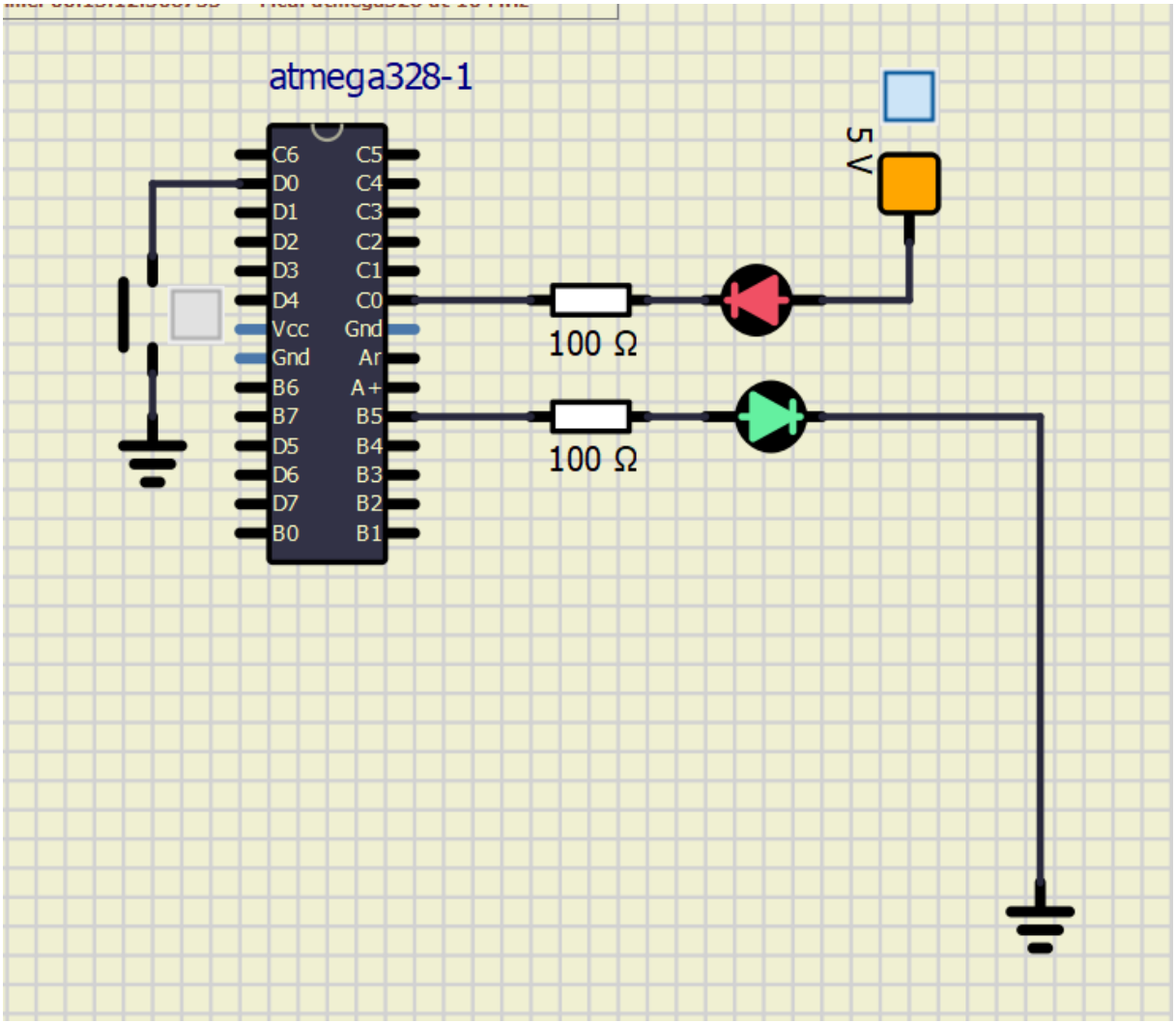
Data type	Number of bits	Range	Description
uint8_t	8	0, 1, …, 255	Unsigned 8-bit integer
int8_t	8	-128, …, 127	Signed 8-bit integer
uint16_t	16	0, …., 655535	Unsigned 16-bit integer
int16_t	16	-32768, …, 32767	Signed 16-bit integer
float	4	-3,4e38, …, 3,4e38	Single-precision floating-point
void	1	-	function return type

Deklarace – je jakýsi krátký předpis (jak funkce vypadá), měli by se psát do hlavičkových souborů.

Př.: void hello_world();

Definice – je kompletní kód celé funkce.

