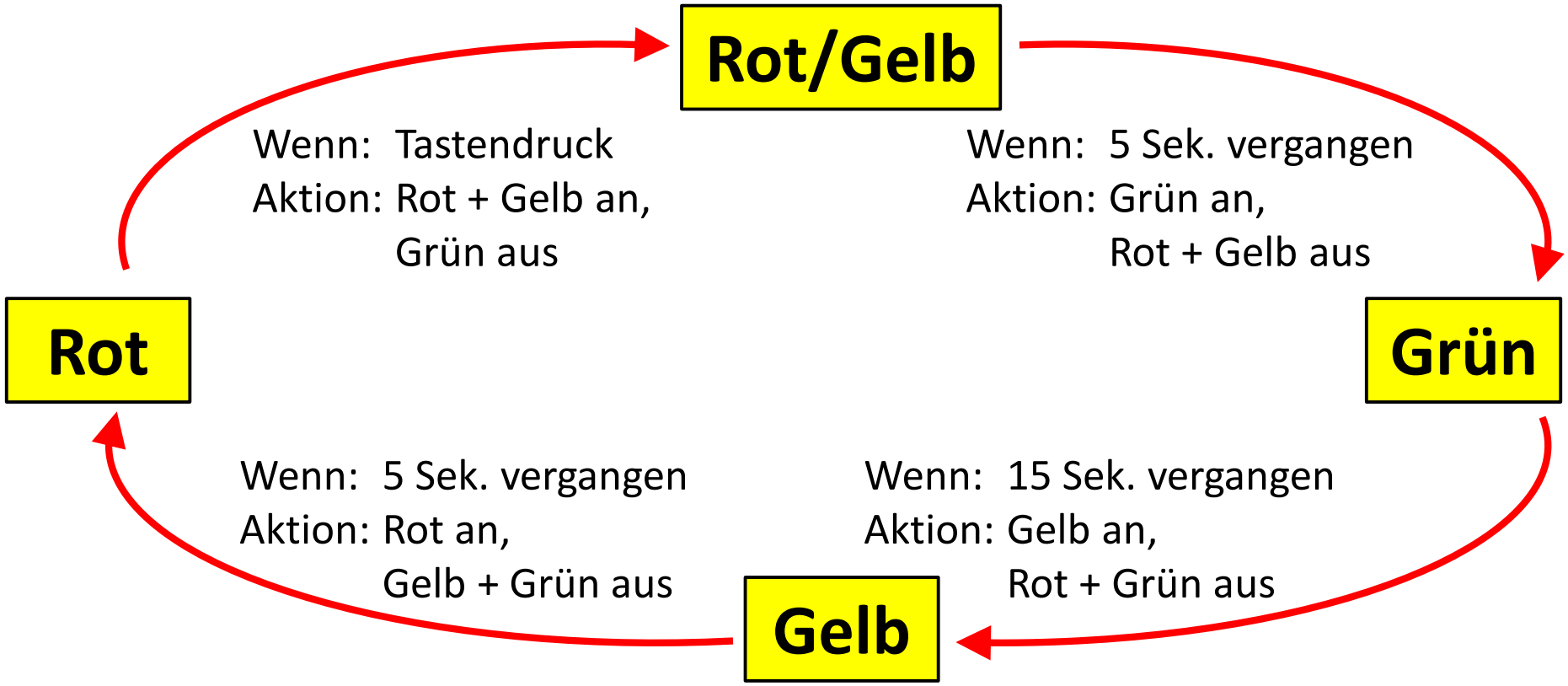
Einleitung

Eine Verkehrsampel durchläuft verschiedene Zustände. Bestimmte Ereignisse – zum Beispiel ein Tastendruck oder der Ablauf einer Wartezeit – führen zum Wechsel des aktuellen Zustands. Ein vereinfachtes Beispiel:



