Praktikum 04 - Polling und Interrupts

Malte Riechmann, André Kirsch

Beschaltung für den Taster

- Pulldown
- High-Active

Aufgabe 1

```
template <const uint8_t PORT_NB>
class TButton {
public:
  TButton() {
    pinMode(PORT_NB, INPUT);
  }

  unsigned int state() {
    return digitalRead(PORT_NB);
  }
};
```

Umsetzung der Klasse TButton

Das Programm hat eine Reaktionszeit von bis zu 2 Sekunden, da der Delay in der Main-Funktion auf zwei Sekunden gesetzt ist. Es wird also nur alle zwei Sekunden geprüft, ob der Button gedrückt ist.

Aufgabe 2

```
const uint32_t Delay = 500;
unsigned short count = 0;

void loop() {
    // if emergency stop, turn led off
    if (Button.state() == HIGH) {
        Led.off();
    } else if (count % 4 == 0) { //otherwise toggle
        Led.toggle();
        count = 0;
    }
    count++;
    // wait
    delay(Delay);
}
```

Wir verbessern die Reaktionszeit, indem wir das Delay verkürzen. Damit die LED weiterhin gleichlang blinkt, nutzen wir einen Counter.

Aufgabe 3

```
void setup() {
  Serial.begin(9600);
}

void loop() {
    // if emergency stop, turn led off
    if (Button.state() == HIGH) {
        Serial.println("Der Not-Aus-Knopf wurde gedrueckt");
        Serial.print("Laufzeit in Millisekunden: ");
        Serial.println(millis());
        Led.off();
    } else if (count % 4 == 0) { //otherwise toggle
        count = 0;
        Led.toggle();
    }
    count++;
    // wait
    delay(Delay);
}
```

Wir initialisieren den Serial Port mit einer Baudrate von 9600. Über Serial.println bzw print schicken wir dann die Nachrichten.

Aufgabe 4

```
void buttonInterruptHandler() {
    GPIOIntDisable(GPIO_PORTC_BASE, GPIO_PIN_4);
    Serial.println("Der Not-Aus-Knopf wurde gedrueckt");
    Serial.print("Laufzeit in Millisekunden: ");
    Serial.println(millis());
    Led.off();
}

void setup() {
  Serial.begin(9600);

  GPIOIntEnable(GPIO_PORTC_BASE, GPIO_PIN_4);
  GPIOIntRegister(GPIO_PORTC_BASE, buttonInterruptHandler);
}

void loop() {
    Led.toggle();

    // wait
    delay(Delay);

}
```

Um die Not-Aus-Funktionalität mit einem Interrupt zu realisieren haben wir zunächst den Pin, an dem der Button angeschlossen ist für Interrupts aktiviert. Anschließend haben wir die Funktion für den Interrupt registriert, so dass diese bei Auftreten der Interrupt ausgeführt wird. Um mehrfaches Ausführen der Funktion zu vermeiden haben wir den Pin am Anfang der Not-Aus-Funktion wieder deaktiviert.

Aufgabe 5

```
const uint32_t Delay = 200;

unsigned int state() {
    unsigned int value = digitalRead(PORT_NB);
    delay(Delay);
    return value;
}

void loop() {
    if (Button.state() == HIGH) {
       Led.toggle();
    }
}
```

In der Loop-Funktion fragen wir den Status des Schalters ab. Sollte dieser High sein, so rufen wir die toggle-Funktion der LED auf. Bei der Abfrage des Status warten wir danach 200 ms zur Entprellung des Tasters. So hat der Taster 200 ms Zeit, um Zurück in seine Ausgangsposition zu gehen.