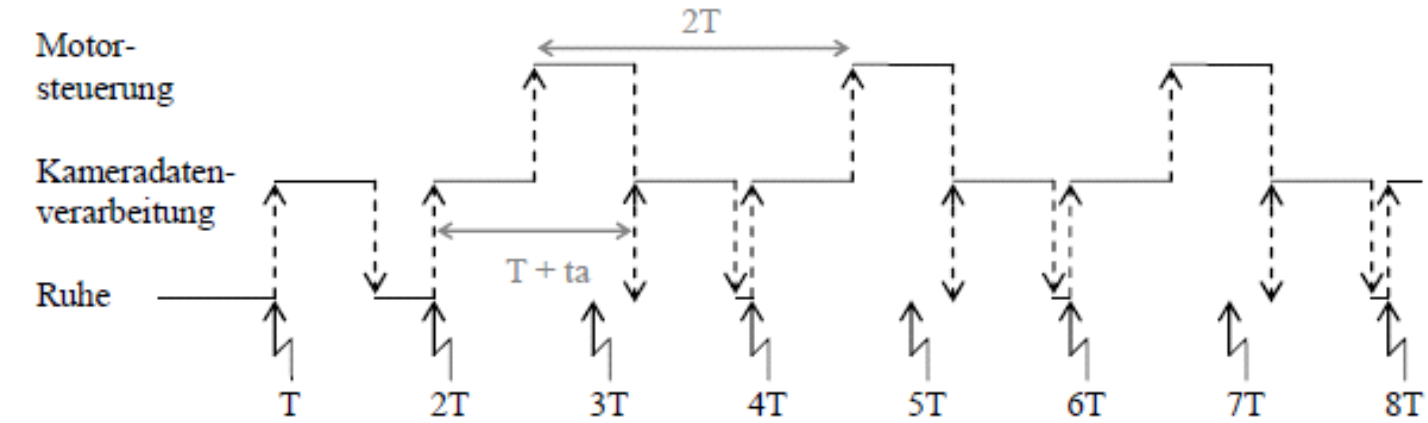


Abb. 5.12. Zeitlicher Ablauf der einfachen FTS-Steuerung



ta : Ausführungszeit Kameradatenverarbeitung + Ausführungszeit Motorsteuerung -T

Abb. 5.13. Periodenabweichung bei Summe der Ausführungszeiten größer T