Einleitung

Ausschnitt aus der Implementierung des endlichen Automaten. Hinweis: Die Funktionen rot(), rot_gelb(), gruen() und gelb() schalten die jeweiligen Lampen der Ampel ein…

```
#define ZUSTAND_ROT      1
#define ZUSTAND_ROTGELB  2
#define ZUSTAND_GRUEN    3
#define ZUSTAND_GELB     4

uint32_t start = 0;
uint8_t state = ZUSTAND_ROT; rot();  // Startzustand: Rot einschalten

while(1)  // Hinweis: Millis() ermittelt die Millisekunden seit Programmstart
{
  switch(state)
  {
  case ZUSTAND_ROT:
    if(tastendruck()) { rot_gelb(); start=Millis(); state=ZUSTAND_ROTGELB; }
    break;
  case ZUSTAND_ROTGELB:
    if(Millis()-start>5000) { gruen(); start=Millis(); state=ZUSTAND_GRUEN; }
    break;
  case ZUSTAND_GRUEN:
    if(Millis()-start>15000) { gelb(); start=Millis(); state=ZUSTAND_GELB; }
    break;
  case ZUSTAND_GELB:
    if(Millis()-start>5000) { rot(); state=ZUSTAND_ROT; }
    break;
  }
}
```

Endlicher Automat

(engl. finite-state machine)

A finite-state machine (FSM) is a mathematical model of computation used to design both computer programs and sequential logic circuits.

It is conceived as an abstract machine that can be in one of a finite number of states. The machine is in only one state at a time; the state it is in at any given time is called the current state. It can change from one state to another when initiated by a triggering event or condition; this is called a transition.

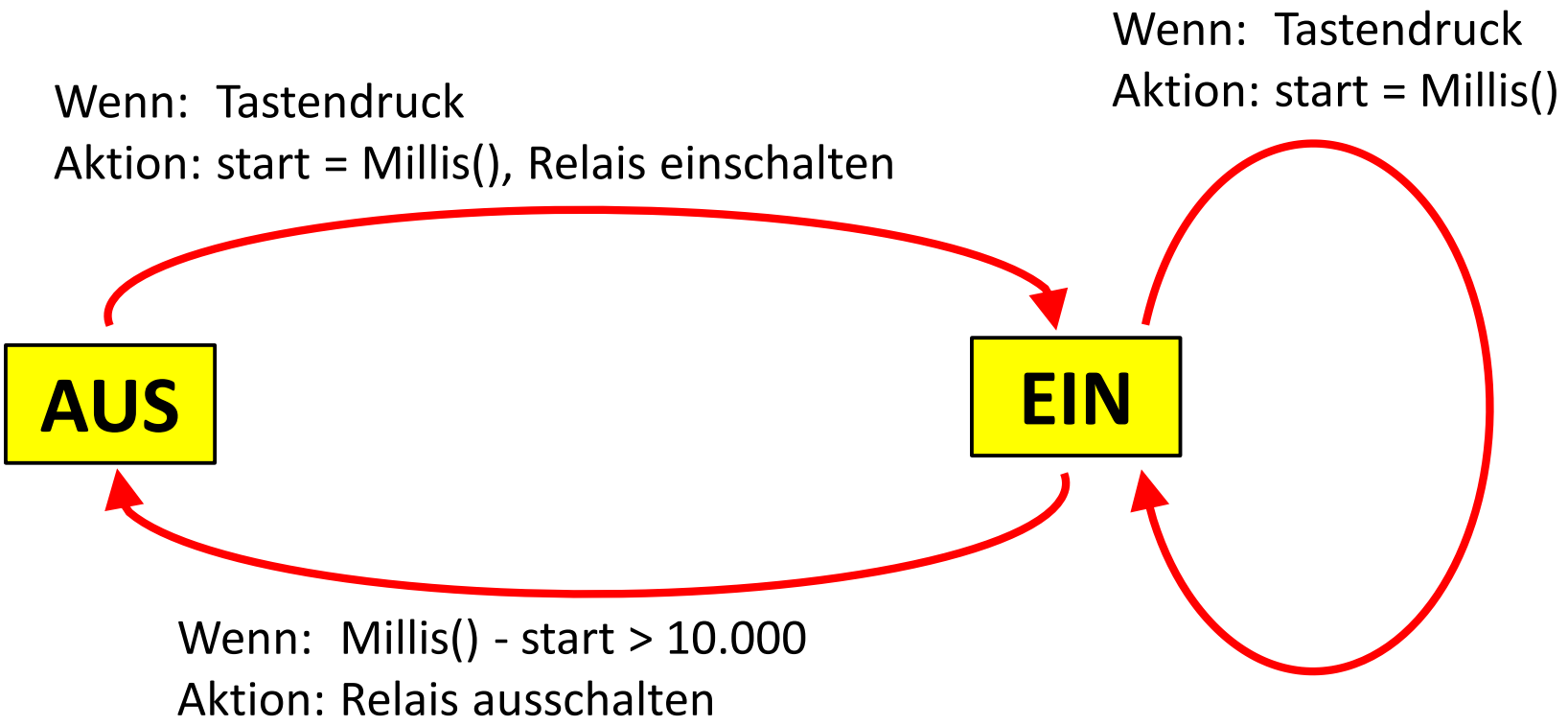
Finite-state machines can model a large number of problems, among which are electronic design automation, communication protocol design, language parsing and other engineering applications.

