https://github.com/DogancanG/Digital-electronics-2/tree/main/Labs/03-gpio

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,…, 65535	Unsigned 16-bit integer
int16_t	16	-32768, …, 32767	Signed 16-bit integer
float	32	-3.4e+38, ..., 3.4e+38	Single-precision floating-point
void	X	X	X

1-)

A function definition means the specification of the function name, the return type, the parameters and the complete function body- the actual function. So it is the complete description of the function.

A function declaration gives information to the compiler about a function name and how to call the function. A compiler reads and translates the source code from top to bottom. If he comes across a word – for example a function name- that he is not yet familiar with at one point in the source text, an error message will be given. It is therefore necessary to make functions known before they are used.

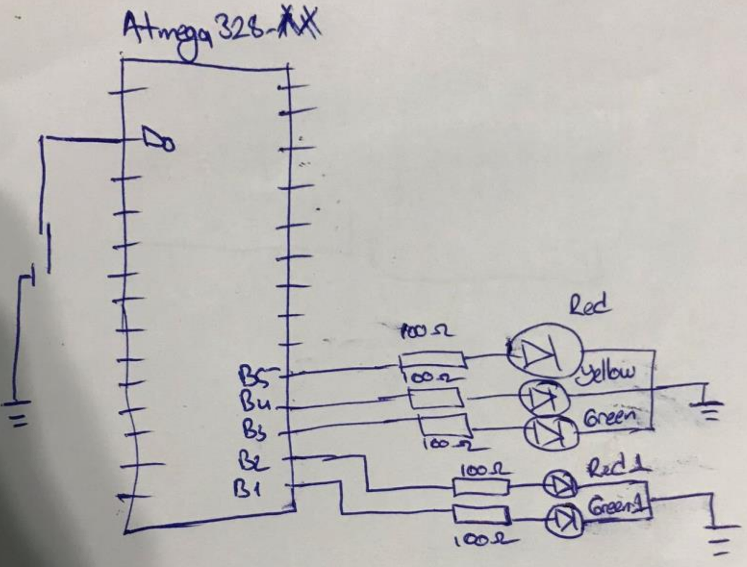
2-)

```
-------------------gpio.c---------------------------------------------

/* Includes ----------------------------------------------------------*/
#include "gpio.h" //Needed for prototypes
#include "avr/sfr_defs.h"
/* Function definitions ----------------------------------------------*/
/**********************************************************************
 * Function: GPIO_config_output()
 * Purpose:  Configure one output pin in Data Direction Register.
 * Input:    reg_name - Address of Data Direction Register, such as &DDRB
 *           pin_num - Pin designation in the interval 0 to 7
 * Returns:  none
 **********************************************************************/
void GPIO_config_output(volatile uint8_t *reg_name, uint8_t pin_num)
{
    *reg_name = *reg_name | (1<<pin_num);
}

/**********************************************************************
 * Function: GPIO_config_input_nopull()
 **********************************************************************/

/**********************************************************************
 * Function: GPIO_config_input_pullup()
 * Purpose:  Configure one input pin and enable pull-up.
 * Input:    reg_name - Address of Data Direction Register, such as &DDRB
 *           pin_num - Pin designation in the interval 0 to 7
```

```c
 * Returns:  none
 **********************************************************************/
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
}

/**********************************************************************
 * Function: GPIO_write_low()
 * Purpose:  Write one pin to a low value.
 * Input:    reg_name - Address of Port Register, such as &PORTB
 *           pin_num - Pin designation in the interval 0 to 7
 * Returns:  none
 **********************************************************************/
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);
}

/**********************************************************************
 * Function: GPIO_write_high()
 **********************************************************************/
void GPIO_write_high(volatile uint8_t *reg_name, uint8_t pin_num)
{
	*reg_name = *reg_name | (1<<pin_num);
}
/**********************************************************************
 * Function: GPIO_toggle()
 **********************************************************************/
void GPIO_toggle(volatile uint8_t *reg_name, uint8_t pin_num)
{
	*reg_name = *reg_name ^ (1<<pin_num);
}
/**********************************************************************
 * Function: GPIO_read()
 **********************************************************************/
uint8_t GPIO_read(volatile uint8_t *reg_name, uint8_t pin_num)

{
	if(bit_is_clear(*reg_name,pin_num))
		return 1;
	else
		return 0;
	value 0;
}

-----------------gpio.h--------------------------------------------

#ifndef GPIO_H_
#define GPIO_H_


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

```
/* Function prototypes ----------------------------------------------*/
/**
 * @name Functions
 */

/**
 * @brief  Configure one output pin in Data Direction Register.
 * @param  reg_name Address of Data Direction Register, such as &DDRB
 * @param  pin_num  Pin designation in the interval 0 to 7
 * @return none
 */
void GPIO_config_output(volatile uint8_t *reg_name, uint8_t pin_num);


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


/**
 * @brief  Configure one input pin and enable pull-up.
 * @param  reg_name Address of Data Direction Register, such as &DDRB
 * @param  pin_num  Pin designation in the interval 0 to 7
 * @return none
 */
void GPIO_config_input_pullup(volatile uint8_t *reg_name, uint8_t pin_num);


/**
 * @brief  Write one pin to a low value.
 * @param  reg_name Address of Port Register, such as &PORTB
 * @param  pin_num  Pin designation in the interval 0 to 7
 * @return none
 */
void GPIO_write_low(volatile uint8_t *reg_name, uint8_t pin_num);


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

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

/**
 * @brief  Read a value from input pin.
 * @param  reg_name Address of Pin Register, such as &PINB
 * @param  pin_num  Pin designation in the interval 0 to 7
 * @return Pin value
 */
uint8_t GPIO_read(volatile uint8_t *reg_name, uint8_t pin_num);

/** @} */



#endif /* GPIO_H_ */



-----------main-c----------------------------------------------------
/* Defines -----------------------------------------------------------*/
#define LED_GREEN   PB5     // AVR pin where green LED is connected
#define LED_RED PC0
```

```c
#define BIN 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
uint8_t perform=0;




int main(void)
{
    // Green LED at port B
    GPIO_config_output(&DDRB, LED_GREEN);
    GPIO_write_low(&PORTB, LED_GREEN);

    // Configure the second LED at port C
       GPIO_config_output(&DDRC, LED_RED);
       GPIO_write_low(&PORTC, LED_RED);

    // Configure Push button at port D and enable internal pull-up resistor
       GPIO_config_input_pullup(&DDRD,BIN);

    // Infinite loop
    while (1)
    {
        // Pause several milliseconds
        _delay_ms(BLINK_DELAY);
              perform=GPIO_read(&PORTD,BIN);

              if (perform==1)
              {
                     GPIO_toggle(&PORTB,LED_GREEN);
                     GPIO_toggle(&PORTC,LED_RED);
              }
    }

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


Atmega 328-~~XX~~
D0
B5
B4
B3
B2
B1
100 Ω
100 Ω
100 Ω
100 Ω
100 Ω
Red
Yellow
Green
Red 1
Green 1