MINISTRY OF SCIENCE AND HIGHER EDUCATION OF THE RUSSIAN Federation
Federal State Budgetary Educational Institution of Higher Education
"Kazan National Research Technical University named after A. N. Tupolev-KAI"
(KNRTU-KAI)
Institute of Computer Technologies and Information Security
Department of Computer Systems

Report №1

«Architecture of embedded systems»

Student: 4167 _____ Edwin G. Carreno
(group number) (signature, data) (full name)

Associate professor of the computer systems' department Daria V. Shirshova

Grade _____

_____
(signature, data)

Kazan 2021

Content

1. Laboratory Practice

a. Control Questions

i. How to create a new project?

Microchip Studio is the newest Integrated Development Environment for programming AVR and SAM microcontrollers. This new version is similar to the AVR Studio 4, so many characteristics were conserved. In order to create a new project you should do the following steps:

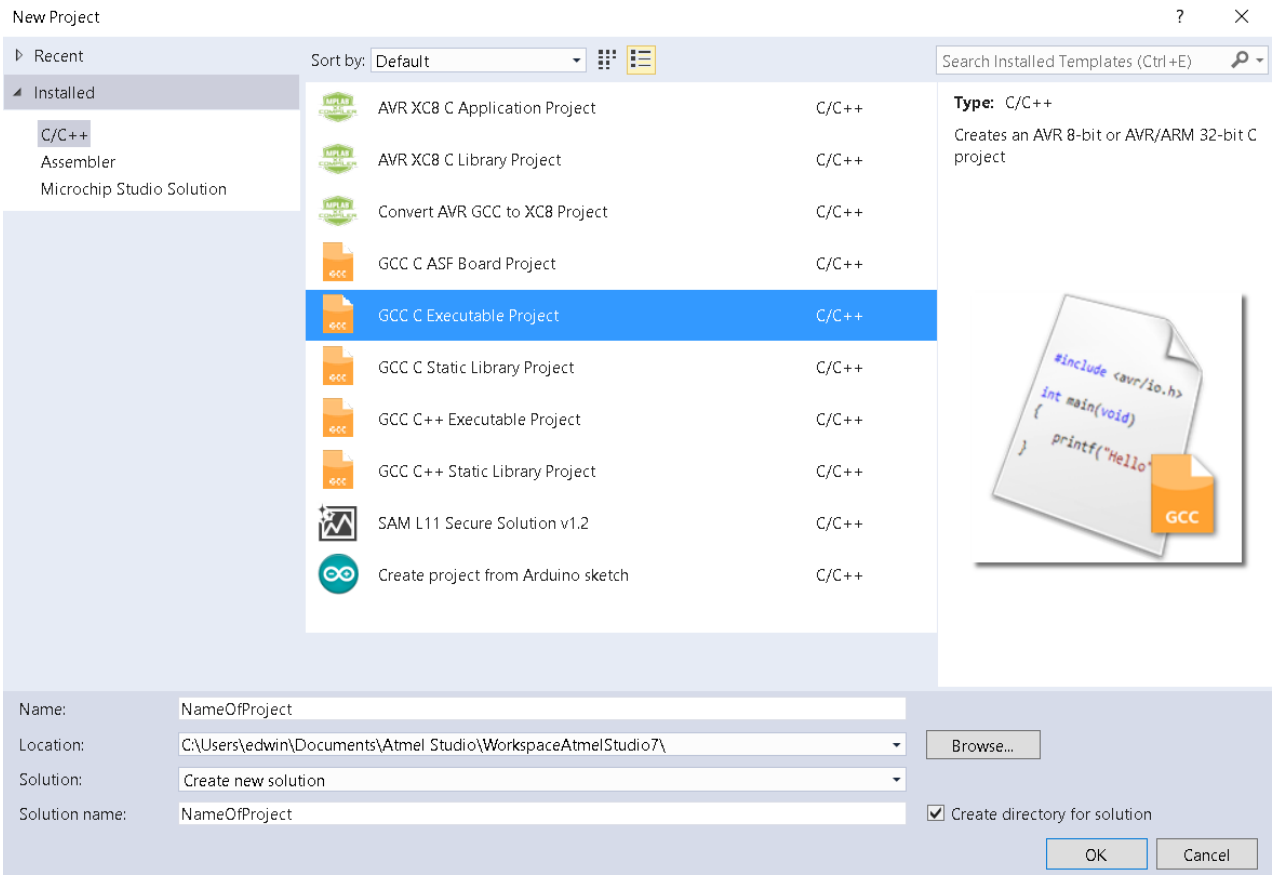

Figure 1. Create a new project in menu File/New Project. Choose the GCC compiler, name the project and define folder destination.

