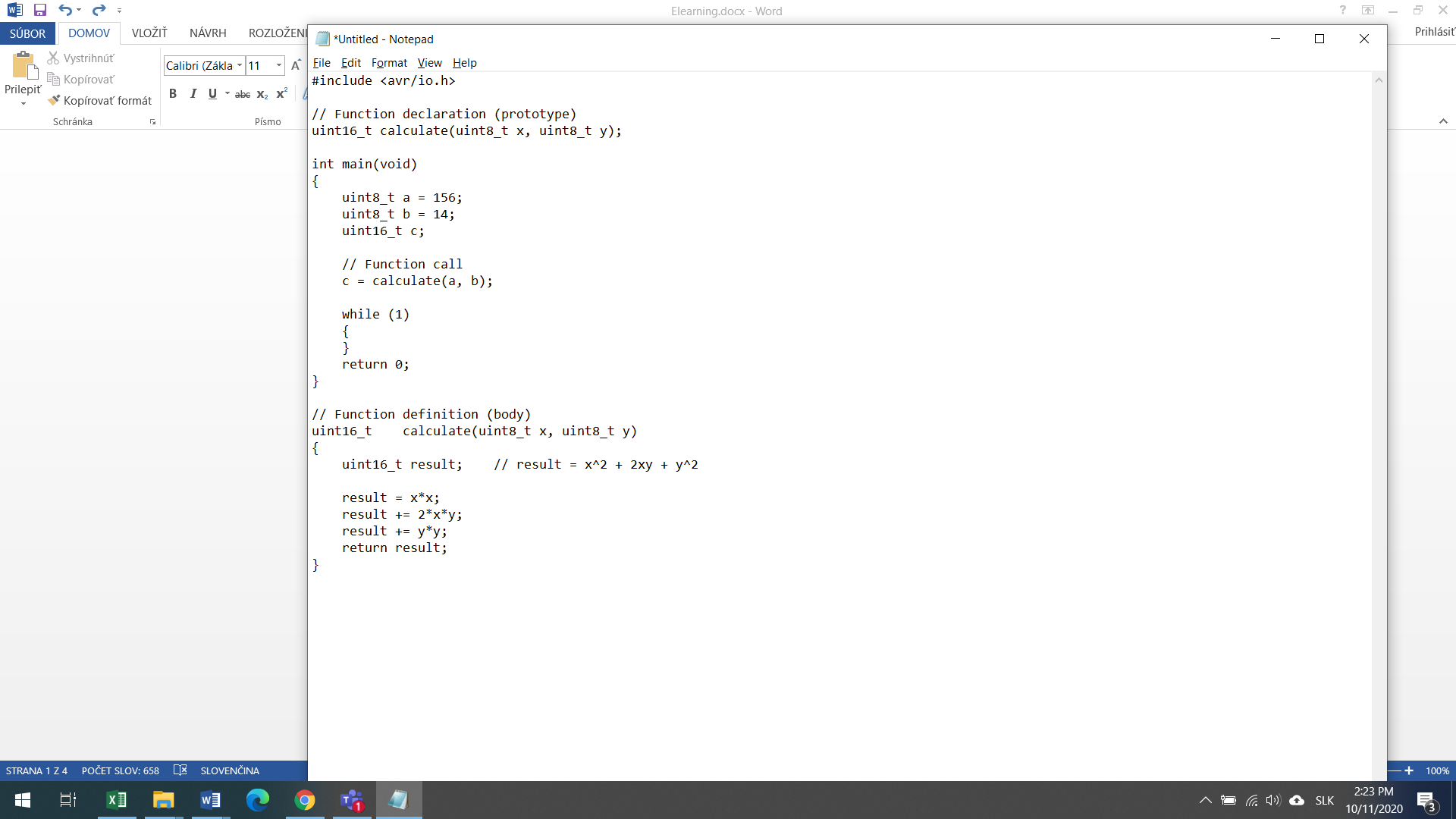
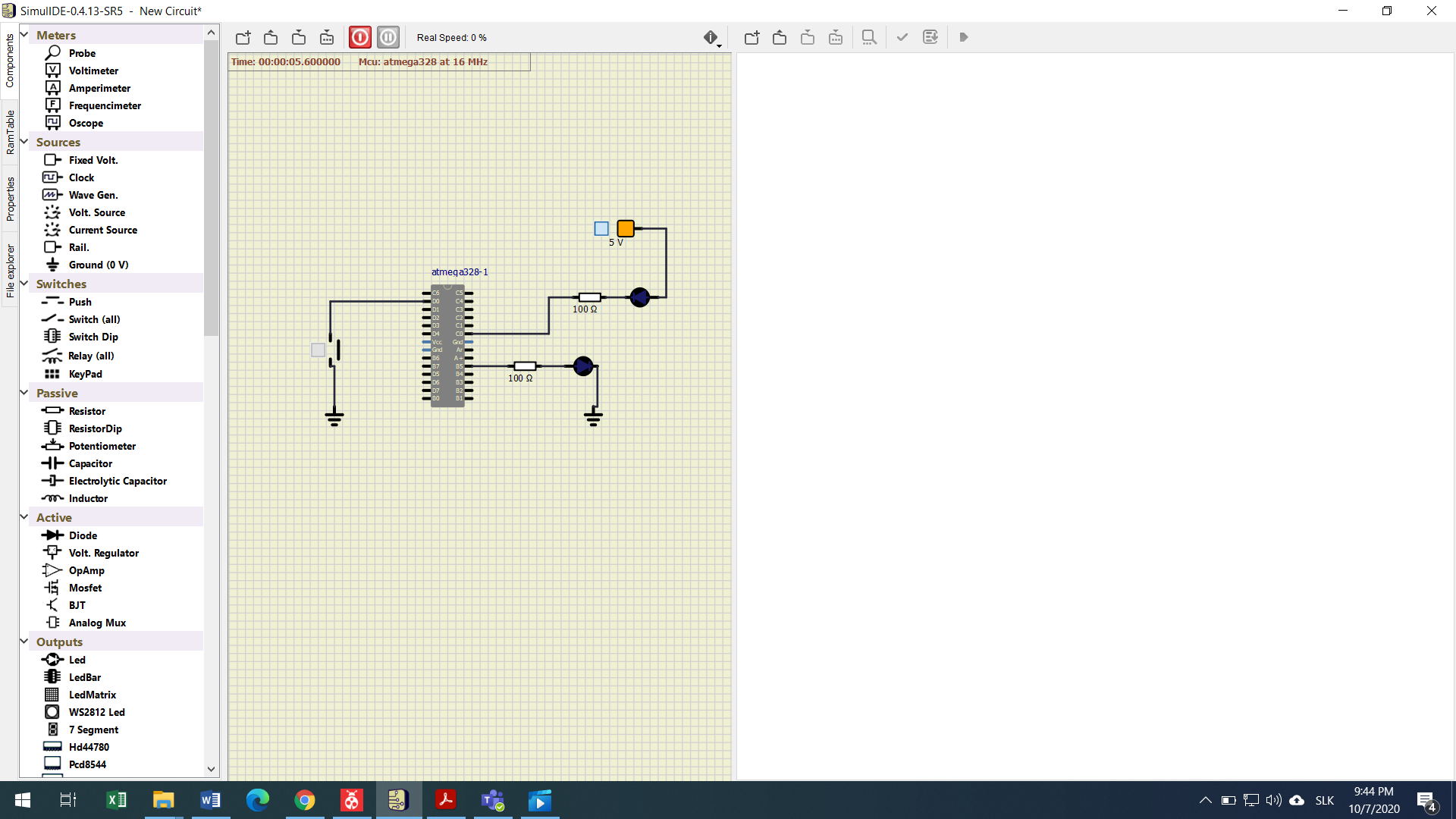
**Function:**



