VUT FEKT

[BPA-DE2] Digital Electronics 2

Assignment 4

GitHub: *https://github.com/ShalaKreshnik*

Name and Surname: *Kreshnik Shala*

Person ID: *226108*

Date: Tuesday, October 20, 2020

1 Table of Contents

Lab assignment 4

VUT FEKT

[BPA-DE2] Digital Electronics 2

Assignment 4

GitHub: *https://github.com/ShalaKreshnik*

Name and Surname: *Kreshnik Shala*

Person ID: *226108*

Date: Tuesday, October 20, 2020

1.1 Preparation tasks. Submit:

1.1.1 Table with overflow times:

Module:	Number of bits:	1	8	32	64	128	256	1024
Timer/Counter0:	8	16u	128u	--	1024u	--	4096u	16384u
Timer/Counter1:	16	4096u	32768u	--	262144u	--	1048576u	4194304u
Timer/Counter2:	8	16u	128u	512u	1024u	2018u	4096u	16384u

1.2 Timer library. Submit:

1.2.1 Listing of library header file timer.h:

```
/*////////////////////////////////////////////////////////////////////////////////////////
///                                                                                      ///
/// VUT FEKT                                    Name and Surname: Kreshnik Shala         ///
/// [BPA-DE2] Digital Electronics 2             Person ID: 226108                        ///
/// Date: Tuesday, October 20, 2020                                                      ///
/// GitHub: https://github.com/ShalaKreshnik                                             ///
///                                                                                      ///
*////////////////////////////////////////////////////////////////////////////////////////
#ifndef TIMER_H
#define TIMER_H


/* Includes ----------------------------------------------------------*/
#include <avr/io.h>

/* Defines -----------------------------------------------------------*/
/*
 * @brief Defines prescaler CPU frequency values for Timer/Counter0.
 * @note  F_CPU = 16 MHz
 */
#define  TIM0_stop()              TCCR0B &= ~((1<<CS02) | (1<<CS01) | (1<<CS00));
#define TIM0_overflow_16us()     TCCR0B &= ~((1<<CS02) | (1<<CS01)); TCCR0B |= (1<<CS00);
#define TIM0_overflow_128us()    TCCR0B &= ~((1<<CS02) | (1<<CS00)); TCCR0B |= (1<<CS01);
#define TIM0_overflow_1ms()      TCCR0B &= ~(1<<CS02); TCCR0B |= (1<<CS01) | (1<<CS00);
#define TIM0_overflow_4ms()      TCCR0B &= ~((1<<CS01) | (1<<CS00)); TCCR0B |= (1<<CS02);
#define TIM0_overflow_16ms()     TCCR0B &= ~(1<<CS01); TCCR0B |= (1<<CS02) | (1<<CS00);
```

