FH Bielefeld
University of
Applied Sciences

SoSe 2015
Prof. M. König

Abgabe: KW 19 2015

Aufgabe 3 (10 Punkte) - Arduino: Polling und Interrupts

Vorbereitung

Lesen Sie die Wikipedia-Einträge über
- Hardware Abstraction [http://en.wikipedia.org/wiki/Hardware_abstraction],
- Prellen [http://de.wikipedia.org/wiki/Prellen],
- Polling in der Informatik [http://de.wikipedia.org/wiki/Polling_(Informatik)] und
- Interrupts [http://de.wikipedia.org/wiki/Interrupt] nach.

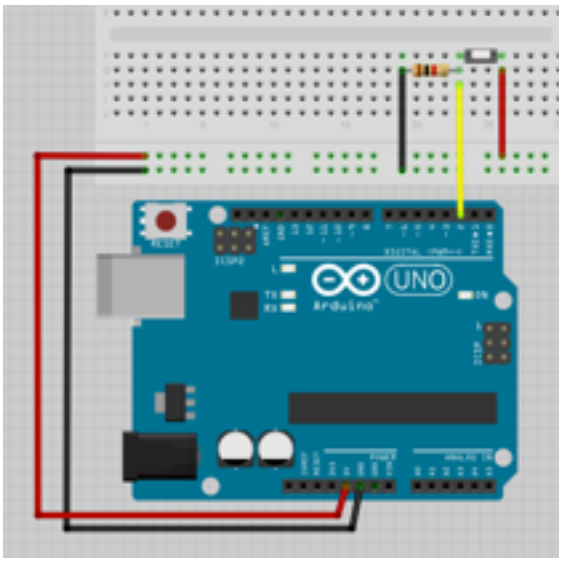
Lesen Sie ferner, wie mittels der Arduino-Bibliothek Interrupts [http://www.arduino.cc/en/Reference/AttachInterrupt] verwendet werden, und das Kapitel 12 der ATmega Dokumentation [http://www.atmel.com/Images/doc8161.pdf].

Sehen Sie Sich die Handhabung der Funktionen `pinMode`, `digitalWrite`, `digitalRead` sowie das Übertragen von Debugging Informationen mittels der `Serial` Klasse in der Arduino 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 Arduino Board über die Pins 2, Gnd, 5V an das Steckbrett gemäß Schaltbild 1 an (mit 10 kOhm Widerstand).



Schaltbild 1

Aufgabe 2.1 *(2 Punkte)*

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

Laden Sie das Programm auf das Arduino Board hoch und testen Sie es. Hinweis, das Programm können Sie mit dem Knopf auf dem Arduino Board neustarten.

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

Aufgabe 2.2 *(2 Punkte)*

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

Aufgabe 2.3 *(1 Punkt)*

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

Aufgabe 2.4 *(3 Punkte)*

Implementieren Sie nun die Not-Aus-Funktionalität mittels direkter Nutzung der ATmega Interrupt-Funktionen, d.h. ohne die Arduino Interrupt Schnittstelle. Schreiben Sie hierzu ein neues Programm. Vollziehen Sie dabei genau nach, wieso Sie welche Register geändert haben, indem Sie in das Kapitel 12 der ATmega Dokumentation heranziehen [http://www.atmel.com/Images/doc8161.pdf]*. Weitere Informationen findet Sie auch unter http://www.atmel.com/webdoc/AVRLibcReferenceManual/group__avr__interrupts.html.*

Aufgabe 2.5 *(2 Punkte)*

Schreiben Sie nun ein weiteres Programm, dass die LED des Arduino Uno Boards über den Knopf an- bzw. ausschaltet. 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 = 13;
// define pin for button input
const uint8_t ButtonPinIn = 2;
// 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`