Session 2

-

Control of GPIO, LED, push button

Author: Guillermo Cortés Orellana

Teacher: Tomáš Frýza

BRNO FACULTY OF ELECTRICAL
UNIVERSITY ENGINEERING
OF TECHNOLOGY AND COMMUNICATION

Lab assignment

1. **LED example**

- Tables for DDRB, PORTB, and their combination

DDRB	Description
0	Input Pin
1	Output Pin

PORTB	Description
0	Output low value
1	Output high value

DDRB	PORTB	Direction	Internal pull-up resistor	Description
0	0	Input	NO	Tri-state (Hi-Z)
0	1	Input	YES	Pxn will source current if ext. Pulled low
1	0	Output	NO	Output Low
1	1	Output	NO	Outout High

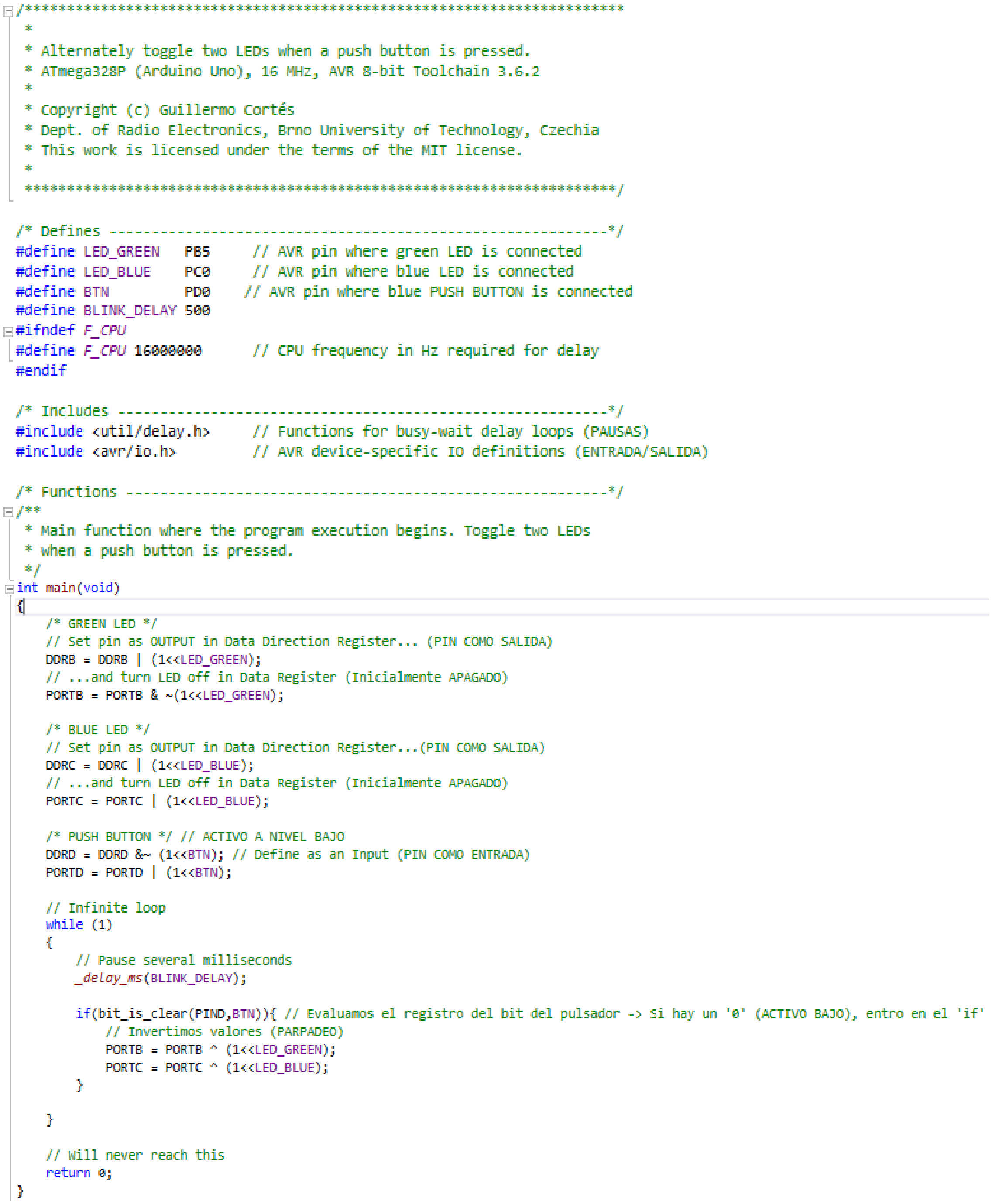
- Table with input/output pins available on ATmega328P