Name and Surname: *Kreshnik Shala*

Person ID: *226108*

Date: Tuesday, October 20, 2020

```
/**
 * @brief Defines interrupt enable/disable modes for Timer/Counter0.
 */
#define TIM0_overflow_interrupt_enable()    TIMSK0 |= (1<<TOIE0);
#define TIM0_overflow_interrupt_disable()   TIMSK0 &= ~(1<<TOIE0);


/**
 * @brief Defines prescaler CPU frequency values for Timer/Counter1.
 * @note  F_CPU = 16 MHz
 */
#define TIM1_stop()             TCCR1B &= ~((1<<CS12) | (1<<CS11) | (1<<CS10));
#define TIM1_overflow_4ms()     TCCR1B &= ~((1<<CS12) | (1<<CS11)); TCCR1B |= (1<<CS10);
#define TIM1_overflow_33ms()    TCCR1B &= ~((1<<CS12) | (1<<CS10)); TCCR1B |= (1<<CS11);
#define TIM1_overflow_262ms()   TCCR1B &= ~(1<<CS12); TCCR1B |= (1<<CS11) | (1<<CS10);
#define TIM1_overflow_1s()      TCCR1B &= ~((1<<CS11) | (1<<CS10)); TCCR1B |= (1<<CS12);
#define TIM1_overflow_4s()      TCCR1B &= ~(1<<CS11); TCCR1B |= (1<<CS12) | (1<<CS10);

/*
 * @brief Defines interrupt enable/disable modes for Timer/Counter1.
 */
#define TIM1_overflow_interrupt_enable()    TIMSK1 |= (1<<TOIE1);
#define TIM1_overflow_interrupt_disable()   TIMSK1 &= ~(1<<TOIE1);


/*
 * @brief Defines prescaler CPU frequency values for Timer/Counter2.
 * @note  F_CPU = 16 MHz
 */
#define  TIM2_stop()               TCCR2B &= ~((1<<CS02) | (1<<CS01) | (1<<CS00));
#define TIM2_overflow_16us()     TCCR2B &= ~((1<<CS02) | (1<<CS01)); TCCR2B |= (1<<CS00);
#define TIM2_overflow_128us()    TCCR2B &= ~((1<<CS02) | (1<<CS00)); TCCR2B |= (1<<CS01);
#define TIM2_overflow_512us()    TCCR2B &= ~(1<<CS02); TCCR2B |= (1<<CS01) | (1<<CS00);
#define TIM2_overflow_1ms()      TCCR2B &= ~((1<<CS01) | (1<<CS00)); TCCR2B |= (1<<CS02);
#define TIM2_overflow_2ms()      TCCR2B &= ~(1<<CS01); TCCR2B |= (1<<CS02) | (1<<CS00);
#define TIM2_overflow_4ms()      TCCR2B &= ~(1<<CS20); TCCR2B |= (1<<CS22) | (1<<CS21);
#define TIM2_overflow_16ms()     TCCR2B |= (1<<CS22) | (1<<CS21) | (1<<CS20);


/**
 * @brief Defines interrupt enable/disable modes for Timer/Counter2.
 */
#define TIM2_overflow_interrupt_enable()    TIMSK2 |= (1<<TOIE2);
#define TIM2_overflow_interrupt_disable()   TIMSK2 &= ~(1<<TOIE2);



#endif
```

Name and Surname: *Kreshnik Shala*

Person ID: *226108*

Date: Tuesday, October 20, 2020

1.2.2 Table with ATmega328P selected interrupt sources:

Program address:	Source:	Vector name:	Description:
0x0000	RESET	--	Reset of the system
0x0002	INT0	INT0_vect	External interrupt request number 0
0x0004	INT1	INT1_vect	External Interrupt Request number 1
0x0006	PCINT0	PCINT0_vect	Pin Change Interrupt Request number 0
0x0008	PCINT1	PCINT1_vect	Pin Change Interrupt Request number 1
0x000A	PCINT2	PCINT2_vect	Pin Change Interrupt Request number 2
0x000C	WDT	WDT_vect	Watchdog Time-out Interrupt
0x0012	TIMER2_OVF	TIMER2_OVF_vect	Overflow of Timer/Counter2 value
0x0018	TIMER1_COMPB	TIMER1_COMPB_vect	Compare match between Timer/Counter1 value and channel B compare value
0x001A	TIMER1_OVF	TIMER1_OVF_vect	Overflow of Timer/Counter1 value
0x0020	TIMER0_OVF	TIMER0_OVF_vect	Overflow of Timer/Counter0 value
0x0024	USART_RX	USART_RX_vect	USART Rx Complete
0x002A	ADC	ADC_vect	ADC Conversion Complete
0x0030	TWI	TWI_vect	2-wire Serial Interface

Module:	Operation:	I/O register(s):	Bit(s)
Timer/Counter0:	Perscaler	TCCR0B	CS02, CS01, CS00 (000: C. stopped, 001: 1, 010: 8, 011: 64, 100: 256, 101: 1024)
	8-bit data value	TCNT0	TCNT0 [7:0]
	Overflow interrupt enable	TIMSK0	TOIE0 (1: enable, 0: disable)
Timer/Counter1:	Perscaler	TCCR1B	CS12, CS11, CS10 (000: stopped, 001: 1,010: 8, 011: 64, 100: 256, 101: 1024)
	16-bit data value	TCNT1HT, TCNT1L	TCNT1 [15:0]
	Overflow interrupt enable	TIMSK1	TOIE1 (1: enable, 0: disable)
Timer/Counter2:	Perscaler	TCCR2B	CS22, CS21, CS20 (000: C. stopped, 001: 1, 010: 8, 011: 32, 100: 64, 101: 128…)
	8-bit data value	TCNT2	TCNT2 [7:0]
	Overflow interrupt enable	TIMSK2	TOIE2 (1: enable, 0: disable)