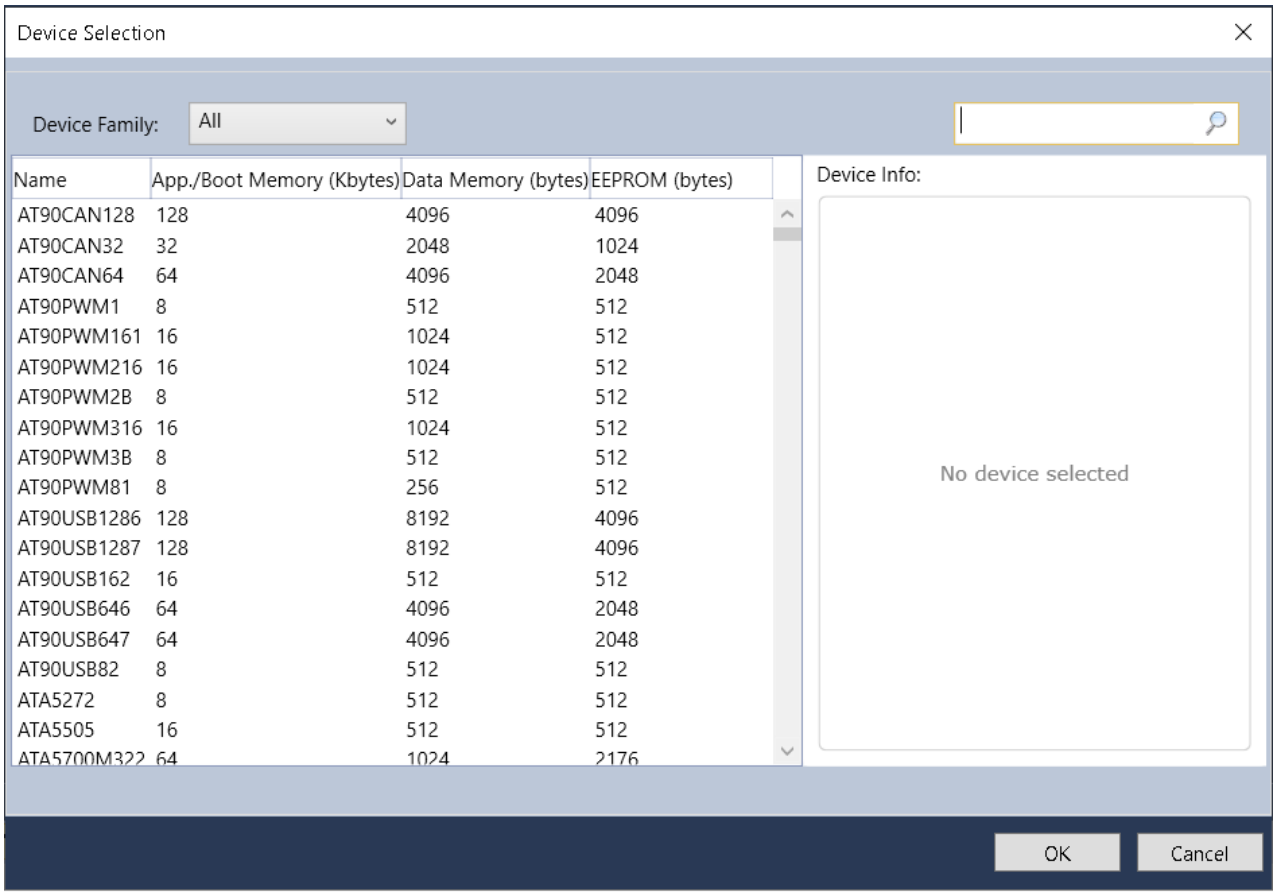

Figure 2. Select the device in list or look for it in the search box.

