HOCHSCHULE
RHEIN-WAAL
Rhine-Waal University
of Applied Sciences

LABORATORY

Microcontroller

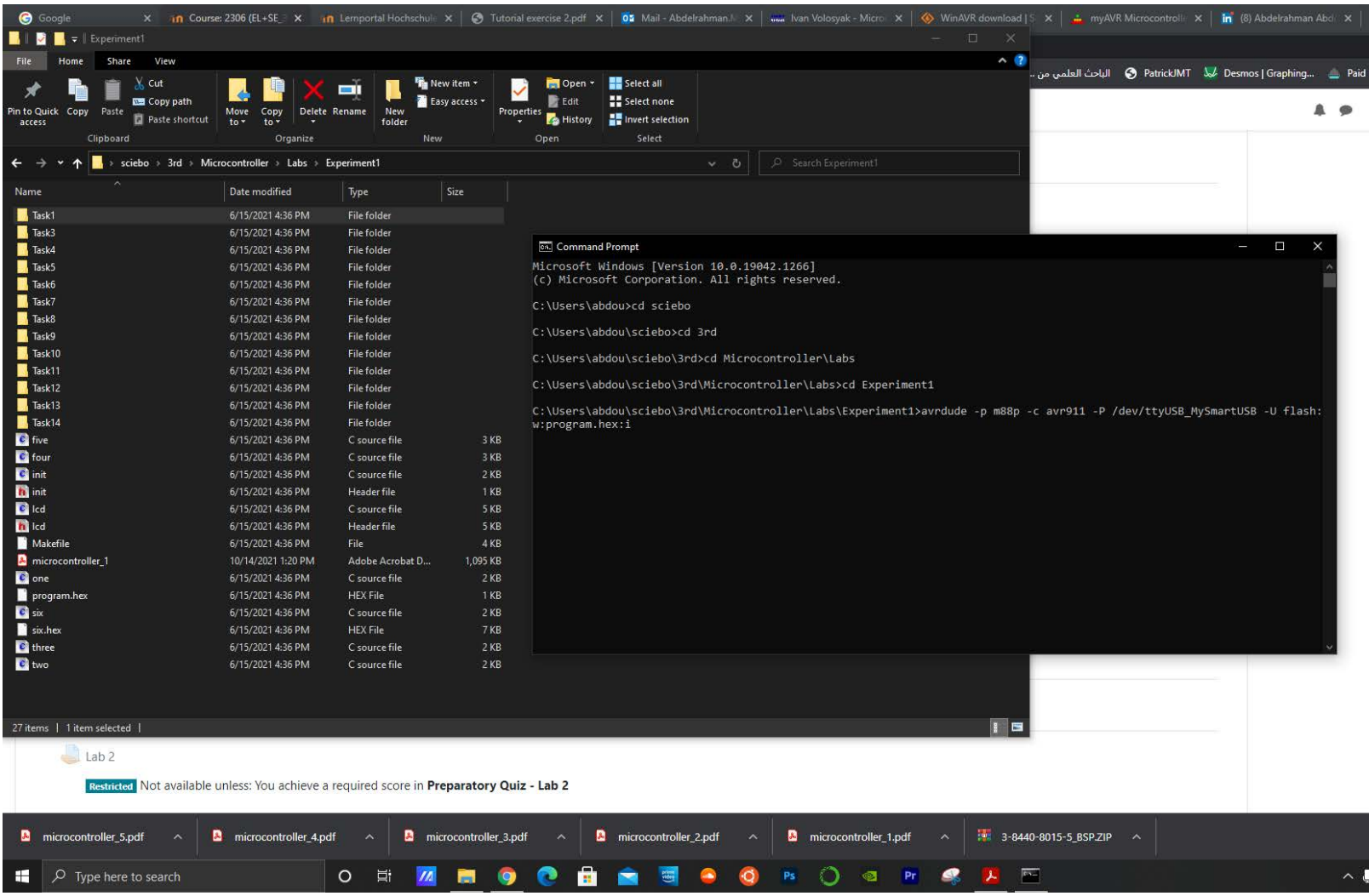
1

EXPERIMENT:

Basic Microcontroller programming

Write your name in every sourcefile you edit and compile the code with your matriculation number (variable in the template). All files should contain all names of the persons who made changes. Do use the @author tag for this (as available in the headline of template files).

Task 1:
To open a terminal, use the shortcut STRG + ALT + T. All files you need are inside a (sub)directory. For all experiments there are pre-written files (templates) available. To go into the directory with all templates enter the command: *cd Templates*. To change into the directory for this experiment enter *cd Experiment1*. This directory already contains the file program.hex.
Write (**and create the screenshot of this step for the lab report**):
avrdude -p m88p -c avr911 -P /dev/ttyUSB_MySmartUSB -U flash:w:program.hex:i
to transfer the file into the microcontroller.