Quelle: [1]

Treppenlicht mit Funktion zum „Nachdrücken“ (a)

Aufgabe G1

Programmieren Sie nochmals ein automatisches Treppenlicht (vergl. letztes Praktikum, Ende von Teil F). Das Verhalten des Treppenlichts wird nun durch den folgenden endlichen Automat beschrieben:



Zusatzaufgabe: Kurz bevor das Treppenlicht ausgeht, blinkt es 5 Sekunden lang zur „Warnung“. Auch während des Blinkens kann bei Bedarf „nachgedrückt“ werden.

Treppenlicht mit Funktion zum „Nachdrücken“ (b)

Aufgabe G1

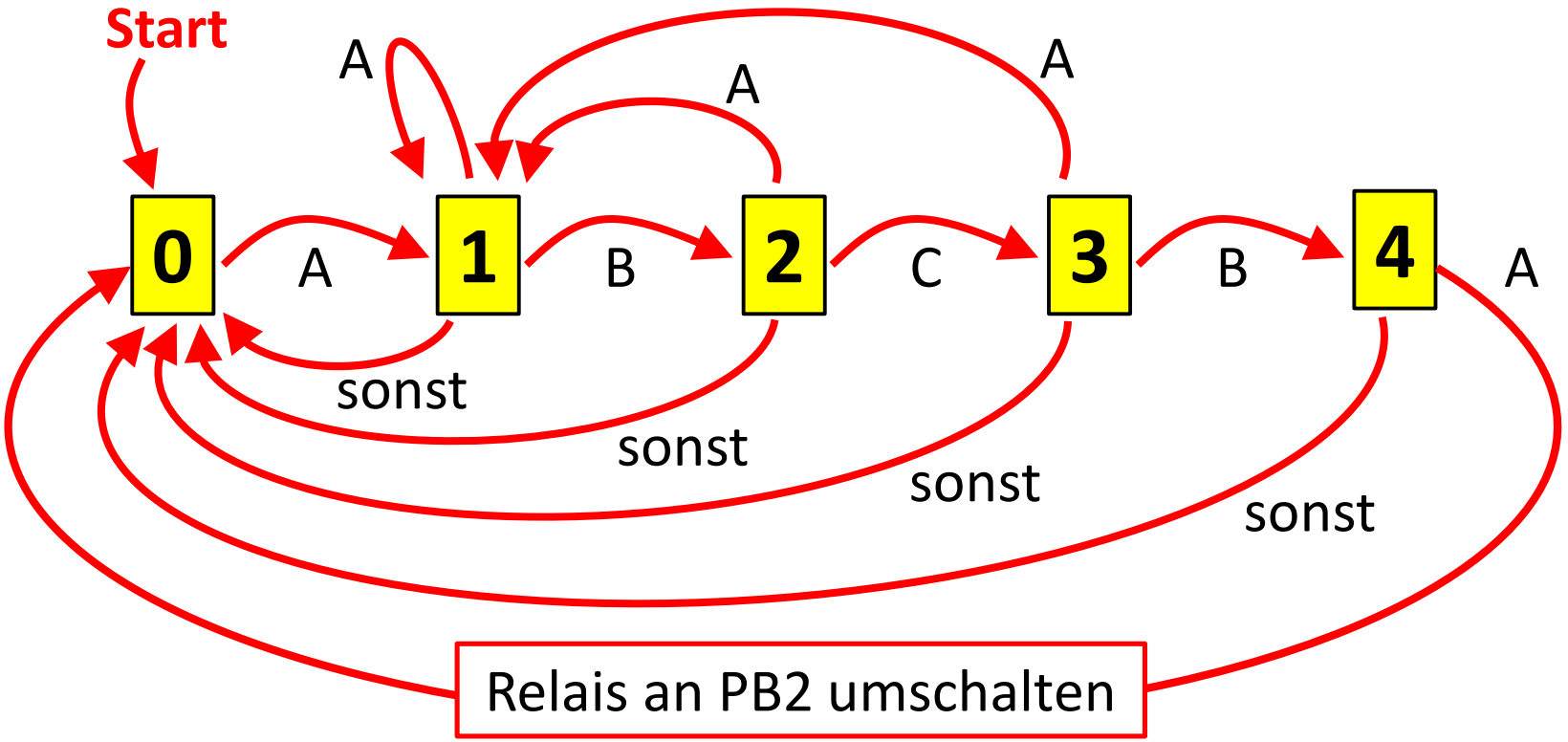
```
volatile uint32_t _millis_intern = 0;  // Millisekunden seit Programmstart

uint32_t Millis(void)  // Millisekunden seit Programmstart abfragen
{
    uint32_t result = 0;
    // Die folgende Zuweisung darf nicht unterbrochen werden!
    cli(); result = _millis_intern; sei();
    return result;
}

ISR(TIMER0_COMPA_vect)  // Timer-Interrupt: Millisekunden zählen
{
    _millis_intern++;
}

void init_timer(void)  // Timer initialisieren, Frequenz = 1 kHz
{
    TCCR0A = _BV(WGM01);              // CTC-Modus (siehe Tab. 19-9, S. 140)
    TCCR0B = _BV(CS01) + _BV(CS00);   // Prescaler = 64 (Tab. 19-10, S. 142)
    OCR0A  = 249;                     // Vergleichswert für Timer

    TIMSK0 = _BV(OCIE0A);             // Compare Match Interrupt (siehe S. 143)
    sei();                            // Interruptsystem des Controllers ein
}
```

Elektronisches Codeschloss

Aufgabe G2

Programmieren Sie ein elektronisches Codeschloss mit drei Tasten „A“, „B“ und „C“. Der Code zum Öffnen des Schlosses lautet „ABCBA“.