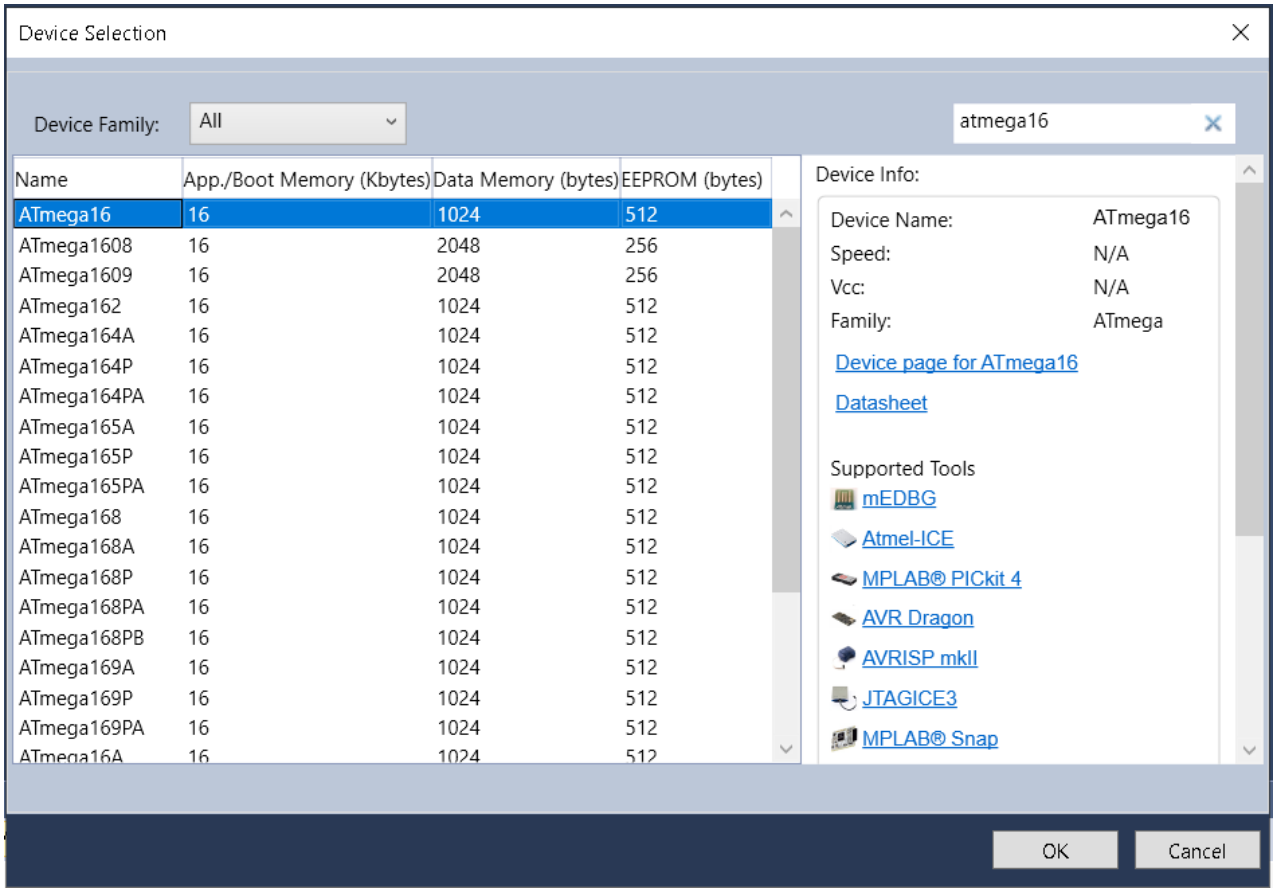

Figure 3. For this laboratory work, the ATmega16 was selected. In the right side, you could see all the supported tools including Simulator at the end of the list.

ii. *How to build a project?*

For building a project just it is needed to press the F7 key or click the option Build Solution in the menu Build. However, when a project configuration is made, it is preferred to use the Rebuild Solution in the same menu. Figure 4 shows the contextual menu for building purposes.

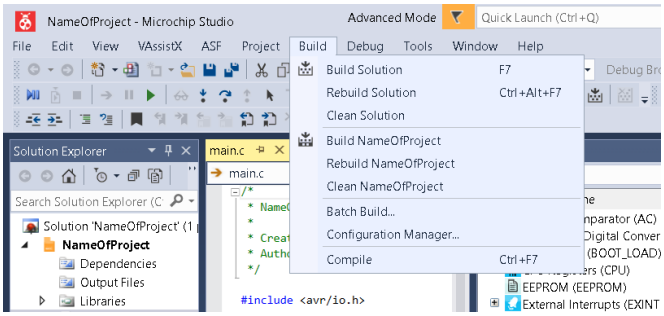

Figure 4. Build project contextual menu.

iii. *How to configure a project?*

Figure 5 shows the menu for setting flags and values that should be used by the compiler. One of those options is the optimizer selector, for simulation purposes it is used the -O0 optimizer.

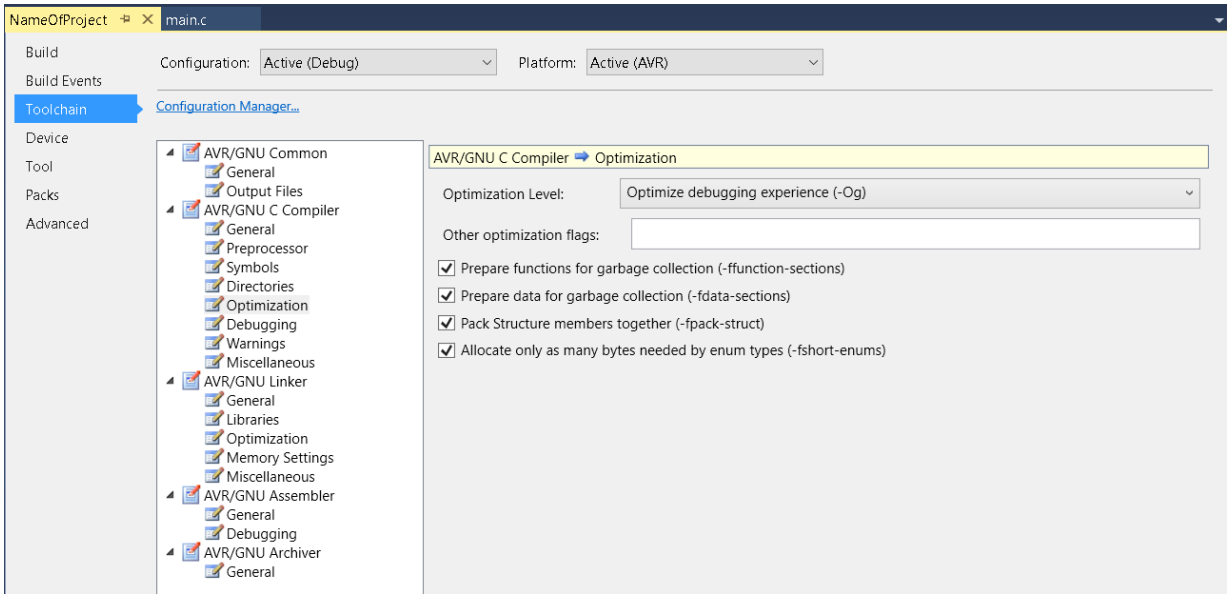

Figure 5. Project settings properties menu, you can find it in menu Project/<name of project>properties.

