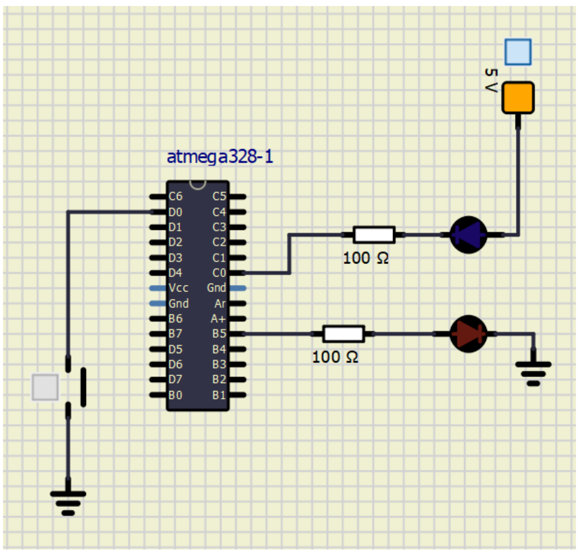
Kamil Káčer 211777   Repository: https://github.com/kamilio14/Digital-electronics-2.git

Work Done in a lab

```
/***********************************************************************
 *
 * Alternately toggle two LEDs when a push button is pressed.
 * ATmega328P (Arduino Uno), 16 MHz, AVR 8-bit Toolchain 3.6.2
 *
 * Copyright (c) 2018-2020 Tomas Fryza
 * Dept. of Radio Electronics, 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 LED_RED     PC0
#define BTN          PD0
#define BLINK_DELAY 250
#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

/* Functions ---------------------------------------------------------*/
/**
 * Main function where the program execution begins. Toggle two LEDs
 * when a push button is pressed.
 */
int main(void)
{
    /* GREEN LED */
    // Set pin as output in Data Direction Register...
    DDRB = DDRB | (1<<LED_GREEN);
    // ...and turn LED off in Data Register
    PORTB = PORTB & ~(1<<LED_GREEN);

    /* second LED */
       // Set pin as output in Data Direction Register...
       DDRC = DDRC | (1<<LED_RED);
       // ...and turn LED off in Data Register
       PORTC = PORTC & ~(1<<LED_RED);

       // Set pin as input
       DDRD = DDRD & ~(1<<BTN);
       // BTN Set to 1 because is BTN active low
       PORTD = PORTD | (1<<BTN);


    // Infinite loop
    while (1)
    {
        // Pause several milliseconds
        _delay_ms(BLINK_DELAY);

             loop_until_bit_is_clear(PIND, BTN);
       {
        PORTB = PORTB ^ (1<<LED_GREEN);
        PORTC = PORTC ^ (1<<LED_RED);
         }
    }

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

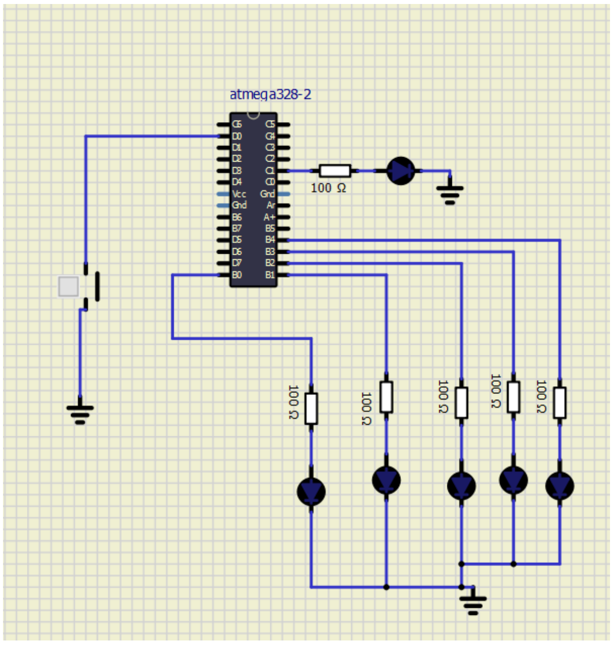


Home assignment

```
/***********************************************************************
 *
 * Alternately toggle two LEDs when a push button is pressed.
 * ATmega328P (Arduino Uno), 16 MHz, AVR 8-bit Toolchain 3.6.2
 *
 * Copyright (c) 2018-2020 Tomas Fryza
 * Dept. of Radio Electronics, Brno University of Technology, Czechia
 * This work is licensed under the terms of the MIT license.
 *
 **********************************************************************/

/* Defines -----------------------------------------------------------*/
#define LED_GREEN0  PB0     // AVR pin where green LED is connected
#define LED_GREEN1  PB1
#define LED_GREEN2  PB2
#define LED_GREEN3  PB3
#define LED_GREEN4  PB4
#define BUTTON_LED  PC1
#define BTN                PD0
#define BLINK_DELAY 500
```

```
#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

/* Functions ---------------------------------------------------------*/
/**
 * Main function where the program execution begins. Toggle two LEDs
 * when a push button is pressed.
 */
int main(void)
{
       int btn_memory = 0; // turnig off and on leds