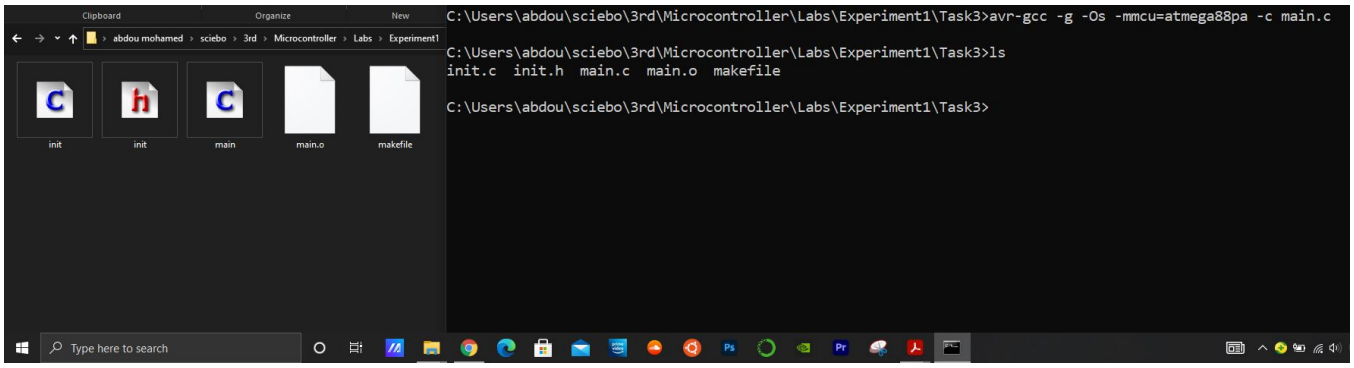


Task 2:
After these connections are done, flash the example programs and test the result. **Describe every program with one-two sentences in the lab report.** Please select **two** most impressive programs (in your opinion).

- *make flash1*
  When Key 1 is pressed once (you don't need to hold), all LEDs are on. Pressing key 2, turns all of them off again.
- *make flash2*
  It's a simulation of a traffic signal, where the red light turns on for 4 or 5 seconds, then the yellow for less than one second, lastly the green light for another 4 or 5 seconds, and it goes back red again.
- *make flash3*
  both keys (1 & 2) are working like car horn, but with different sounds.
- *make flash4*
  It is like someone playing piano for almost 10 seconds.
- *make flash5*
  Nothing has happened at all for me, no errors appeared, and everything went fine. But nothing happened when pressing key 1 or 2.
- *make flash6*
  When potentiometer is at the least value, the frequency is very low (there is almost less than half a second between each sound), the more you increase the resistance the more you increase the frequency.

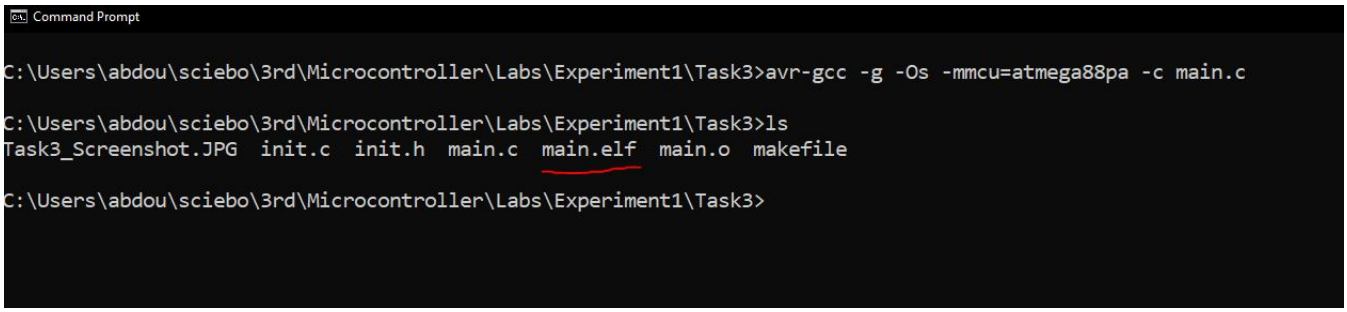
Task 3:
Open the command prompt (STRG + ALT + T), navigate to the folder that contains the .c file (Template/Experiment1/Task3). Type “ls” to see all files in that directory and use “cd” command to change the directory. Compile the .c file using the previously introduces command. Type “ls” again. Which file(s) have been created now? **Describe it in your lab report.** Provide the screenshots (optional). (To have a closer look at the compiler options *avr-gcc –help* can be called.)