iv. *How to debug a project?*

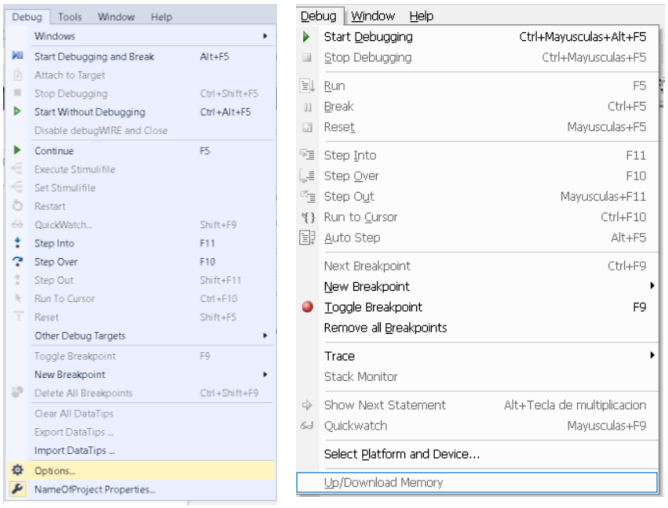

Figure 6. Debug Menu in Microchip Studio(left) and AVR Studio 4 (right). For simulation purposes, in Microchip Studio the correct option is Start Debugging and Break, this option is the same Start Debugging in AVR Studio 4.

b. Circuit Description for Laboratory Work

In the Figure 7, it is shown the schematic proposed for all laboratory tasks given in this document (see sections 2 to 4). Hardware setup is needed before start programming the algorithm to resolve the task. First of all, it is needed to define the use of pull-up or pull-down resistors in the input PORTA. Secondly, every LED should be connected to the output PORTB using resistors in order to reduce the current flow (12mA per pin @ 3.3V) through the entire port. Finally, a LED would be used for simulating the speaker behavior, this hardware setup is sufficient for our purposes in this laboratory practice. Figure 1 shows all the hardware setup for the given tasks, in there, the microcontroller is the ATmega16.

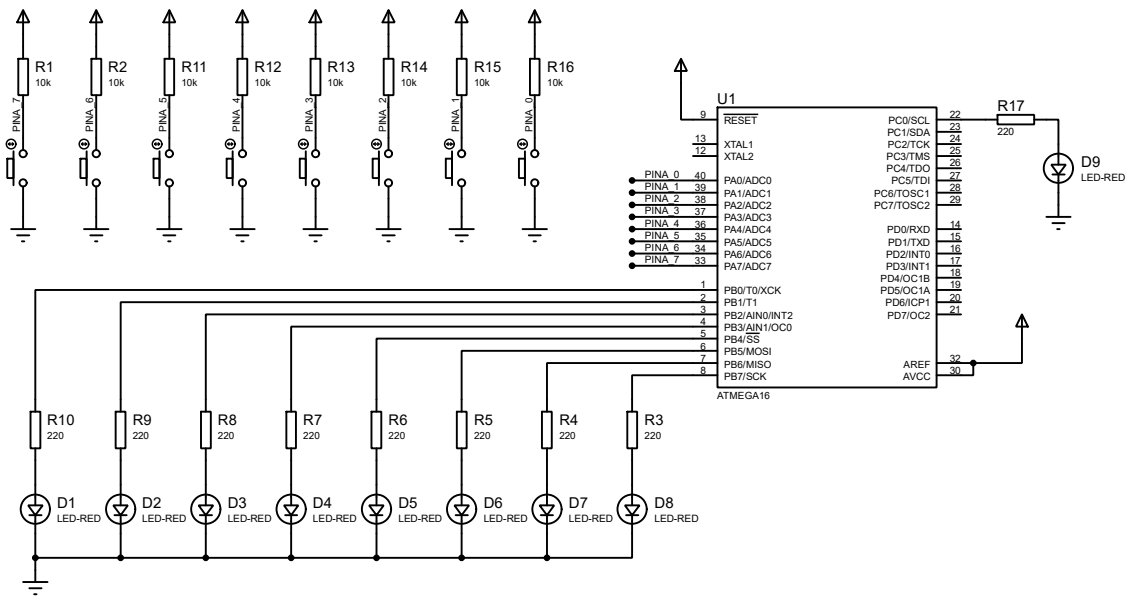

Figure 7. Schematic circuit for the laboratory work (proposed tasks).

2. Development Task 1

Task1: “Modify the program so that it acts as a “dead man's switch”, that is, that an alarm sounds whenever a button is not being pressed. Assume that an alarm is connected to the low pin of PORTC and that it sounds when you drive that line's output high”.