**Chart**



**Scheme**



/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* 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)

{

\*reg\_name = \*reg\_name | (1<<pin\_num);

}

/\*--------------------------------------------------------------------\*/

void GPIO\_config\_input\_nopullup(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);

}

/\*--------------------------------------------------------------------\*/

void GPIO\_write\_high(volatile *uint8\_t* \*reg\_name, *uint8\_t* pin\_num)

{

\*reg\_name = \*reg\_name | (1<<pin\_num);

}

/\*--------------------------------------------------------------------\*/

/\* GPIO\_toggle \*/

void GPIO\_toggle(volatile *uint8\_t* \*reg\_name, *uint8\_t* pin\_num)

{

\*reg\_name = \*reg\_name ^(1<<pin\_num);

}

/\*--------------------------------------------------------------------\*/

/\* GPIO\_read \*/

*uint8\_t* GPIO\_read(volatile *uint8\_t* \*reg\_name, *uint8\_t* pin\_num)

{

if (bit\_is\_clear(\*reg\_name, pin\_num))

{

return 0;

}

else

{

return 1;

}

}

Main.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

#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

#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 \*/

GPIO\_config\_output(&DDRC, LED\_RED);

GPIO\_write\_high(&PORTC, LED\_RED);

/\* push button \*/

GPIO\_config\_input\_pullup(&DDRD, BTN);

//GPIO\_write\_high(&PORTD, BTN);

// WRITE YOUR CODE HERE

// Infinite loop

while (1)

{

// Pause several milliseconds

*\_delay\_ms*(BLINK\_DELAY);

if(GPIO\_read(&PIND, BTN) == 0)

{

GPIO\_toggle(&PORTB, LED\_GREEN);

GPIO\_toggle(&PORTC, LED\_RED);

}

}

// Will never reach this

return 0;

}

Gpio.h

#ifndef GPIO\_H

#define GPIO\_H

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* 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.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*

\* @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.

\*

\* @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 <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);

void GPIO\_toggle(volatile *uint8\_t* \*reg\_name, *uint8\_t* pin\_num);

*uint8\_t* GPIO\_read(volatile *uint8\_t* \*reg\_name, *uint8\_t* pin\_num);

#endif

Deklarácia:

* Hovorí compileru ako sa funkcia volá aký ma typ a aké ma parametre.

Int8\_t calculate(int x, int y );

Definicia

* Poskytuje informácie a tele funkcie ako sa funkcia správa.

Int8\_t calculate(int x, int y);{

Return(x +y);

}