Hochschule **Rosenheim**
University of Applied Sciences

Fakultät für Informatik (Lösung) **Prüfung: Embedded Systems (ESy)**	*(Name, Vorname)* *(Matrikelnummer)*
Erreichte Punktzahl und Gesamtnote: **Gesamtnote:**	*(Erstkorrektor)* *(Zweitkorrektor)*

Aufgabe 1: Digitale Ausgabe, LEDs

a) $R = \frac{5V-2,2V}{20mA} = 140\Omega$

b)

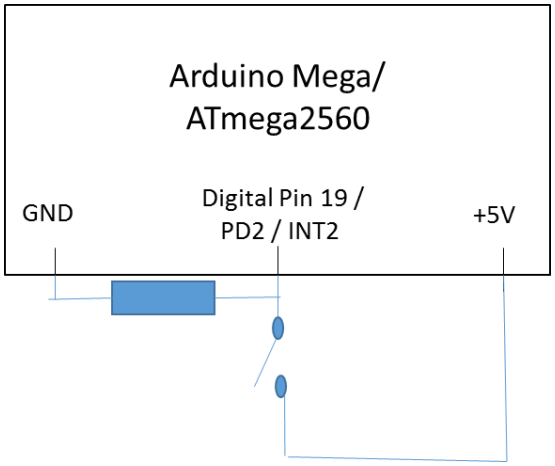
```
void setup() {
    DDRA |= (1 << DDA4);        //configure output
}

void loop() {
    PORTA = PORTA ^ (1 << PA4);        // toggle
    delay(1000);

}
```

c)

- `var &= ~(1 << 7);`
- `var |= (1 << 7);`
- `var = var >> 1;`

Aufgabe 2: Interrupts

a)



b)

```
volatile unsigned long timeOfPreviousRisingEdge = 0;  // used for debouncing

void setup()
{
	Serial.begin(9600);
	EIMSK |= (1 << INT2);                  // turn on INT2
	EICRA |= (1 << ISC20) | (1 << ISC21);// trigger an any rising edge (
	sei();                                 // globally activate interrupts
}

void loop()
{

}


ISR (INT2_vect)
{
	if (millis() - timeOfPreviousRisingEdge > 100) {
		Serial.println("Hallo");
	}
	timeOfPreviousRisingEdge = millis();

}
```

Aufgabe 3: Timer

a) Man rechnet sich am besten aus, welche Auslösung man mit den verschiedenen Prescaler erreicht.
Zeit für 1 Tick: $\frac{1}{2MHz} \cdot x$, falls *x* der Prescaler-Wert ist.

-

Prescaler	1	8	64	256	1024
Auflösung	500 ns	4 us	32 us	128 us	512 us

Man erkennt, dass man sogar einen 1024 Prescaler verwenden darf , da die Auflösung passt. Da dauert es auch am längsten bis ein Overflow eintritt.

Es dauert $\frac{256}{\frac{2}{1024}MHz} = 131\,ms$ bis ein Overflow eingetreten ist.

b) *Input Capture***:** Bei Eintreten eines bestimmten Ereignisses wird der aktuelle Wert des Timers in einem Register festgehalten, z.B. Messen von Zeitintervallen.
Output Compare: Bei einem bestimmten Zählerwert wird automatisch durch Hardware ein bestimmtes Ereignis ausgelöst (z.B. Toggle einer Leitung)

Aufgabe 4: A/D Umsetzung

a) Man muss als Referenzspannung AREF den Wert 5V wählen. Der Grund ist, dass die maximale Ausgangsspannung bei 100%RH bereits 3,94 V ist und die Referenzspannung immer größer sein muss, als der maximale Wert.

b) Es gilt die Formel $ADC = \frac{V_{In} \cdot 1024}{V_{ref}}$.

Damit ergibt sich:

- Bei 0%RH: 0
- Bei 100%RH: 807

c) Man nutzt den linearen Zusammenhang aus. y = mx + t (wobei y %RH und x der Integerwert)
für Ansatz lineare Interpolation
Da bei 0%RH der Wert von result gleich 0 ist, weiß man sofort, dass t = 0.
Es gilt also: y = mx
Einsetzen von y = 100 %RH und x = 807 ergibt: 100 %RH = m * 807
-> m = 0,124 %RH]
Der Zusammenhang lautet also: y = 0,124 * x

d)

```
void setup()
{
	// activate serial console
	Serial.begin(9600);

	// enable ADC functionality
	ADCSRA |= (1 << ADEN);

	// use /128 prescaler (ADC requires 50 kHz to 200 kHz, see manual p271, but system
clock is 16 MHz)
	ADCSRA |= (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0);

	// select ADC0 as input pin;
	ADMUX |= (1 << MUX0);

	// use reference voltage 5V (Note: AVCC is AREF), manual, p281
	ADMUX |= (1 << REFS0);
}

void loop()
{
	// trigger ADC conversion
	ADCSRA |= (1 << ADSC);

	// wait until conversion is finished, see manual p286
	while (ADCSRA & (1 << ADSC));

	// read analog value, first LOW then HIGH register
	unsigned int read = ADCL + 256 * ADCH;

	double result = 0.12 * read;
	Serial.print(result);
	Serial.println(" Volt");

}
```

Aufgabe 5: Kommunikationsschnittstellen, SPI

a)

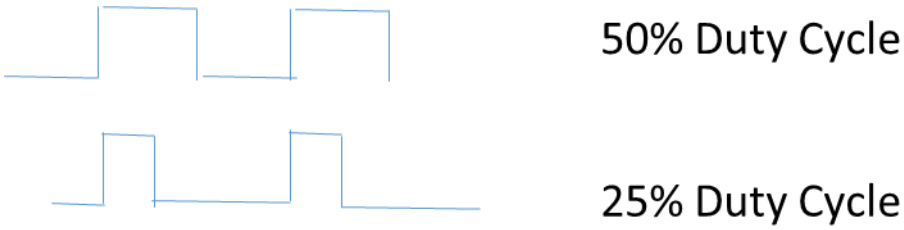
	asynchron/synchron?	**Single-ended / differential**
I2C	Synchron	Single-ended
SPI	Synchron	Single-ended
UART	Asynchron	Single-ended

b)

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

       // MOSI and SCK to output, all others as input
       DDRB = (1<<DDB2) | (1<<DDB1); //alternative command

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

 }

 // send and receive data
  für Anpassung des Rückgabetyps)
  unsigned char spi_transceive(unsigned char data) {

        // load data to be sent into buffer
        SPDR = data;
        // wait until transmission completes
        while(!(SPSR & (1<<SPIF)));

        // return received data
        return(SPDR);


 }

 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)



b) Folgendes könnte in Antwort enthalten sein:

- SW Download
- über serielle Schnittstelle
- Bootloader in getrenntem, geschützten Speicherbereich
- Bootloader wird z.B. immer nach Reset ausgeführt
- Falls dann keine Dateien kommen, Übergabe an Anwendung

c) Man muss In-System Programming z.B. über SPI verwenden.