DIGITAL ELECTRONICS 2 LAB ASSIGNMENT 2

Name: Demirkan Korbey Baglamac

1)

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, high-impedance
0	1	input	yes	Pxn will source current
1	0	output	no	Output low
1	1	output	no	Output high

Port	Pin	Input/output usage?
A	x	Microcontroller ATmega328P 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	
	7	
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	
	7	

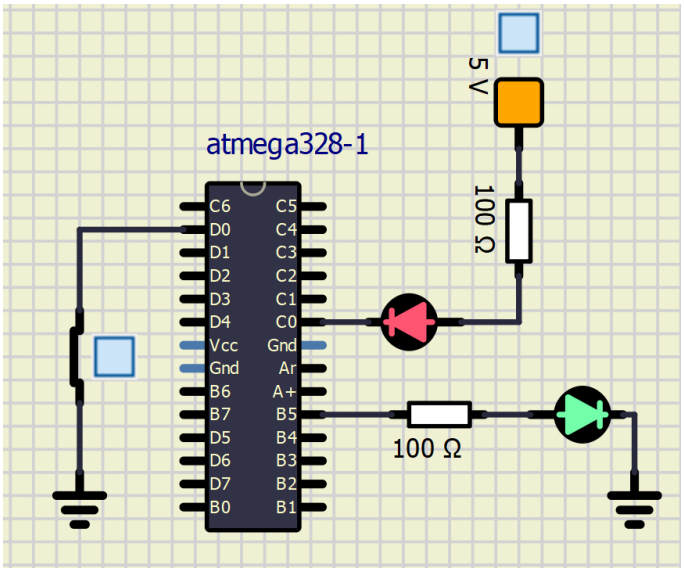
D	0	Yes (Arduino pin RX<-0)
	1	Yes (Arduino pin TX ->1)
	2	Yes (Arduino pin 2)
	3	Yes (Arduino pin ~3)
	4	Yes (Arduino pin 4)
	5	Yes (Arduino pin ~5)
	6	Yes (Arduino pin ~6)
	7	Yes (Arduino pin 7)

C Code with two leds and a push button:

```
/* Defines -----------------------------------------------------------
--*/
#define LED_GREEN   PB5     // AVR pin where green LED is connected
#define LED_RED     PC0
#define BLINK_DELAY 500
#define BTN         PD0
#ifndef F_CPU
#define F_CPU 16000000      // CPU frequency in Hz required for dela
y
#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);  //Turn OFF

    /* second LED */
    DDRC = DDRC | (1<<LED_RED);
    PORTC = PORTC & ~(1<<LED_RED); // Turn ON

    /* PUSH BUTTON*/
    DDRD = DDRD & ~(1<<BTN); //input
    PORTD = PORTD | (1<<BTN); //enable internal pull up

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

        // WRITE YOUR CODE HERE
```

```
        if(bit_is_clear(PIND, BTN))
        {
            PORTB = PORTB ^ (1<<LED_GREEN); //Invert
            PORTC = PORTC ^ (1<<LED_RED); //Invert
        }

    }

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

Screenshot of SimulIDE Ciruit:



2) Knight Rider Application Code:

```
/*
 * knight_rider.c
 *
 * Created: 5.10.2020 09:29:44
 * Author : dkorb
 */

/* Defines -----------------------------------------------------------
--*/
#define LED1 PB1
#define LED2 PB2
#define LED3 PB3
#define LED4 PB4
#define LED5 PB5
#define BLINK_DELAY 500
#define BTN         PD0
```

```c
#ifndef F_CPU
#define F_CPU 16000000      // CPU frequency in Hz required for dela
y
#endif

/* Includes ------------------------------------------------------
--*/
#include <util/delay.h>     // Functions for busy-wait delay loops
#include <avr/io.h>         // AVR device-specific IO definitions

/* Functions -----------------------------------------------------
--*/

void clear();

/**
 * Main function where the program execution begins. Toggle two LEDs

 * when a push button is pressed.
 */
int main(void)
{
    /* Setting the DDRx as output for LEDs */
    DDRB = DDRB | (1<<LED1);
    DDRB = DDRB | (1<<LED2);
    DDRB = DDRB | (1<<LED3);
    DDRB = DDRB | (1<<LED4);
    DDRB = DDRB | (1<<LED5);

    /* PUSH BUTTON*/
    DDRD = DDRD & ~(1<<BTN); //input
    PORTD = PORTD | (1<<BTN); //enable internal pull up

    // Infinite loop
    while (1)
    {

            clear();
            loop_until_bit_is_clear(PIND, BTN);
            PORTB = PORTB | (1<<LED1);
            loop_until_bit_is_clear(PIND, BTN);
            clear();
            PORTB = PORTB | (1<<LED2);
            loop_until_bit_is_clear(PIND, BTN);
            clear();
            PORTB = PORTB | (1<<LED3);
            loop_until_bit_is_clear(PIND, BTN);
            clear();
            PORTB = PORTB | (1<<LED4);
```

```
            loop_until_bit_is_clear(PIND, BTN);
            clear();
            PORTB = PORTB | (1<<LED5);
            loop_until_bit_is_clear(PIND, BTN);
    }

    // Will never reach this
    return 0;
}

void clear() {

    // Pause several milliseconds
    _delay_ms(BLINK_DELAY);

    //Making all outputs low for the LEDs
    PORTB = PORTB & ~(1<<LED1);
    PORTB = PORTB & ~(1<<LED2);
    PORTB = PORTB & ~(1<<LED3);
    PORTB = PORTB & ~(1<<LED4);
    PORTB = PORTB & ~(1<<LED5);

}
```

Knight Rider Application Circuit: