FH Bielefeld
University of
Applied Sciences

Embedded Systems: Aufgabenblatt 4

SoSe 2018 Abgabe: 27.05.2018
Prof. M. König

Praktikum 4 (10 Punkte) - Energia: Polling und Interrupts

Die Abgabe der Lösungen (Beantwortung der Fragen und Quellcode) erfolgt im Ilias bis zum angegebenen Abgabezeitpunkt. Die Besprechung und Bewertung der **eingereichten** Lösungen erfolgt am folgenden Praktikumstermin.

Vorbereitung

Lesen Sie die Wikipedia-Einträge über
- Hardware Abstraction [http://en.wikipedia.org/wiki/Hardware_abstraction],
- Logikpegel “High-Active” und “Low-Active” [https://de.wikipedia.org/wiki/Logikpegel],
- Prellen [http://de.wikipedia.org/wiki/Prellen],
- Polling in der Informatik [http://de.wikipedia.org/wiki/Pollin_(Informatik)] und
- Interrupts [http://de.wikipedia.org/wiki/Interrupt] nach.

Lesen Sie ferner, wie mittels der Energia-Bibliothek Interrupts [http://energia.nu/AttachInterrupt.html] verwendet werden.

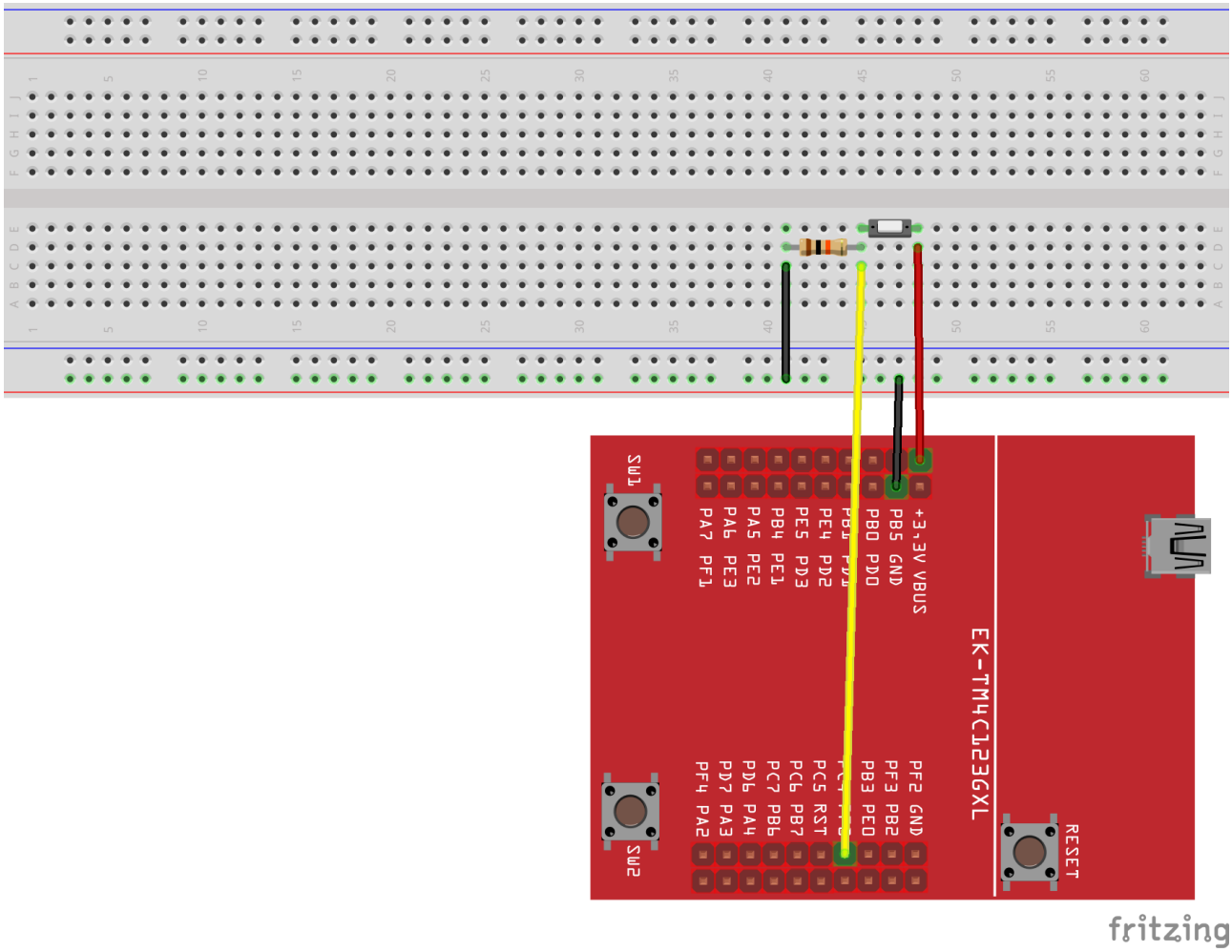
Sehen Sie Sich die Handhabung der Funktionen `pinMode, digitalWrite, digitalRead` sowie das Übertragen von Debugging Informationen mittels der `Serial` Klasse in der Energia-Umgebung an.