    // LED 0
    DDRB = DDRB | (1<<LED_GREEN0);
    PORTB = PORTB & ~(1<<LED_GREEN0);

       // LED 1
       DDRB = DDRB | (1<<LED_GREEN1);
       PORTB = PORTB & ~(1<<LED_GREEN1);

       // LED 2
       DDRB = DDRB | (1<<LED_GREEN2);
       PORTB = PORTB & ~(1<<LED_GREEN2);

       // LED 3
       DDRB = DDRB | (1<<LED_GREEN3);
       PORTB = PORTB & ~(1<<LED_GREEN3);

       // LED 4
       DDRB = DDRB | (1<<LED_GREEN4);
       PORTB = PORTB & ~(1<<LED_GREEN4);

       //BUTTON

       DDRD = DDRD & ~(1<<BTN);
       PORTD = PORTD | (1<<BTN);

   //button led
   DDRC = DDRC | (1<<BUTTON_LED);
   PORTC = PORTC & ~(1<<BUTTON_LED);

    // Infinite loop
    while (1)
       {

              if (bit_is_clear(PIND, BTN))
              {
                     if (btn_memory == 0)
                     {
                            btn_memory = 1;  // turning on
                            PORTC = PORTC ^ (1<<BUTTON_LED);  // Turning on button led
to see if it works
                     }else{
                            btn_memory = 0;
                            PORTC = PORTC ^ (1<<BUTTON_LED);  // Turning off button led
                            _delay_ms(5000); // Time for turning off button
```

```c
				}
			}

			if (btn_memory == 1)
			{
				PORTB = PORTB ^ (1<<LED_GREEN0);  // Turning on Led0
				_delay_ms(BLINK_DELAY);

				PORTB = PORTB ^ (1<<LED_GREEN0);  // Turning off Led0
				PORTB = PORTB ^ (1<<LED_GREEN1);  // Turning on  Led1
				_delay_ms(BLINK_DELAY);

				PORTB = PORTB ^ (1<<LED_GREEN1);  // Turning off Led1
				PORTB = PORTB ^ (1<<LED_GREEN2);  // Turning on  Led2
				_delay_ms(BLINK_DELAY);

				PORTB = PORTB ^ (1<<LED_GREEN2);  // Turning off Led2
				PORTB = PORTB ^ (1<<LED_GREEN3);  // Turning on  Led3
				_delay_ms(BLINK_DELAY);

				PORTB = PORTB ^ (1<<LED_GREEN3);  // Turning off Led3
				PORTB = PORTB ^ (1<<LED_GREEN4);  // Turning on  Led4
				_delay_ms(BLINK_DELAY);

				PORTB = PORTB ^ (1<<LED_GREEN4);  // Turning off Led4
				PORTB = PORTB ^ (1<<LED_GREEN3);  // Turning on  Led3
				_delay_ms(BLINK_DELAY);

				PORTB = PORTB ^ (1<<LED_GREEN3);  // Turning off Led3
				PORTB = PORTB ^ (1<<LED_GREEN2);  // Turning on  Led2
				_delay_ms(BLINK_DELAY);

				PORTB = PORTB ^ (1<<LED_GREEN2);  // Turning off Led2
				PORTB = PORTB ^ (1<<LED_GREEN1);  // Turning on  Led1
				_delay_ms(BLINK_DELAY);

				PORTB = PORTB ^ (1<<LED_GREEN1);  // Turning off Led1
				PORTB = PORTB ^ (1<<LED_GREEN0);  // Turning on  Led0
				_delay_ms(BLINK_DELAY);

				PORTB = PORTB ^ (1<<LED_GREEN0);  // Turning on  Led0
				_delay_ms(BLINK_DELAY);

			}else
			{
			}

		}
		return 0;

		}
```



Charts

DDRB	Description
0	Input pin
1	Output

PORTB	Description
0	Output low value
1	Output High value

DDRB	PORTB	Direction	Internal pull up rezistor	Discription
0	0	input	no	Tri-state, high-impedance
0	1	input	yes	pulled low
1	0	output	no	output low
1	1	output	no	output high

PORT	PIN	I/O usage
A	x	Does not contain port A
B	0	Yes (Arduino pin 8)
	1	Yes (Arduino pin ~ 9)
	2	Yes (Arduino pin ~10)
	3	Yes (Arduino pin ~11)
	4	Yes (Arduino pin 12)
	5	Yes (Arduino pin 13)
	6	no
	7	no
C	0	Yes (Arduino pin A0)
	1	Yes (Arduino pin A1)
	2	Yes (Arduino pin A2)
	3	Yes (Arduino pin A3)
	4	Yes (Arduino pin A4)
	5	Yes (Arduino pin A5)
	6	no
	7	no
D	0	Yes (Arduino pin RX<-0)
	1	Yes (Arduino pin TX->)
	2	Yes (Arduino pin 2)
	3	Yes (Arduino pin ~3)
	4	Yes (Arduino pin 4)
	5	Yes (Arduino pin ~5)
	6	Yes (Arduino pin ~6)
	7	Yes (Arduino pin 7)