1.2.3 Listing of final application main.c:

```
/*////////////////////////////////////////////////////////////////////////////////
///                                                                             ///
/// VUT FEKT                                  Name and Surname: Kreshnik Shala  ///
/// [BPA-DE2] Digital Electronics 2           Person ID: 226108                 ///
/// Date: Tuesday, October 20, 2020                                             ///
/// GitHub: https://github.com/ShalaKreshnik                                    ///
///                                                                             ///
*////////////////////////////////////////////////////////////////////////////////


/* Defines -----------------------------------------------------------*/
#define LED_D1  PB5
#define LED_D2  PB4
#define LED_D3  PB3
#define LED_D4  PB2
#define BTN     PD0

/* Includes ----------------------------------------------------------*/

#include <avr/io.h>         // AVR device-specific IO definitions
#include <avr/interrupt.h>  // Interrupts standard C library for AVR-GCC

#include "gpio.h"           // GPIO library for AVR-GCC
#include "timer.h"          // Timer library for AVR-GCC

uint8_t arr[] = {LED_D1,LED_D2,LED_D3,LED_D4};
uint8_t count =0;
uint8_t last_count = -1;
uint8_t back = 0;
/* Function definitions ----------------------------------------------*/
/**
 * Main function where the program execution begins. Toggle one LED
 * on the Multi-function shield using the internal 8- or 16-bit
 * Timer/Counter.
 */

int main(void)
{
    /* Configuration of LED(s) */

       //MAKING LED_D2 AS AN OUTPUT PIN
    GPIO_config_output(&DDRB, LED_D2);


    GPIO_write_low(&PORTB, LED_D2);


       //MAKING LED_D1 AS AN OUTPUT PIN
       GPIO_config_output(&DDRB, LED_D1);
       GPIO_write_low(&PORTB, LED_D1);
```

Name and Surname: *Kreshnik Shala*
Person ID: *226108*

Date: Tuesday, October 20, 2020

```
        //MAKING LED_D3 AS AN OUTPUT PIN
        GPIO_config_output(&DDRB, LED_D3);
        GPIO_write_low(&PORTB, LED_D3);

        //MAKING LED_D4 AS AN OUTPUT PIN
        GPIO_config_output(&DDRB, LED_D4);
        GPIO_write_low(&PORTB, LED_D4);

        //PDO as a BUTTON
        GPIO_config_input_pullup(&DDRD, BTN);



    /* Configuration of 16-bit Timer/Counter1
     * Set prescaler and enable overflow interrupt */
    TIM1_overflow_1s();
    TIM1_overflow_interrupt_enable();


     // Enables interrupts by setting the global interrupt mask
    sei();

    // Infinite loop
    while (1)
    {
        /* Empty loop. All subsequent operations are performed exclusively
         * inside interrupt service routines ISRs */
    }

    // Will never reach this
    return 0;
}

/* Interrupt service routines ----------------------------------------*/

 /* ISR starts when Timer/Counter1 overflows. Toggle D1 LED on
 * Multi-function shield. */

ISR(TIMER1_OVF_vect)
{

        // Checking button status to control the speed of LEDs
        if(GPIO_read(&PIND,BTN)==1)
        {



                TIM1_overflow_262ms(); // If button is pressed the state will change after
262ms
        }
        else if(GPIO_read(&PIND,BTN)==0)
```

Name and Surname: *Kreshnik Shala*
Person ID: *226108*

Date: Tuesday, October 20, 2020

```
	{
		TIM1_overflow_1s();   // If button is not pressed the state will change
after 1 second
	}



	 GPIO_toggle(&PORTB, arr[last_count]);  // TURN OFF THE PREVIOUS LED
	 GPIO_toggle(&PORTB, arr[count]);   // TURN ON THE LED (Because it was off)



	 // Deciding the movement of LEDs (forward or backward)
	 if (count == 3) // Checking maximum size
	 {
		back =1;  // If it has reached maximum size then LEDs will move backwards
	 }
	 else if (count == 0)  // Checking minimum size
	 {



		back =0; // If it has reached minimum size then LEDs will move in forward
direction
	 }



	 // Increasing or decreasing the index of array depending upon its direction
	 if (back == 0) // Direction is forward
	 {
		last_count =count; // Index to turn off the Last LED
		count++; // Index increasing because of forward direction
	 }
	 else if (back == 1) // Direction is backwards
	 {
		last_count=count; // Index to turn off the Last LED
		count--; // Index decreasing because of backward direction
	 }


}
```