Pseudocode

```
START PROGRAM
    PREPROCESSOR DIRECTIVES
        define pina_end_value 0xFF  // Desire value in PORTA
        define alarm_output 0       // Pin 0 as Output
    FUNCTION init
        SET PORTA direction as input
        SET PORTB as output
        SET PORTB as 0x00
        SET PORTC pin0 direction as output
        SET PORTC pin0 as 0b0
    FUNCTION main
        currentState: 0
```

```
init()
WHILE(1) // infinite loop
    currentstate:= read PORTA
    PORTB:= currentstate
    IF currentState != pina_end_value
        PORTC:= PORTC or 0x01
    ELSE
        PORTC: PORTC and (not(0x01))
```

Code Implementation

```
/*
*       Dead Man's switch
*
*       This program outputs a signal whenever a button is not being pressed.
*       PORTB shows what are the current pressed buttons, this configuration
*       serves as a physical inspection or for debugging purposes.
*
*       Author: Edwin Carreño Lozano
*
*/

#include <avr/io.h>
#include <stdint.h>

#define PINA_END_VALUE 0xFF
#define ALARM_OUTPUT 0  // 7...0, 7 => MSB and 0 => LSB

void init(){
        DDRA = 0x00;  // PORTA used entirely for input
        //PORTA = 0xFF; // PORTA, Pull-up activation for the entire port.

        DDRB = 0xFF;  // PORTB used entirely for output
        PORTB = 0x00; // PORTB outputs a LOW value for safety purposes

        DDRC |= (1 << ALARM_OUTPUT); // PORTC, configuring ONLY the first bit as output
        PORTC &= ~(1 << ALARM_OUTPUT); // PORTC, configuring just the pin 0 with value 0
}

int main(){
        uint8_t thisState = 0x00;
        init();

        while(1){
                thisState = PINA;
                PORTB = thisState;
                (thisState != PINA_END_VALUE) ? (PORTC |= 0x01) : (PORTC &= ~(1 <<
ALARM_OUTPUT));
        }
}
```

Simulation

Microchip Studio 7.0 (newest version) was used for programming and simulation.

Figure 8. Initialization of variables in the program.

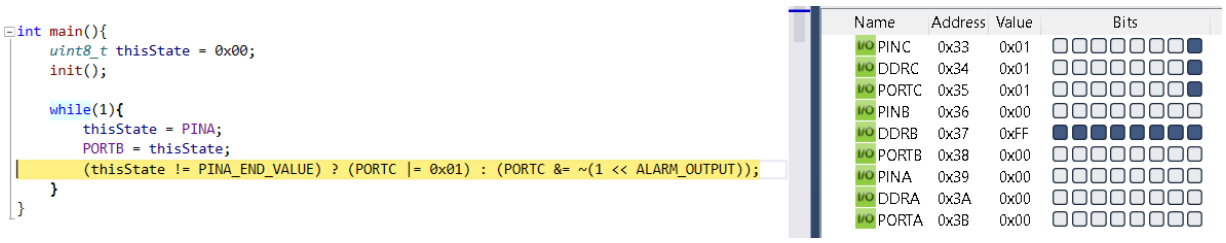

Figure 9. All PORTA is reading zero values, then, the speaker(LED in PINC_0) goes to one(1).

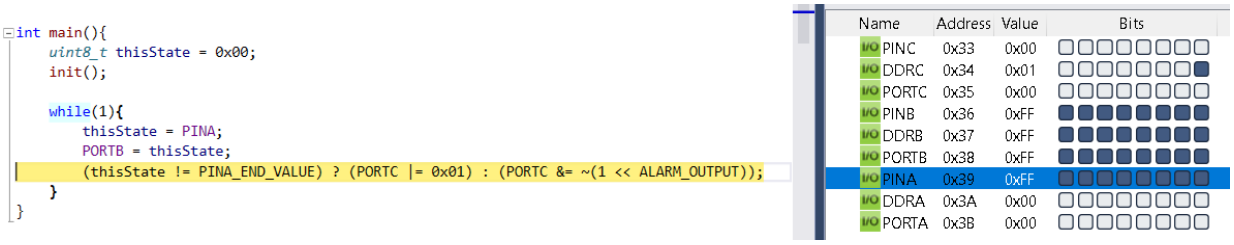

Figure 10. All PORTA is fully connected to one(1), then speaker signal goes zero(0).

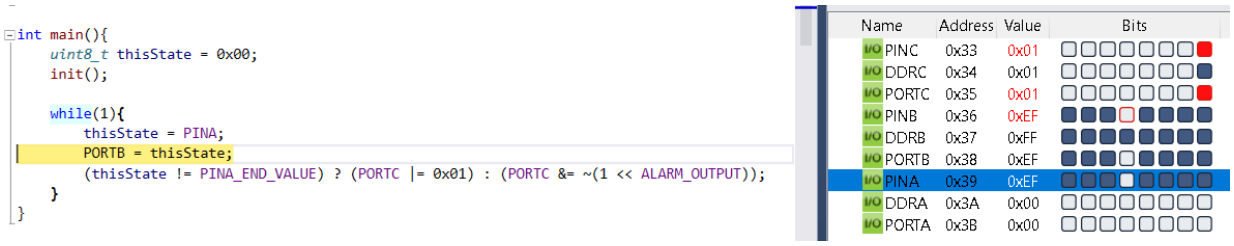

Figure 11. Speaker signal goes high if at least there is one unpressed button.

3. *Development Task 2*

Task 2**:** "Modify the program to detect when a button is pressed when one are more buttons are already down, and sound the alarm when this occurs".