Praktikum

Speichern Sie das Listing 1 `button_loop.ino` (hinterlegt in ILIAS) lokal ab. Das Listing 1 enthält nur einen Teil eines Programms, das eine blinkende LED mittels eines Knopfdrucks auf den Taster ausschalten (Not-Aus) soll.

Schließen Sie das LaunchPad über die Pins PC4, GND, +3.3V an das Steckbrett gemäß Schaltbild 1 an (mit 10 kOhm Widerstand).

Welche Art der Beschaltung wurde für den Taster verwendet (Pull-Up oder Pull-Down und High-Active oder Low-Active)?



Schaltbild 1

Aufgabe 1 *(2 Punkte)*

Implementieren Sie die fehlende Klasse `TButton` *(Konstruktor und Funktion* `state`*).*

Laden Sie das Programm auf das LaunchPad hoch und testen Sie es. (Tipp: Das Programm können Sie mit dem Knopf „Reset“ auf dem LaunchPad neustarten.)

Wie verhält sich ein Auslösen des “Not-Aus-Knopfs” hinsichtlich der Reaktionszeit?

Aufgabe 2 *(2 Punkte)*

Verbessern Sie das Programm hinsichtlich der Reaktionszeit, indem Sie Polling einsetzen.

Aufgabe 3 *(1 Punkt)*

Geben Sie mittels der Serial-Klasse Debug-Informationen vom LaunchPad an Ihr System aus, wenn der Not-Aus-Knopf betätigt wird.

Aufgabe 4 *(3 Punkte)*

Implementieren Sie nun die Not-Aus-Funktionalität mittels Nutzung der TivaWare-ROM-Funktionen zur Auslösung eines Interrupts, d.h. ohne Energia-Funktion attachInterrupt() zu benutzen. Schreiben Sie hierzu ein neues Programm.
(Siehe TivaWare ™ Peripheral Driver Library. Dazu könnten Ihnen auch Lab3 und Lab4 des LaunchPad-Workbooks helfen.)

Aufgabe 5 *(2 Punkte)*

Schreiben Sie nun ein weiteres Programm, das die LED des LaunchPads über den Taster an- bzw. ausschaltet (toggeln). Stellen Sie dabei sicher, dass Ihre Implementierung der Klasse `TButton` *eine Entprellung des Tasters vornimmt.*

```
// Demo illustrating blinking led and reading of button.
// Led is finally turned off when button is pressed.

// define port for led output
const uint8_t LedPortOut = RED_LED;
// define pin for button input
const uint8_t ButtonPinIn = PC_4;
// define delay for blinking in ms
const uint32_t Delay = 2000;

//! LED handling class. Has disable() function for emergency stop.
//! Parameter (in): PORT_NB (output port for connected led)
template <const uint8_t PORT_NB>
class TLed {
public:
    //! Constructor takes state (HIGH, LOW) only if given.
    //! Defaults: value for state = LOW, and is not disabled.
    TLed(const uint8_t f_ledState = LOW)
    : m_ledState(f_ledState), m_disabled(false) {
        pinMode(PORT_NB, OUTPUT); // led is always output
        digitalWrite(PORT_NB, m_ledState); // set led to default state
    }
    //! If this led is disable, nothing happens, otherwise
    //! toggles state of led (from HIGH to LOW or from LOW to HIGH).
    void toggle() {
        if (m_disabled) // somehow no longer active
            return;
        if (m_ledState == LOW) { // toggle state
            m_ledState = HIGH;
        }
        else {
            m_ledState = LOW;
        }
        digitalWrite(PORT_NB, m_ledState); // set led to current state
    }
    //! Turn led finally off (emergency stop), state is set LOW, functionality off.
    void off() {
        m_disabled = true;
        m_ledState = LOW;
        digitalWrite(PORT_NB, m_ledState); // set led to current state
    }
private:
    uint8_t m_ledState; // current state of led
    bool m_disabled; // disable flag (on if led is finally turned off)
};

....TODO: INSERT CODE FOR CLASS TBUTTON....

// global instances for led output
TLed<LedPortOut> Led;
// and for button pin
TButton<ButtonPinIn> Button;

void setup() {}

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

Listing 1: `button_loop.ino`