PORT	Pin	Input/Output usage
A	x	Doesn’t contain PORT A
B	0	Yes (Pin 8)
B	1	Yes (Pin -9)
B	2	Yes (Pin -10)
B	3	Yes (Pin -11)
B	4	Yes (Pin 12)
B	5	Yes (Pin 13)
B	6	NO
B	7	NO

PORT	Pin	Input/Output usage
C	0	Yes (Pin A0)
C	1	Yes (Pin A1)
C	2	Yes (Pin A2)
C	3	Yes (Pin A3)
C	4	Yes (Pin A4)
C	5	Yes (Pin A5)
C	6	NO
C	7	NO

PORT	Pin	Input/Output usage
D	0	Yes (Pin RX <- 0)
D	1	Yes (Pin TX -> 1)
D	2	Yes (Pin 2)
D	3	Yes (Pin -3)
D	4	Yes (Pin 4)
D	5	Yes (Pin -5)
D	6	Yes (Pin -6)
D	7	Yes (Pin 7)

- C code with two LEDs and push Button

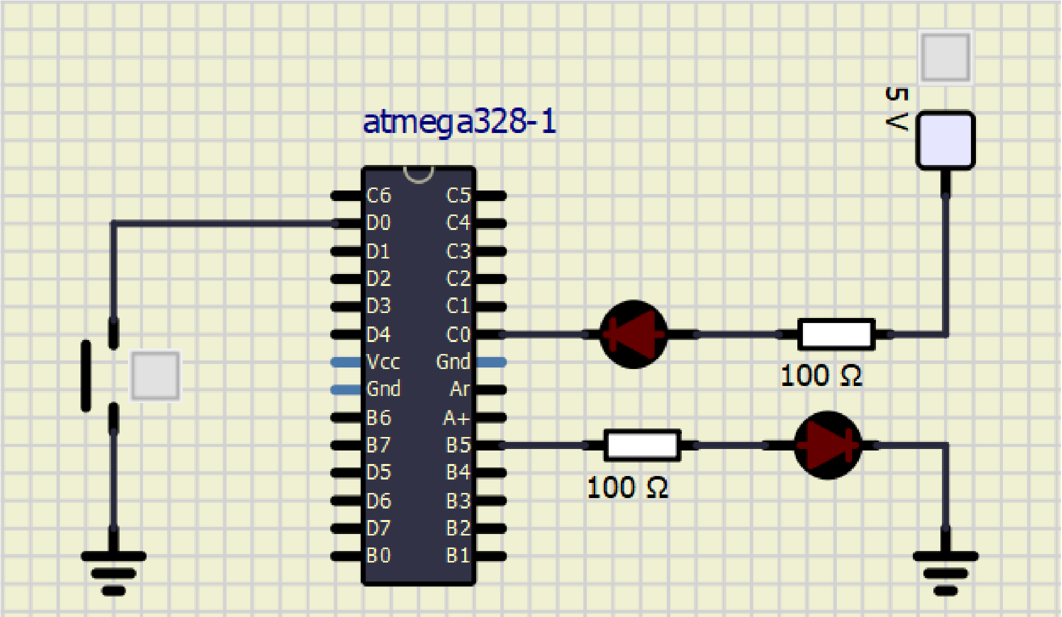


```c
/***********************************************************************
 *
 * Alternately toggle two LEDs when a push button is pressed.
 * ATmega328P (Arduino Uno), 16 MHz, AVR 8-bit Toolchain 3.6.2
 *
 * Copyright (c) Guillermo Cortés
 * Dept. of Radio Electronics, Brno University of Technology, 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_BLUE    PC0     // AVR pin where blue LED is connected
#define BTN         PD0    // AVR pin where blue PUSH 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 (PAUSAS)
#include <avr/io.h>         // AVR device-specific IO definitions (ENTRADA/SALIDA)

/* 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... (PIN COMO SALIDA)
    DDRB = DDRB | (1<<LED_GREEN);