Pseudocode

```
START PROGRAM
    PREPROCESSOR DIRECTIVES
        define pina_end_value 0xFF  // Desire value in PORTA
        define alarm_output 0       // Pin 0 as Output
    FUNCTION init
        SET PORTA direction as input
        SET PORTB as output
        SET PORTB as 0x00
        SET PORTC pin0 direction as output
        SET PORTC pin0 as 0b0

    FUNCTION main
        current_state, previous_state, changed:= 0
        init()
        WHILE(1) // infinite loop
            current_state:= read PORTA
            changed:= current_state xor prev_State
            previous_state:= current_state
            IF (current_state == pina_end_value) or (changed == 0)
                SET PORTC pin0 as 0
            ELSE IF (changed != 0)
                SET PORTC pin0 as 1
                DELAY 500ms
            previous_state:= current_state
```

Code Implementation

```
#include <avr/io.h>
#include <stdint.h>
#include <util/delay.h>

#define F_CPU 8000000UL
#define PINA_END_VALUE 0xFF
#define ALARM_OUTPUT 0  // 7…0, 7 => MSB and 0 => LSB

void init(){
        DDRA = 0x00;  // PORTA used entirely for input
        PORTA = 0xFF; // PORTA, Pull-up activation for the entire port.

        DDRB = 0xFF;  // PORTB used entirely for output
        PORTB = 0x00; // PORTB outputs a LOW value for safety purposes

        DDRC |= (1 << ALARM_OUTPUT); // PORTC, configuring ONLY the first bit as output
        PORTC &= ~(1 << ALARM_OUTPUT); // PORTC, configuring just the pin 0 with value 0
```

```
}

int main(){
        uint8_t thisState = 0x00, prevState = 0x00, changed = 0x00;
        init();

        while(1){
                thisState = PINA;
                changed = thisState ^ prevState;

                if((thisState == PINA_END_VALUE) || (changed == 0)){
                        PORTC &= 0xFE;
                }
                else if(changed != 0){
                        PORTC |= 0x01;
                        //_delay_ms(500); // sounds by 0.5 Seconds
                }
                prevState = thisState;
        }
}
```

Simulation

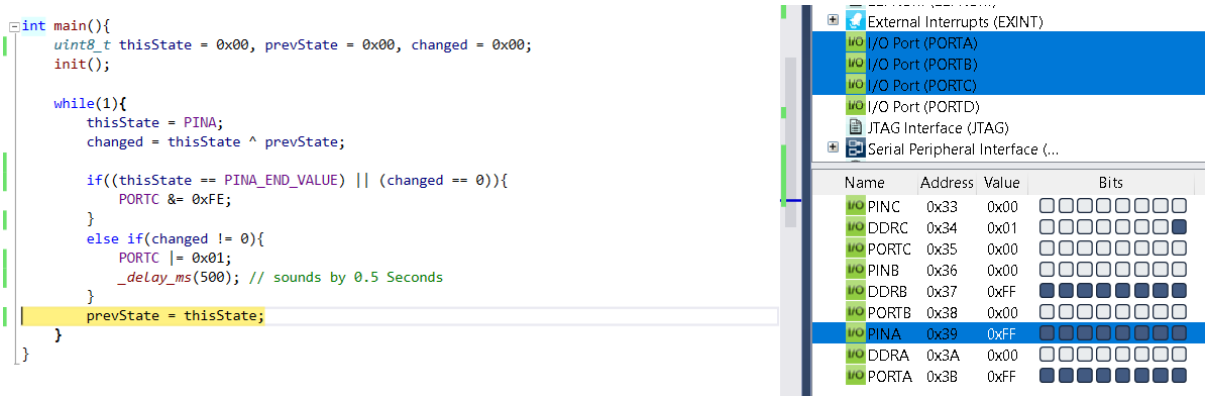

Figure 12. Speaker signal goes zero(0), all input PORTA is reading 0xFF.

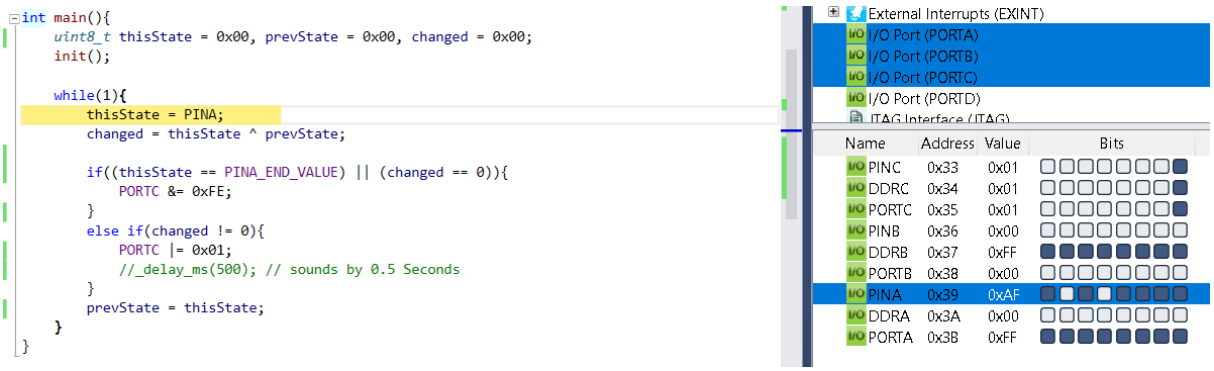

Figure 13. Speaker signal goes high (1), if any input is detected in a low state(0).

