Hochschule **Rosenheim**
University of Applied Sciences

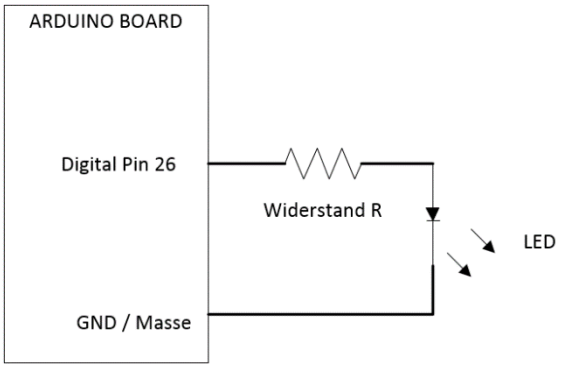
Fakultät für Informatik **Probeklausur Embedded Systems (ESy)**	*(Name, Vorname)* *(Matrikelnummer)*
	(Erstkorrektor) *(Zweitkorrektor)*

Hinweise:

- Zur Lösung der Aufgaben benötigtes Dokumentationsmaterial wird in einem **gesonderten Anhang** zur Verfügung gestellt. …
- …
- Soweit nicht anders erwähnt, beziehen sich die Aufgabenstellungen auf den **ATmega2560** bzw. das entsprechende Arduino Mega Board.
- Auf die Angabe von `#include` Statements wird grundsätzlich verzichtet, Sie müssen diese selbst auch nicht angeben.
- Makros zur Auflösung der Registernamen und Registerinhalte dürfen wie in den Übungen verwendet werden (Beispiel: `DDRA`).

Aufgabe 1: Digitale Ausgabe, LEDs

Gegeben sei die abgebildete Schaltung. Ein Arduino Mega bzw. ein ATmega2560 steuert eine LED über den digitalen Pin 26 an.



a) Der Flussstrom für die verwendete gelbe LED soll $I_F = 20mA$ betragen, die Flussspannung $U_F = 2{,}2V$. Wie groß muss der Widerstand *R* in Ohm *exakt* sein, falls der Mikrocontroller über Digital Pin 26 eine konstante Spannung von *5V* liefert?

b) Schreiben Sie ein Mikrocontroller-Programm, das jede Sekunde den Zustand (AN/AUS) der LED **ändert**. Beachten Sie:

- Es dürfen **keine** Funktionen der Arduino Bibliothek verwendet werden. Ausnahme ist die Funktion `delay(unsigned long v)`, die den Programmablauf um *v* Millisekunden verzögert.
- Digital Pin 26 entspricht beim ATmega2560 dem Pin PA4 des Ports A. Konfigurieren Sie die benötigten Register `DDRx, PINx, PORTx`.

```
void setup() {



}

void loop() {









}
```

c) Gegeben sei eine vorzeichenlose 8-Bit Variable `uint8_t var`. Setzen Sie die folgenden Modifikationen je durch eine **einzige C-Anweisung** um (kein Strichpunkt **in** einer Anweisung!)

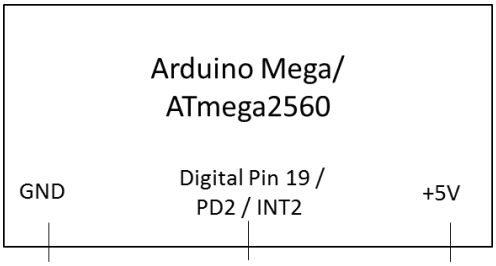
- Lösche **nur** das höchstwertigste Bit von `var` (Löschen bedeutet: auf 0 setzen):

- Änderes **nur** das höchstwertigste Bit von `var` auf 1:

- Dividiere den Wert von `var` durch 2 (kein „/“ oder „÷“ oder „· " erlaubt!):