    // ...and turn LED off in Data Register (Inicialmente APAGADO)
    PORTB = PORTB & ~(1<<LED_GREEN);

    /* BLUE LED */
    // Set pin as OUTPUT in Data Direction Register...(PIN COMO SALIDA)
    DDRC = DDRC | (1<<LED_BLUE);
    // ...and turn LED off in Data Register (Inicialmente APAGADO)
    PORTC = PORTC | (1<<LED_BLUE);

    /* PUSH BUTTON */ // ACTIVO A NIVEL BAJO
    DDRD = DDRD &~ (1<<BTN); // Define as an Input (PIN COMO ENTRADA)
    PORTD = PORTD | (1<<BTN);

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

        if(bit_is_clear(PIND,BTN)){ // Evaluamos el registro del bit del pulsador -> Si hay un '0' (ACTIVO BAJO), entro en el 'if'
            // Invertimos valores (PARPADEO)
            PORTB = PORTB ^ (1<<LED_GREEN);
            PORTC = PORTC ^ (1<<LED_BLUE);
        }

    }

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

You can find the code on my GitHub:

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

- Screenshot of SimulIDE circuit



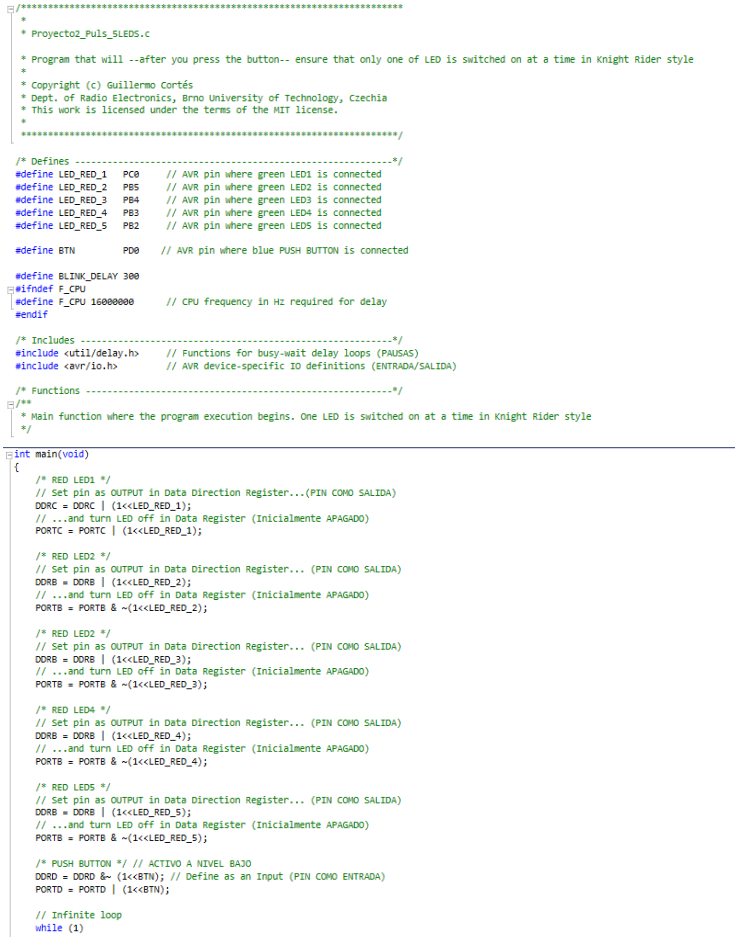
2. Knight Rider application

- C code

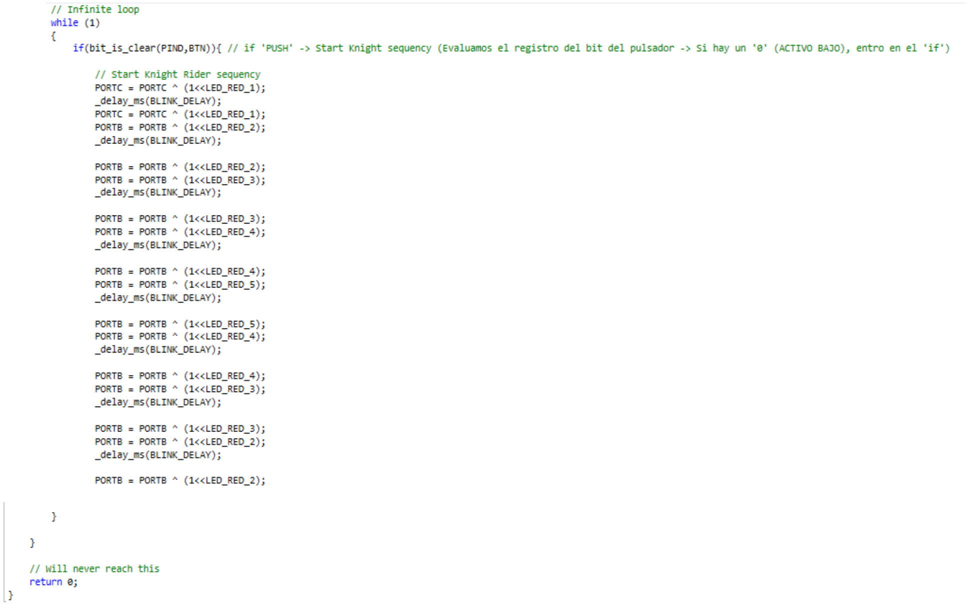
Note 1: all LEDs are **RED** in order to simúlate ‘Knight Rider style’

Note 2: LED 1 is designed in active-low way

Note 3: LED 2, LED 3, LED 4 and LED 5 are designed in active-high way