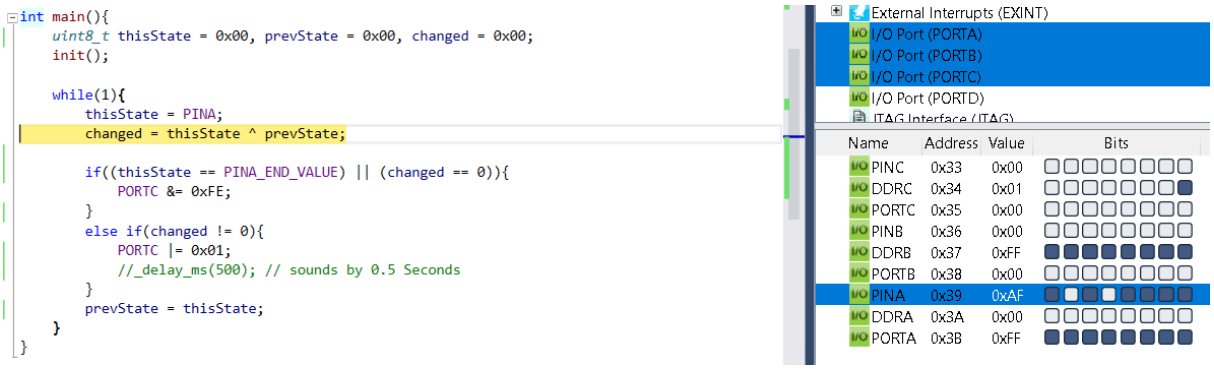

Figure 14. After 500ms, the speaker signal goes zero until a new transition is detected.

4. Development Task 3

Task 3: "Use the existing stimulus file to drive the same sequence of LEDs flashing as output on **PORTB**. When any button is pressed on **PORTA**, the corresponding LED should be lit for the duration of the press, and turned-off when released".

Pseudocode

```
START PROGRAM
    PREPROCESSOR DIRECTIVES
        define pina_end_value 0xFF  // Desire value in PORTA
        define alarm_output 0       // Pin 0 as Output

    FUNCTION init
        SET PORTA direction as input
        SET PORTB as output
        SET PORTB as 0x00
        SET PORTC pin0 direction as output
        SET PORTC pin0 as 0b0

    FUNCTION led_sequence
        CREATE switch cases with all the possible outputs

    FUNCTION main
        button_presses, current_state, previous_state, changed:= 0
        init()
        WHILE(1) // infinite loop
            current_state:= read PORTA
            changed:= current_state xor prev_State // Zero if both values are equal

            IF (changed != 0)
                WHILE(1)
                    current_state:= read PORTA
                    IF current_state == 0
                        break
                    ELSE
                        led_sequence()
                button_presses:= button_presses + 1
                led_sequence(case 0)
                changed:=0
```

Code Implementation

```
#include <avr/io.h>

void init(void){
        DDRA = 0x00;
        //PORTA = 0xFF;

        DDRB = 0xFF;
        PORTB = 0x00;
}

void ledSequence(uint8_t _cycleNumber){
                switch(_cycleNumber){
                case 0:
                        PORTB = 0x00;
                        break;
                case 1:
                        PORTB = 0x01;
                        break;
                case 2:
                        PORTB = 0x00;
                        break;
                case 3:
                        PORTB = 0x01;
                        break;
                case 4:
                        PORTB = 0x00;
                        break;
                case 5:
                        PORTB = 0x01;
                        break;
                case 6:
                        PORTB = 0x03;
                        break;
                case 7:
                        PORTB = 0x01;
                        break;
```

```
                case 8:
                        PORTB = 0x03;
                        break;
                case 9:
                        PORTB = 0x01;
                        break;
                case 10:
                        PORTB = 0x00;
                        break;
                case 11:
                        PORTB = 0x0F;
                        break;
                case 12:
                        PORTB = 0x00;
                        break;
                case 13:
                        PORTB = 0x03;
                        break;
                case 14:
                        PORTB = 0x02;
                        break;
                case 15:
                        PORTB = 0x01;
                        break;
                case 16:
                        PORTB = 0x00;
                        break;
                case 17:
                        PORTB = 0xFF;
                        break;
                default:
                        PORTB = 0x00;
                        break;
        }
}

int main(void)
{
        uint8_t buttonPresses = 0;
        uint8_t thisState = 0, prevState = 0, changed = 0;

        init();

        while(1){
                thisState = PINA;
                changed = thisState ^ prevState;

                if((changed != 0)){
                        while(1){
                                thisState = PINA;
                                if(thisState == 0)
                                        break;
                                else
                                        ledSequence(buttonPresses);
                        }
                        buttonPresses++;
                        ledSequence(0);
                        changed = 0;
                }
        }
}
```