Which file(s) have been created now?
= after using "avr-gcc -g -Os -mmcu=atmega88pa -c main.c", The file main.o is created.



Task 4:
Open the command prompt / terminal. Link the .c file (compiled in the previous task) using the command above. Which file(s) have been created now? **Describe it in your lab report.** Provide the screenshots (optional).

Which file(s) have been created now?
= after using "avr-gcc -g -mmcu=atmega88pa -o main.elf main.o", The file main.elf is created.



Task 5:
Open this file in editor or your choice. Provide the screenshot (or the code listing of this hex file) in your lab report.

Task 6:
Edit the main.c with the editor of your choice (vi, nano, gedit). Remove all lines between "// Init START" and "// Init END" and replace it with the function call, just "init();". Now execute this program on the microcontroller. Describe what happens in your lab report.

Firstly I thought nothing has happened except that the .o file, .elf, and .hex files have generated. However, after I read the C code, I realized that I have to connect Port-D2, D3 as inputs, and B1 as an output with the LEDs. The LED which is connected to Port-B1 (output) only turns on when both the other two LEDs are connected to Port-D2, and D3. I could also connect Port D2, and D3 (inputs) to the keys, and the LED which is connected to B1 will turn on only if both keys are pressed.

Task 7:
Complete the following tasks (in the Task7 subfolder):

1. Write the makefile, run the automated compilation by typing ”make all” on the command prompt (navigate to the working directory first).
2. Flash the hex file (flash means: program the controller with the generated HEX-file.).
3. Invert the behaviour of the **red** LED. This means: previously, in the example program the red LED used the AND or OR function. Change this now to the **NOR function**. Then compile the program again and flash it onto the controller.
4. If that works fine, add a second, **yellow** LED, which indicates that **button one and button two** are pressed (**AND function**).
5. Recompile and flash.
6. Use the third, **green** LED, this LED should represent the **XOR function** for the two buttons. In C, there is no logical XOR operator. To get a logical "A xor B" you can use the following logical operation: "!A != !B".
7. Recompile for last time and flash the hex file to the MCU.

```c
const char MtrNum[] __attribute__((__progmem__)) = "29528";

#include <avr/io.h>
#include "init.h"

int main(void){


    // Init START
    init();
    // Init END

    while(1){
        // RED light
        if (!(~PIND & (1 << PD2)) && !(~PIND & (1 << PD3))) // If neither the buttons, which are connected to PORT D2, & D3 are pressed.
            PORTB |= (1 << PB0);    // RED Light. which is connected to PORT B0 will turn on.
        else
            PORTB &= ~(1 << PB0);   // Otherwise, it will turn off.

        if ((~PIND & (1 << PD2)) && (~PIND & (1 << PD3))) // If Both buttons are pressed.
            PORTB |= (1 << PB1); // Yellow light, which is connected to PORT B1 turns on.
        else
            PORTB &= ~(1 << PB1); // Otherwise, it turns off.

        if (!(~PIND & (1 << PD2)) != !(~PIND & (1 << PD3))) // If either button 1 or button 2 is pressed.
            PORTB |= (1 << PB2); // Green LED turns on.
        else
            PORTB &= ~(1 << PB2); // Otherwise it turns off.
            }

    return 0;
}
```

Task 9:
The LED connected to pin B1, should be on if the button connected to pin B0 is pressed and off if the button is not pressed. Compile, flash and test the program.

```
/**
    @author Abdelrahman Mostafa
    @author 29528
    @version 0.1
    @file init.c
    @brief Simple logic function with LEDs and Buttons
*/

const char MtrNum[] __attribute__((__progmem__)) = "29528";

#include <avr/io.h>
#include "init.h"

int main(void){

    // After editing init.c file to make the input comes from PORT-B0!
    init();

    while(1){
        if ((~PINB & (1 << PB0)))
            PORTB |= (1 << PB1);
        else
            PORTB &= ~(1 << PB1);

    }

    return 0;
}
```

Task 10:
Edit the last program to toggle the LED with a button push (toggle operates just like a light switch: once pressed, “on”, by the next button press, it is again “off”).

```c
/**
    @author Abdelrahman Mostafa
    @author 29528
    @version 0.1
    @file init.c
    @brief Simple logic function with LEDs and Buttons
*/

const char MtrNum[] __attribute__((__progmem__)) = "29528";

#define F_CPU 8000000UL

#include <avr/io.h>
#include <util/delay.h>
#include "init.h"

int main(void){

    // After Editing init.c file.
    init();

    while(1){


        if ((~PINB & (1 << PB0)) ){ // If Button at B0 is pressed:
            PORTB ^= (1 << PB1); // LED at B1 turns on

            _delay_ms(70); // Debouncing

            while(~PINB & (1<<PB0)); // Wait Until the button is pressed down:
            _delay_ms(70); // Debouncing
        }
```