```c
/***********************************************************************
 *
 * Proyecto2_Puls_5LEDS.c

 * Program that will --after you press the button-- ensure that only one of LED is switched on at a time in Knight Rider style
 *
 * Copyright (c) Guillermo Cortés
 * Dept. of Radio Electronics, Brno University of Technology, Czechia
 * This work is licensed under the terms of the MIT license.
 *
 **********************************************************************/

/* Defines -----------------------------------------------------------*/
#define LED_RED_1   PC0     // AVR pin where green LED1 is connected
#define LED_RED_2   PB5     // AVR pin where green LED2 is connected
#define LED_RED_3   PB4     // AVR pin where green LED3 is connected
#define LED_RED_4   PB3     // AVR pin where green LED4 is connected
#define LED_RED_5   PB2     // AVR pin where green LED5 is connected

#define BTN         PD0    // AVR pin where blue PUSH BUTTON is connected

#define BLINK_DELAY 300
#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 (PAUSAS)
#include <avr/io.h>         // AVR device-specific IO definitions (ENTRADA/SALIDA)

/* Functions ---------------------------------------------------------*/
/**
 * Main function where the program execution begins. One LED is switched on at a time in Knight Rider style
 */
int main(void)
{
    /* RED LED1 */
    // Set pin as OUTPUT in Data Direction Register...(PIN COMO SALIDA)
    DDRC = DDRC | (1<<LED_RED_1);
    // ...and turn LED off in Data Register (Inicialmente APAGADO)
    PORTC = PORTC | (1<<LED_RED_1);

    /* RED LED2 */
    // Set pin as OUTPUT in Data Direction Register... (PIN COMO SALIDA)
    DDRB = DDRB | (1<<LED_RED_2);
    // ...and turn LED off in Data Register (Inicialmente APAGADO)
    PORTB = PORTB & ~(1<<LED_RED_2);

    /* RED LED2 */
    // Set pin as OUTPUT in Data Direction Register... (PIN COMO SALIDA)
    DDRB = DDRB | (1<<LED_RED_3);
    // ...and turn LED off in Data Register (Inicialmente APAGADO)
    PORTB = PORTB & ~(1<<LED_RED_3);

    /* RED LED4 */
    // Set pin as OUTPUT in Data Direction Register... (PIN COMO SALIDA)
    DDRB = DDRB | (1<<LED_RED_4);
    // ...and turn LED off in Data Register (Inicialmente APAGADO)
    PORTB = PORTB & ~(1<<LED_RED_4);

    /* RED LED5 */
    // Set pin as OUTPUT in Data Direction Register... (PIN COMO SALIDA)
    DDRB = DDRB | (1<<LED_RED_5);
    // ...and turn LED off in Data Register (Inicialmente APAGADO)
    PORTB = PORTB & ~(1<<LED_RED_5);

    /* PUSH BUTTON */ // ACTIVO A NIVEL BAJO
    DDRD = DDRD &~ (1<<BTN); // Define as an Input (PIN COMO ENTRADA)
    PORTD = PORTD | (1<<BTN);