Aufgabe 2: Interrupts

a) Am Arduino Mega soll ein Taster angeschlossen werden. Der eingelesene Zustand bei **geschlossenem** Taster soll **HIGH** sein, der eingelesene Zustand bei **geöffnetem** Taster soll **LOW** sein. Gehen Sie davon aus, dass keinerlei interne Pull-up bzw. Pull-Down Widerstände vorhanden sind.
Zeichnen Sie eine entsprechende Schaltung! Verwenden Sie die bereits vorgegebene Abbildung!



b) Das Drücken des Tasters aus a) soll über den **externen Interrupt `INT2 (PD2 / INT2)`** erkannt werden. Verwenden Sie das vorgegebene Codegerüst und schreiben Sie ein Mikrocontrollerprogramm für den ATmega2560, das folgenden Anforderungen genügt:
   - Bei Tastendruck: Ausgabe von „Hallo“ über die serielle Konsole (beliebige Baudrate).
   - Bei **steigender Flanke**: Auslösen des externen Interrupts `INT2`.
   - Entprellung: Ausgabe von „Hallo“ nur in ISR, falls letzter Interrupt mehr als 100 ms zurückliegt.
   - Funktionen der Arduino-Library sind **verboten**! **Ausnahme:** `Serial.begin(<baudrate>)`, `Serial.println(<string>)` und `millis()`, das die Anzahl von Millisekunden seit dem Programmstart als `unsigned long` zurückgibt.
   - Beachten Sie Auszüge aus dem ATmega2560 Handbuch im Anhang!

```
void setup() {




    sei();    // globally activate interrupts
}

void loop() {


}

ISR (INT2_vect) {




}
```

Aufgabe 3: Timer

a) Gegeben sei ein 8-Bit Timer, der mit 2 MHz getaktet ist. Der Takt kann durch eine Prescaler-Einheit mit den Faktoren 1, 8, 64, 256 und 1024 skaliert werden. Für eine Anwendung ist eine Auflösung von **mindestens** 1 ms gefordert.

- Welchen Prescaler müssen Sie wählen, falls Sie möglichst lange Zeitperioden abdecken möchten, ohne dass ein Overflow-Ereignis auftritt?
- Wie lange dauert es bei Ihrer Wahl bis ein Overflow Ereignis eintritt?

Begründen Sie jeweils Ihre Antwort! Der Weg wie Sie zum Ergebnis kommen muss erkennbar sein.

b) Beschreiben Sie jeweils in **maximal 1 bis 2 Sätzen**, was man unter *Input Capture* und *Compa*re versteht!

Aufgabe 4: A/D Umsetzung

Über den analogen Eingang **ADC1 (Pin PF1 des Ports F)** wird die Ausgangsspannung des Sensors HIH 5030 zur **Bestimmung der relativen Luftfeuchte** (Angabe in **%**) eingelesen. Annahme: Beim Betrieb des Sensors besteht ein **linearer Zusammenhang** zwischen der Ausgangsspannung des Sensors und der gemessenen relativen Luftfeuchte. Konkret ergibt sich:

- Bei 0 % relativer Luftfeuchte: 0,00 V am Pin ADC1
- Bei 100 % relativer Luftfeuchte: 3,94 V am Pin ADC1

Im Folgenden dürfen ungerade Werte stets sinnvoll gerundet werden.

a) Als Referenzspannung AREF für die A/D Umsetzung stehen die Werte 1,1 *V* und 2,56 *V* und 5 *V* zur Verfügung. Welchen Wert wählen Sie? Kurze Begründung!

b) Welchen vorzeichenlosen Integer-Wert liest eine Mikrocontroller-Anwendung bei 0% rel. Luftfeuchte und bei 100% rel. Luftfeuchte nach einer A/D Umsetzung aus dem Ergebnisregister ADC aus? Annahme: Der ATmega2560 verwendet einen 10-Bit A/D Wandler im *Single-Ended Mode* („Normalmodus“) und verwendet die Referenzspannung AREF aus a).

c) Das Ergebnisregister ADC wird nach jeder A/D Umsetzung ausgelesen. Stellen Sie eine **Formel** auf, so dass Sie direkt aus dem (vorzeichenlosen) Wert des Registers auf den Wert für die relative Luftfeuchte schließen können!