Name and Surname: *Kreshnik Shala*
Person ID: *226108*

Date: Tuesday, October 20, 2020

1.2.4 gpio.h:

```
/*////////////////////////////////////////////////////////////////////////////////
///                                                                             ///
/// VUT FEKT                                  Name and Surname: Kreshnik Shala  ///
/// [BPA-DE2] Digital Electronics 2           Person ID: 226108                 ///
/// Date: Tuesday, October 20, 2020                                             ///
/// GitHub: https://github.com/ShalaKreshnik                                    ///
///                                                                             ///
*////////////////////////////////////////////////////////////////////////////////

#ifndef GPIO_H
#define GPIO_H

/**********************************************************************
 *
 * GPIO library for AVR-GCC.
 * ATmega328P (Arduino Uno), 16 MHz, AVR 8-bit Toolchain 3.6.2
 *
 **********************************************************************/

/**
 * @file  gpio.h
 * @brief GPIO library for AVR-GCC.
 *
 * @details
 * The library contains functions for controlling AVRs' gpio pin(s).
 *
 * @note
 * Based on AVR Libc Reference Manual. Tested on ATmega328P (Arduino Uno),
 * 16 MHz, AVR 8-bit Toolchain 3.6.2.
 */

/* Includes ----------------------------------------------------------*/
#include <avr/io.h>

/* Function prototypes -----------------------------------------------*/
/**
 * @brief Configure one output pin in Data Direction Register.
 * @param reg_name - Address of Data Direction Register, such as &DDRA,
 *                   &DDRB, ...
 * @param pin_num - Pin designation in the interval 0 to 7


 */
void GPIO_config_output(volatile uint8_t *reg_name, uint8_t pin_num);

void GPIO_config_input_nopull(volatile uint8_t *reg_name, uint8_t pin_num);

void GPIO_config_input_pullup(volatile uint8_t *reg_name, uint8_t pin_num);
```

```
void GPIO_write_low(volatile uint8_t *reg_name, uint8_t pin_num);


void GPIO_write_high(volatile uint8_t *reg_name, uint8_t pin_num); // Declaration of
function (To assign it a value of 1) with parameters *reg_name and pin_num (void does not
return any value)

void GPIO_toggle(volatile uint8_t *reg_name, uint8_t pin_num); // Declaration of function
(To toggle pin) with parameters *reg_name and pin_num (void does not return any value)
/* GPIO_toggle */

uint8_t GPIO_read(volatile uint8_t *reg_name, uint8_t pin_num);

#endif
```

1.2.5 gpio.c:

```
/*////////////////////////////////////////////////////////////////////////////////////
///                                                                                  ///
/// VUT FEKT                                  Name and Surname: Kreshnik Shala       ///
/// [BPA-DE2] Digital Electronics 2           Person ID: 226108                      ///
/// Date: Tuesday, October 20, 2020                                                  ///
/// GitHub: https://github.com/ShalaKreshnik                                         ///
///                                                                                  ///
*////////////////////////////////////////////////////////////////////////////////////



/*************************************************************************
 *
 * GPIO library for AVR-GCC.
 * ATmega328P (Arduino Uno), 16 MHz, AVR 8-bit Toolchain 3.6.2
 *
 ************************************************************************/

/* Includes ----------------------------------------------------------*/
#include "gpio.h"



/* Function definitions ----------------------------------------------*/
void GPIO_config_output(volatile uint8_t *reg_name, uint8_t pin_num)

{
    *reg_name = *reg_name | (1<<pin_num); // Set bit (or)
}
```