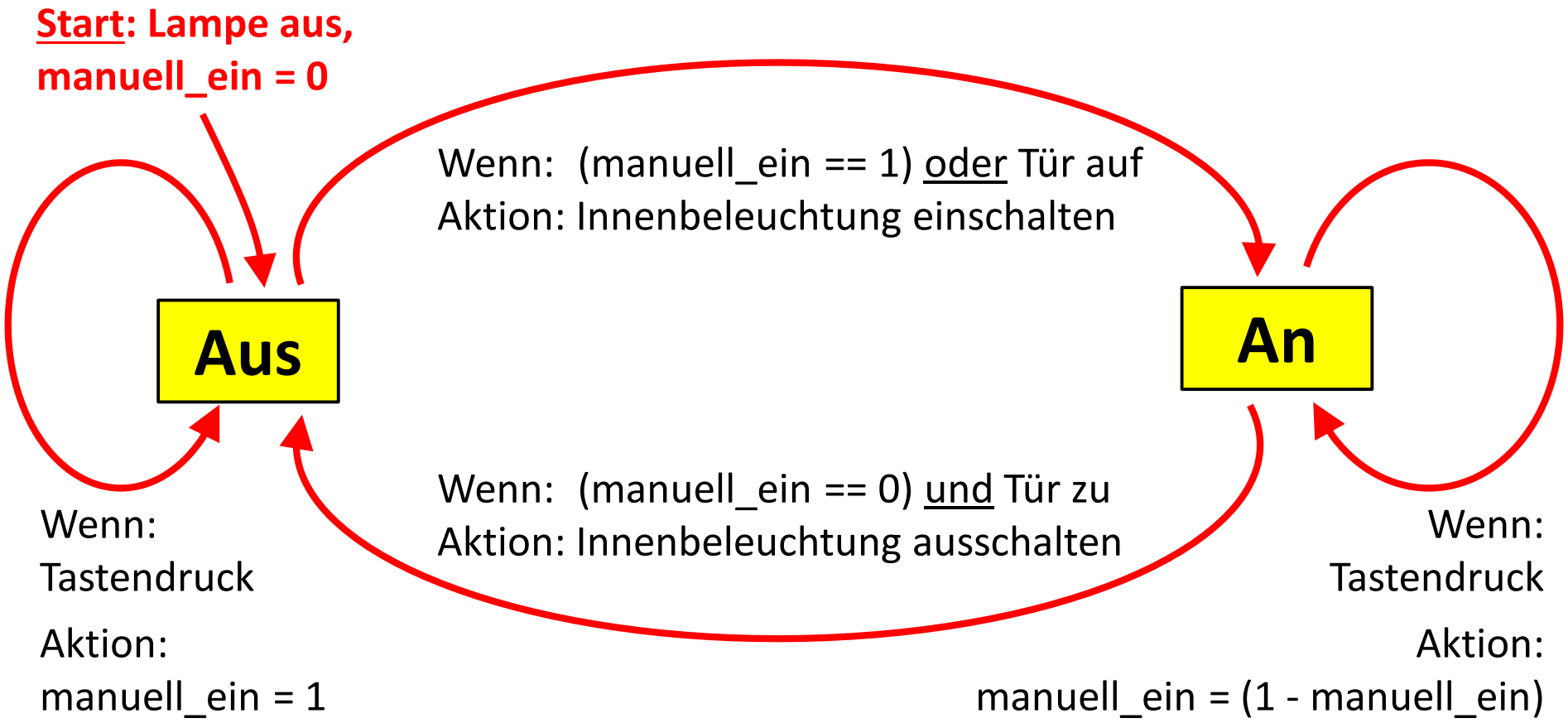
- Die Betätigung der Tasten „A“, „B“ und „C“ wird über die serielle Schnittstelle vom PC an den Mikrocontroller übermittelt.
- Nach Eingabe des korrekten Codes wird das Relais an PB2 umgeschaltet.



Kfz-Innenbeleuchtung

Aufgabe G3

Programmieren Sie ein Steuergerät für die Innenbeleuchtung eines Autos. Die Innenbeleuchtung reagiert auf die Stellung der Fahrzeugtür (regelbarer Widerstand an ADC0/PC0) und auf einen Taster direkt an der Leuchte (Anschluss PD2). Die Stromversorgung der Lampe geschieht über ein Relais (Anschluss PB2).



Quellenverzeichnis

[1] Engl. Wikipedia: „Finite-state machine“ (Stand: 05.04.2016)