Př.: void hello_world(){printf("Hello world!");}



```c
/***********************************************************************
 *
 * GPIO library for AVR-GCC.
 * ATmega328P (Arduino Uno), 16 MHz, AVR 8-bit Toolchain 3.6.2
 *
 * Copyright (c) 2019-2020 Tomas Fryza
 * Dept. of Radio Electronics, Brno University of Technology, Czechia
 * This work is licensed under the terms of the MIT license.
 *
 **********************************************************************/

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

/* Function definitions ----------------------------------------------*/
void GPIO_config_output(volatile uint8_t *reg_name, uint8_t pin_num)
{
    //na danem bitu adresy nastavi 1
    *reg_name |= (1<<pin_num);
}

/*-----------------------------------------------------------------------*/
/* GPIO_config_input_nopull */
void GPIO_config_input_nopull(volatile uint8_t *reg_name, uint8_t pin_num)
{
    //(inp nopull)
    //na danem bitu adresy nastavi 0
    //na danem bitu o jednu vyssi adresy nastavi 0
    *reg_name &= ~(1<<pin_num);  // Data Direction Register
    *reg_name++;                    // Change pointer to Data Register
    *reg_name &= ~(1<<pin_num);   // Data Register
}

/*-----------------------------------------------------------------------*/
void GPIO_config_input_pullup(volatile uint8_t *reg_name, uint8_t pin_num)
{
    //(inp pull)
    //na danem bitu adresy nastavi 0
    //na danem bitu o jednu vyssi adresy nastavi 1
    *reg_name &= ~(1<<pin_num);  // Data Direction Register
    *reg_name++;                    // Change pointer to Data Register
    *reg_name |= (1<<pin_num);   // Data Register
}

/*-----------------------------------------------------------------------*/
void GPIO_write_low(volatile uint8_t *reg_name, uint8_t pin_num)
{
    //na danem bitu adresy nastavi 0
    *reg_name &= ~(1<<pin_num);
}

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

```
54  void GPIO_write_high(volatile uint8_t *reg_name, uint8_t pin_num)
55  {
56      //na danem bitu adresy nastavi 1
57      *reg_name |= (1<<pin_num);
58  }
59
60  /*-------------------------------------------------------------------*/
61  /* GPIO_toggle */
62  void GPIO_toggle(volatile uint8_t *reg_name, uint8_t pin_num)
63  {
64      //na danem bitu adresy nastavi negaci bitu
65      *reg_name ^= (1<<pin_num);
66  }
67
68  /*-------------------------------------------------------------------*/
69  /* GPIO_read */
70  uint8_t GPIO_read(volatile uint8_t *reg_name, uint8_t pin_num)
71  {
72      // kdyz je dany bit na dane adrese 0
73      // tak se vrati 0 jinak 1
74      if(bit_is_clear(*reg_name, pin_num))
75          return 0;
76      else
77          return 1;
78  }
```

```c
/***********************************************************************
 *
 * Alternately toggle two LEDs when a push button is pressed. Use
 * functions from GPIO library.
 * ATmega328P (Arduino Uno), 16 MHz, AVR 8-bit Toolchain 3.6.2
 *
 * Copyright (c) 2019-2020 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     // AVR pin where red LED is connected
#define BTN         PD0     // AVR pin where button is connected
#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
#include "gpio.h"           // GPIO library for AVR-GCC

/* Function definitions ----------------------------------------------*/
/**
 * Main function where the program execution begins. Toggle two LEDs
 * when a push button is pressed. Functions from user-defined GPIO
 * library is used instead of low-level logic operations.
 */
int main(void)
{
    /* GREEN LED */
    GPIO_config_output(&DDRB, LED_GREEN);
    GPIO_write_low(&PORTB, LED_GREEN);

    /* second LED */
    // WRITE YOUR CODE HERE
    /* RED LED */
    GPIO_config_output(&DDRC, LED_RED);
    GPIO_write_high(&PORTC, LED_RED);

    /* push button */
    // WRITE YOUR CODE HERE
    GPIO_config_input_pullup(&DDRD, BTN);

    // Infinite loop
    while (1)
    {
        // Pause several milliseconds
```

```
54          _delay_ms(BLINK_DELAY);
55
56          // WRITE YOUR CODE HERE
57          // sepnuti tlacitka rozblika LEDky
58          if(!GPIO_read(&PIND, BTN))
59          {
60              GPIO_toggle(&PORTB, LED_GREEN);
61              GPIO_toggle(&PORTC, LED_RED);
62          }
63      }
64
65      // Will never reach this
66      return 0;
67  }
```