Name and Surname: *Kreshnik Shala*
Person ID: *226108*

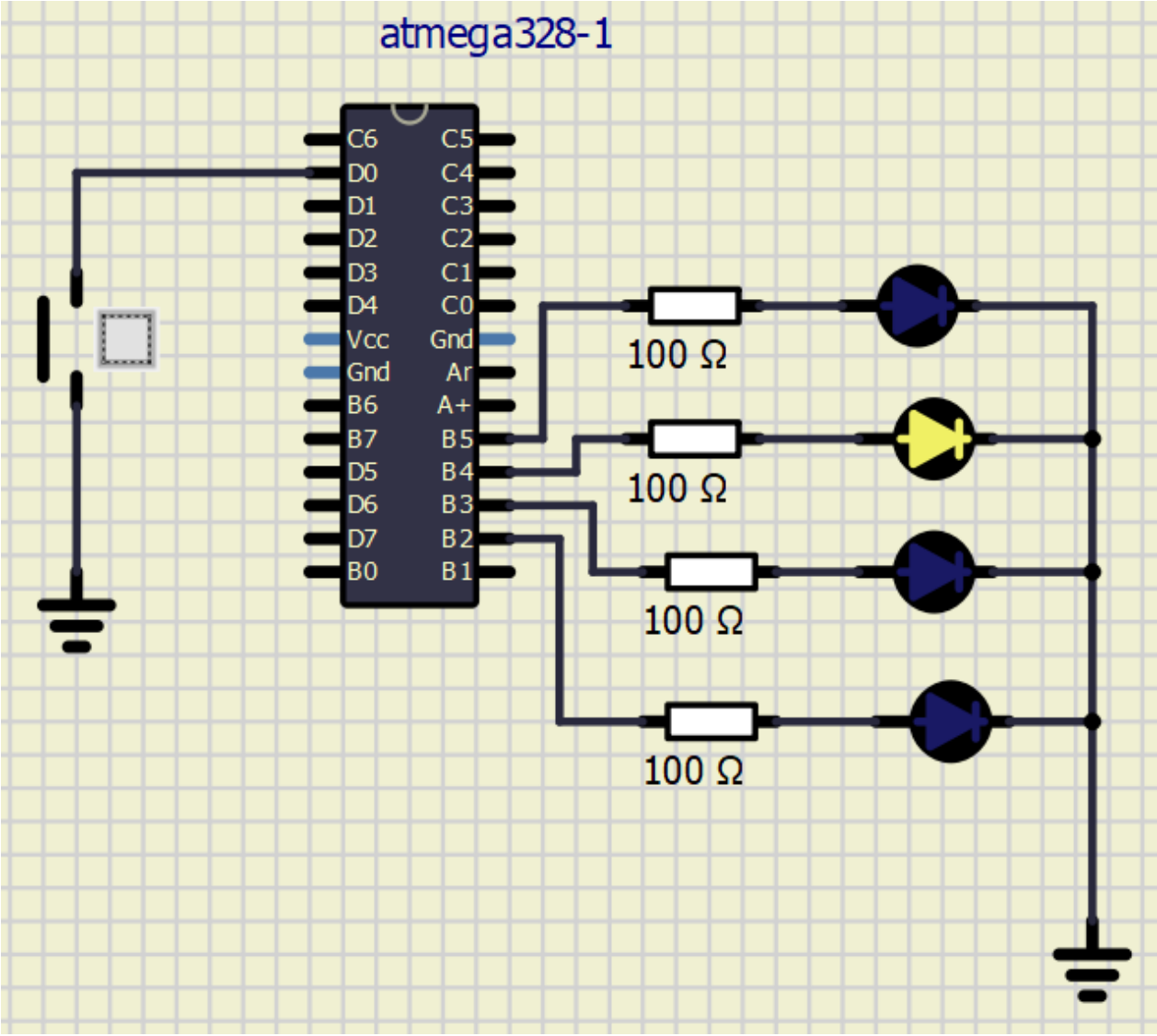
Date: Tuesday, October 20, 2020

```
/*-----------------------------------------------------------------------*/

/* GPIO_config_input_nopull */
void GPIO_config_input_nopull(volatile uint8_t *reg_name, uint8_t pin_num)
{
    *reg_name = *reg_name & ~(1<<pin_num);  // Data Direction Register


    *reg_name++;                    // Change pointer to Data Register
    *reg_name = *reg_name & ~(1<<pin_num);   // Data Register
}
/*-----------------------------------------------------------------------*/
void GPIO_config_input_pullup(volatile uint8_t *reg_name, uint8_t pin_num)
{
    *reg_name = *reg_name & ~(1<<pin_num);  // Data Direction Register
    *reg_name++;                    // Change pointer to Data Register
    *reg_name = *reg_name | (1<<pin_num);   // Data Register
}

/*-----------------------------------------------------------------------*/
void GPIO_write_low(volatile uint8_t *reg_name, uint8_t pin_num)
{
    *reg_name = *reg_name & ~(1<<pin_num); //Clear bit (and not)
}

/*-----------------------------------------------------------------------*/
/* GPIO_write_high */
void GPIO_write_high(volatile uint8_t *reg_name, uint8_t pin_num)
{
        *reg_name = *reg_name | (1<<pin_num); // Set bit (or)
}

/*-----------------------------------------------------------------------*/
/* GPIO_toggle */

void GPIO_toggle(volatile uint8_t *reg_name, uint8_t pin_num)
{
      *reg_name = *reg_name ^ (1<<pin_num); // Toggle bit (xor)
}

/*-----------------------------------------------------------------------*/
/* GPIO_read */
uint8_t GPIO_read(volatile uint8_t *reg_name, uint8_t pin_num)
{
      uint8_t result = 0; // Initializing result with 0
      if (*reg_name == pin_num) // If PIN is pressed
      {


            result = 1; // Value of result becomes 1 if PD0 is pressed.
      }
      return result; // Return the value stored in result (1 OR 0)
}
```