Simulation

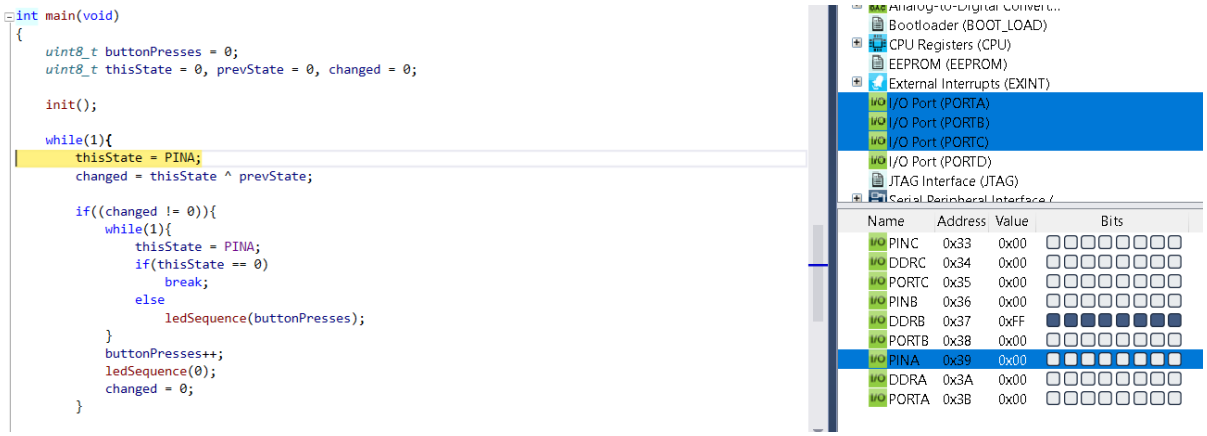

Figure 15. PORTB outputs zero values when there are not buttons pressed.

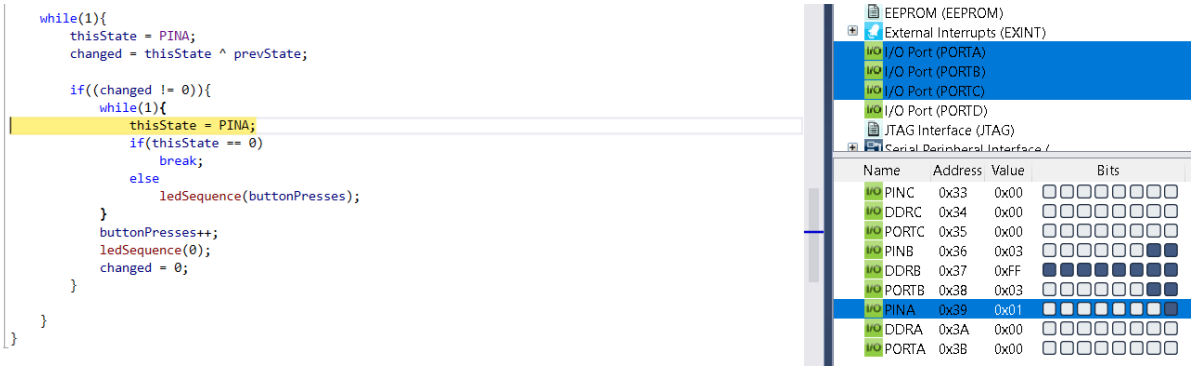

Figure 16. PORTB outputs the value 0x03 when the button is pressed, it removes the value when the button is released.

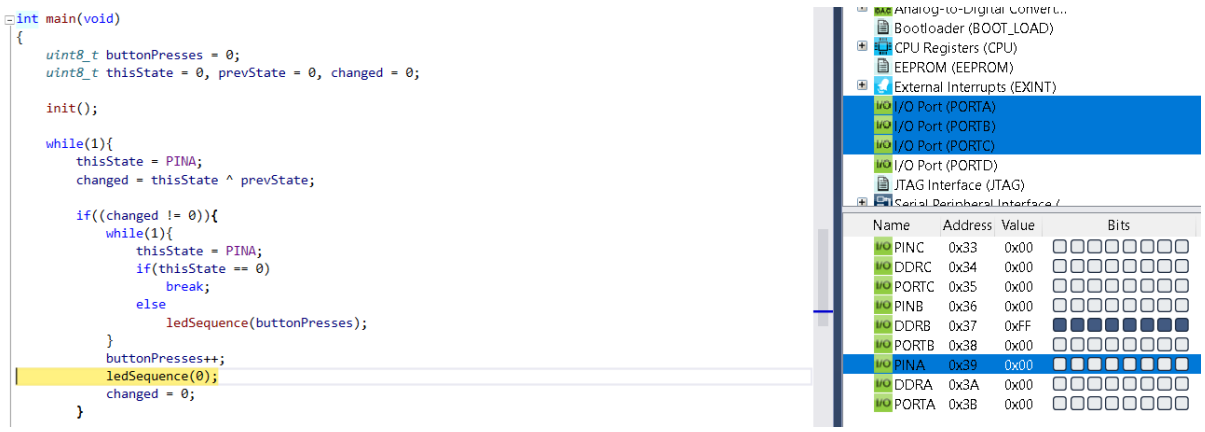

Figure 17. PORTB outputs 0x00 when the button is released.

5. Summary

In this laboratory work, all the tasks were related to GPIO configuration in the ATmega16 using C language, building the project and debugging. Many aspects were not covered such the bouncing related to mechanic effects in switches, those undesired oscillations could not be treated in simulation, however, when the implementation goes physical it is needed to eliminate them. Finally, it was tested the Microchip Studio rather than AVR Studio 4, for now there are not important differences between both software tools.