    // Infinite loop
    while (1)
```

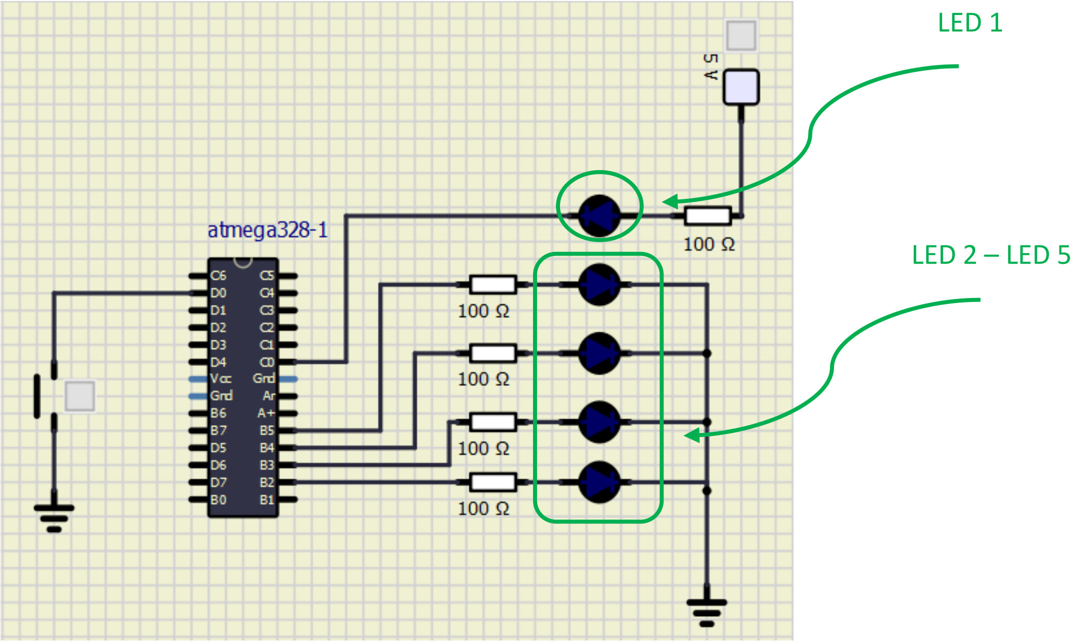


```
        // Infinite loop
        while (1)
        {
            if(bit_is_clear(PIND,BTN)){ // if 'PUSH' -> Start Knight sequency (Evaluamos el registro del bit del pulsador -> Si hay un '0' (ACTIVO BAJO), entro en el 'if')

                // Start Knight Rider sequency
                PORTC = PORTC ^ (1<<LED_RED_1);
                _delay_ms(BLINK_DELAY);
                PORTC = PORTC ^ (1<<LED_RED_1);
                PORTB = PORTB ^ (1<<LED_RED_2);
                _delay_ms(BLINK_DELAY);

                PORTB = PORTB ^ (1<<LED_RED_2);
                PORTB = PORTB ^ (1<<LED_RED_3);
                _delay_ms(BLINK_DELAY);

                PORTB = PORTB ^ (1<<LED_RED_3);
                PORTB = PORTB ^ (1<<LED_RED_4);
                _delay_ms(BLINK_DELAY);

                PORTB = PORTB ^ (1<<LED_RED_4);
                PORTB = PORTB ^ (1<<LED_RED_5);
                _delay_ms(BLINK_DELAY);

                PORTB = PORTB ^ (1<<LED_RED_5);
                PORTB = PORTB ^ (1<<LED_RED_4);
                _delay_ms(BLINK_DELAY);

                PORTB = PORTB ^ (1<<LED_RED_4);
                PORTB = PORTB ^ (1<<LED_RED_3);
                _delay_ms(BLINK_DELAY);

                PORTB = PORTB ^ (1<<LED_RED_3);
                PORTB = PORTB ^ (1<<LED_RED_2);
                _delay_ms(BLINK_DELAY);

                PORTB = PORTB ^ (1<<LED_RED_2);

        }

    }

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

You can find the code on my GitHub:

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

- Screenshot of SimulIDE circuit


LED 1
5 V
atmega328-1
100 Ω
LED 2 – LED 5
C6
D0
D1
D2
D3
D4
Vcc
Gnd
B6
B7
D5
D6
D7
B0
C5
C4
C3
C2
C1
C0
Gnd
Ar
A+
B5
B4
B3
B2
B1
100 Ω
100 Ω
100 Ω
100 Ω