Name and Surname: *Kreshnik Shala*
Person ID: *226108*

Date: Tuesday, October 20, 2020

1.2.6 Screenshot of SimullDE circuits:

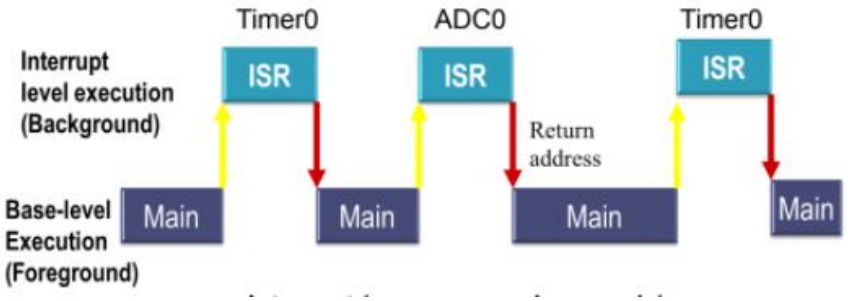


Name and Surname: *Kreshnik Shala*
Person ID: *226108*

Date: Tuesday, October 20, 2020

1.2.7 Difference between a common C function and interrupt service routine:

C function and ISR both work in a similar way, both consist of set of instructions. The difference is that we call a C function in a main function (int main) but ISR is called itself whenever the respective interrupt occurs. When the interrupt occurs, the program jumps to its ISR, performs the instruction written in ISR and then comes back to while loop.



Name and Surname: *Kreshnik Shala*
Person ID: *226108*

Date: Tuesday, October 20, 2020

1.3 PWM. Submit:

1.3.1 Table with PWM channels of ATmega328P:

Module:	Description:	MCU:	Arduino pin:
Timer/Counter0	OC0A	PD6	~6
	OC0B	PD5	~5
Timer/Counter1	OC1A	PB1	~9
	OC1B	PB2	~10
Timer/Counter2	OC2A	PB3	~11
	OC2B	PD3	~3

1.3.2 Describe the behavior of Clear Timer on Compare and Fast PWM modes:

First of all, a timer has a counter. And it does what a counter should do - namely counting. It counts repeatedly up or down to a limit or up to adjustable limits. It becomes a timer when it counts depending on time. And the timer only makes sense when reaching the limit or an intermediate value can trigger an event.

To put it simply, PWM is the technique of switching the digital outputs of our microcontroller periodically HIGH and LOW, i.e. generating square-wave signals with specific patterns. And since this is supposed to happen very quickly and, I n the background, a timer is used for this.