(Fortsetzung nächste Seite)

d) Ergänzen Sie das folgende Programm, so dass in einer Endlosschleife der Wert des Sensors ausgelesen wird und die relative Luftfeuchte **in Prozent** über die Konsole ausgegeben wird.
   - Falls Sie bei Aufgabe c) kein Ergebnis berechnet haben: Machen Sie eine beliebige Annahme für die Umrechnungsformel!
   - Der analoge Wert wird über den **Pin ADC1 (PF1)** eingelesen. Dazu muss im Register `ADMUX` das niederwertigste Bit `MUX0` auf 1 gesetzt werden!
   - Als **Referenzspannung** wird `AVCC` verwendet. Ein externer Kondensator ist angeschlossen.
   - Es wird der *Single Conversion Mode* verwendet: Man startet jede A/D Umsetzung manuell; die Anwendung wartet/blockiert bis die Umsetzung beendet ist.
   - Beachten Sie die Auszüge aus dem ATmega2560 Handbuch im Anhang!

```
void setup() {

	Serial.begin(9600);  // activate serial console

	// configure an adequate prescaler for the ADC
	ADCSRA |= (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0);

	// TODO: enable ADC, select input, set reference voltage









}

void loop() {

	double result = 0.0;	// will store result in % rel. Luftfeuchte

	// TODO:  trigger ADC conversion and wait until conversion is finished









	// TODO read result of conversion and put % rel. Luftfeuchte in variable result









	Serial.print(result);
}
```

Aufgabe 5: Kommunikationsschnittstellen, SPI

a) Geben Sie für die folgenden Kommunikationsschnittstellen jeweils an, ob sie *asynchron bzw. synchron* oder *single-ended bzw. differential* sind.

	asynchron/synchron?	**Single-ended / differential**
I²C		
SPI		
UART		

b) Gegeben sei das folgende (korrekte) ATmega2560 Programm zum Senden von Daten über die SPI Schnittstelle. Es handelt sich um den **SPI Master**. Modifizieren Sie das Programm, so dass der Master **gleichzeitig auch ein Zeichen empfangen kann**.

- **Nur** die Funktion `spi_transceive()` und `loop()` muss **leicht modifiziert** werden.
- Das Programm muss **lauffähig** sein.
- Das empfangene Zeichen soll in der Variable `received` abgelegt werden.
- Beachten Sie ggfs. den Auszug aus dem ATmega2560 Handbuch im Anhang!

```
void setup() {
	Serial.begin(9600);
	DDRB = (1<<DDB2) | (1<<DDB1); // MOSI and SCK as output

	// Enable SPI, set as master, set clock rate to fck/128
	SPCR = (1<<SPE)|(1<<MSTR)|(1<<SPR0)|(1<<SPR1);
 }

void spi_transceive(unsigned char data) {


	 SPDR = data;                  // load data to be sent into SPI data register (SPDR)


	 while(!(SPSR & (1<<SPIF)));   // wait until transmission completes


 }

 void loop() {
	char text[] = "Hallo Slave!";
	for (int i = 0; i < sizeof(text); i++)  {
		char received = ' ';


		spi_transceive(text[i]);



		Serial.print(received);
	}
	delay(1000);
}
```

Aufgabe 6: Pulsweitenmodulation, Bootloader

a) Zeichnen Sie qualitativ die Zeitverläufe der folgenden PWM Signale: Ein Signal habe einen Duty Cycle von 50%, das andere Signal einen Duty Cycle von 25%! Beide Signale sollen die gleiche Periodendauer haben.

b) Erklären Sie in wenigen Sätzen die Aufgabe und die Funktionsweise eines Bootloaders am Beispiel des Arduino Mega bzw. ATmega2560?
Gehen Sie darauf ein wie das Umschalten zwischen normaler Anwendung und Bootloader funktioniert!

c) Wie kann man einen zerstörten Bootloader beim Arduino Mega bzw. ATmega2560 wiederherstellen?