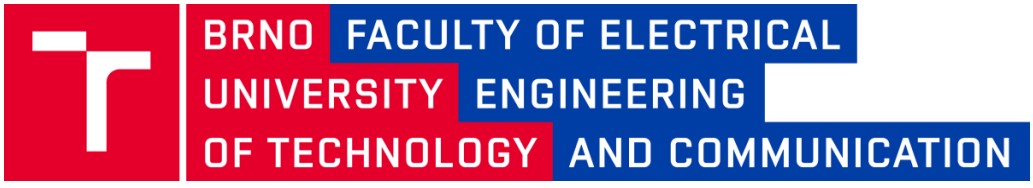
Session 1

-

Git version-control system, AVR tools

Author: Guillermo Cortés Orellana

Teacher: Tomáš Frýza



Lab assignment

1. **Link to my repository on GitHub**

   https://github.com/GuicoRM/Digital-Electronics-2

2. **What is the meaning of: | , & , ^ , ~ , << binary operators?**

   | → **OR**

A	B	OR
0	0	0
0	1	1
1	0	1
1	1	1

   & → **AND**

A	B	AND
0	0	0
0	1	0
1	0	0
1	1	1

^ → **XOR**

A	B	XOR
0	0	0
0	1	1
1	0	1
1	1	0

~ → **One's complement**

A	~
0	1
1	0

<< → **Left shift operator**

Number of places to shift

3. Morse code application: “DE2”

I designed the programm considering:

a) Little period of time between letters
b) Long morse code is three times greater tan short morse code

```c
/*
 * Proyecto1.c
 *
 * Created: 29/09/2020 16:24:34
 * Author : Guillermo Cortés Orellana (@GuicoRM on GitHub)
 */

/*****************************************************************************
 *
 * Blink "DE2" a LED and use function from the delay library.
 * ATmega328P (Arduino Uno), 16 MHz, AVR 8-bit Toolchain 3.6.2
 *
 * Copyright (c) 2020 Guillermo Cortés Orellana

 * Spaniard student at Brno University of Technology, Czechia
 * This work is licensed under the terms of the MIT license.
 *
 *****************************************************************************/

/* Defines -----------------------------------------------------------*/
#define LED_GREEN   PB5     // AVR pin where green LED is connected
#define SHORT_DELAY 500      // Delay in milliseconds
#define SHORT_DELAY2 1500      // Delay in milliseconds
#ifndef F_CPU
#define F_CPU 16000000      // CPU frequency in Hz required for delay
#endif

/* Includes ----------------------------------------------------------*/
#include <util/delay.h>     // Functions for busy-wait delay loops
#include <avr/io.h>         // AVR device-specific IO definitions
```

```c
/* Includes ------------------------------------------------------------*/
#include <util/delay.h>     // Functions for busy-wait delay loops
#include <avr/io.h>         // AVR device-specific IO definitions

/* Functions -----------------------------------------------------------*/
/**
 * Main function where the program execution begins. Toggle one LED
 * and use function from the delay library.
 */
int main(void)
{
    // Set pin as output in Data Direction Register
    // DDRB = DDRB or 0010 0000
    DDRB = DDRB | (1<<LED_GREEN);

    // Set pin LOW in Data Register (LED off)
    // PORTB = PORTB and 1101 1111
    PORTB = PORTB & ~(1<<LED_GREEN);

    // Infinite loop
    while (1)
    {
```

```c
    // Infinite loop
    while (1)
    {
        // Pause several milliseconds
        _delay_ms(SHORT_DELAY);

        // Invert LED in Data Register
        // PORTB = PORTB xor 0010 0000

        // D
        PORTB = PORTB ^ (1<<LED_GREEN);

        _delay_ms(SHORT_DELAY2); // 3 HIGH
        PORTB = PORTB ^ (1<<LED_GREEN); // CHANGE
        _delay_ms(SHORT_DELAY); // 1 LOW
        PORTB = PORTB ^ (1<<LED_GREEN); // CHANGE
        _delay_ms(SHORT_DELAY);// 1 HIGH
        PORTB = PORTB ^ (1<<LED_GREEN); // CHANGE
        _delay_ms(SHORT_DELAY); // 1 LOW
        PORTB = PORTB ^ (1<<LED_GREEN); // CHANGE
        _delay_ms(SHORT_DELAY); // 1 HIGH
        PORTB = PORTB ^ (1<<LED_GREEN); // CHANGE
        _delay_ms(SHORT_DELAY2); // 3 LOW (WAIT BETWEEN LETTERS)

        // E
        PORTB = PORTB ^ (1<<LED_GREEN); // CHANGE
        _delay_ms(SHORT_DELAY); // 1 UP
        PORTB = PORTB ^ (1<<LED_GREEN); // CHANGE
        _delay_ms(SHORT_DELAY); // 1 LOW
        _delay_ms(SHORT_DELAY2); // 3 LOW (WAIT BETWEEN LETTERS)
```

```
        // 2
        PORTB = PORTB ^ (1<<LED_GREEN); // CHANGE
        _delay_ms(SHORT_DELAY); // 1 HIGH
        PORTB = PORTB ^ (1<<LED_GREEN); // CHANGE
        _delay_ms(SHORT_DELAY); // 1 LOW

        PORTB = PORTB ^ (1<<LED_GREEN); // CHANGE
        _delay_ms(SHORT_DELAY); // 1 HIGH
        PORTB = PORTB ^ (1<<LED_GREEN); // CHANGE
        _delay_ms(SHORT_DELAY); // 1 LOW

        PORTB = PORTB ^ (1<<LED_GREEN); // CHANGE
        _delay_ms(SHORT_DELAY2); // 3 HIGH
        PORTB = PORTB ^ (1<<LED_GREEN); // CHANGE
        _delay_ms(SHORT_DELAY); //1 LOW

        PORTB = PORTB ^ (1<<LED_GREEN); // CHANGE
        _delay_ms(SHORT_DELAY2); // 3 HIGH
        PORTB = PORTB ^ (1<<LED_GREEN); // CHANGE
        _delay_ms(SHORT_DELAY); //1 LOW

        PORTB = PORTB ^ (1<<LED_GREEN); // CHANGE
        _delay_ms(SHORT_DELAY2); // 3 HIGH
        PORTB = PORTB ^ (1<<LED_GREEN); // CHANGE
        _delay_ms(SHORT_DELAY); // 1 LOW
        _delay_ms(SHORT_DELAY2); // 3 LOW (WAIT BETWEEN LETTERS)

    }

    // Will never reach this
    return 0;
}
```