Task 11:
To understand PWM write a simple program that enables and disables the LED in an infinite loop when the button is pressed, otherwise the LED should be constantly on.

```c
/**
    @author Abdelrahman Mostafa
    @author 29528
    @version 0.1
    @file main.c
    @brief Analog output via PWM
*/

const char MtrNum[] __attribute__((__progmem__)) = "29528";

#define F_CPU 8000000UL

#include <avr/io.h>
#include <util/delay.h>
#include "init.h"

int main(void){


    init();

    while(1){
        if ((~PIND & (1 << PD2)) ){
            // _delay_ms(100);
            PORTB ^= (1 << PB1);
        } else {
            PORTB |= (1 << PB1);
        }
    }

    return 0;
```

Task 12:
Write a program, which

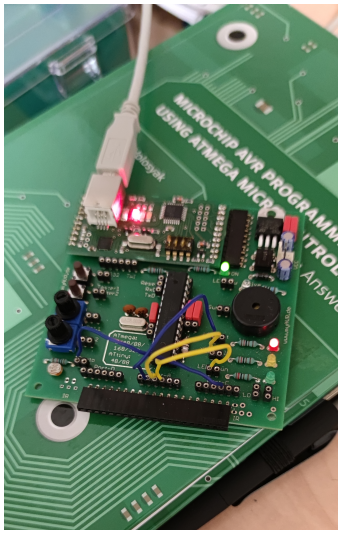
- initialises the ADC, use the free running mode,
- reads the voltage on the potentiometer's wiper,
- outputs the voltage range on the 3 LEDs,
  - green: voltage range 0 – 1.66V,
  - yellow: voltage range 1.66 – 3.32V,
  - red: voltage range 3.32 – 5V.

```
/**
    @author Abdelrahman Mostafa
    @brief Main function
    @return only a dummy to avoid a compiler warning, not used
*/
int main(void){

    // Init
    init();
    // Loop forever
    while (1){

        // Read Pin status of PIN B0
        if (ADCW <= 340) // If volts less than or eqal 1.66V
            PORTB |= (1 << PB3); //Green LED at Port B3 turns on.
        else
            PORTB &= ~(1 << PB3); // Otherwise off.

        if ((ADCW > 340) && (ADCW <= 679)) // If volts maotr than 1.66V and less than or equal 3.32V
            PORTB |= (1 << PB2); // Yellow LED at Port B2 turns on.
        else
            PORTB &= ~(1 << PB2); // Otherwise off.

        if ((ADCW > 679) && (ADCW <= 1023)) // If volts More than 3.32V and less than or equal 5V.
            PORTB |= (1 << PB1); //Red LED at Port B1 turns on.
        else
            PORTB &= ~(1 << PB1); // Otherwise off.

    }
```

Task 13:
In order to use a "real" sensor: connect the photo sensor instead of the potentiometer to the input pin ADC3. Change the illumination condition of this sensor e.g. with your finger (or e.g. with your smartphone light). Is it possible to see all three LEDs on without additional light source? Describe it in your lab report. Disconnect the input pin ADC3 from the photo sensor and simply touch the wire connected to ADC3 with your finger. What happens? Describe it in your lab report and provide a short explanation.

When connected to the sensor:
An initial position (Pic. 1) when my finger is too far from the sensor, the red LED is on (which means the voltage input is between 3.32V and 5V. Moreover, when my finger is more close to the sensor (Pic. 2), with no touch, the red LED is off and yellow LED is on (which mean the input is between 1.66V and 3.32V). Lastly (Pic. 3), when I touch the sensor, the green LED is alone on (which means the input has changed to less than or equal 1.66V).

Pic. 1



Pic. 2

Pic. 3

When disconnected from the sensor, and touched with fingers:
after disconnection and before touching the wire, the green LED is on (which means the input is simple less than 1.66V). When touched with my finger all LEDs are turned on together while the red LED is not always on brightly. I believe the explanation to this is that my body (finger) acts like an AC input, but with high frequency which can not be detected with my eyes. But my body does have a lot of free electrons to share, thats why the red LED (which requires at least 3.32 volts) does not turn on properly. And after some time, when my body is not charged anymore, touching the wire will not make any difference.