B31DD – Embedded Systems

Lab 1

LED, Switch, Timer and Interrupt

SUBMISSION DEADLINE 20 oct 2021

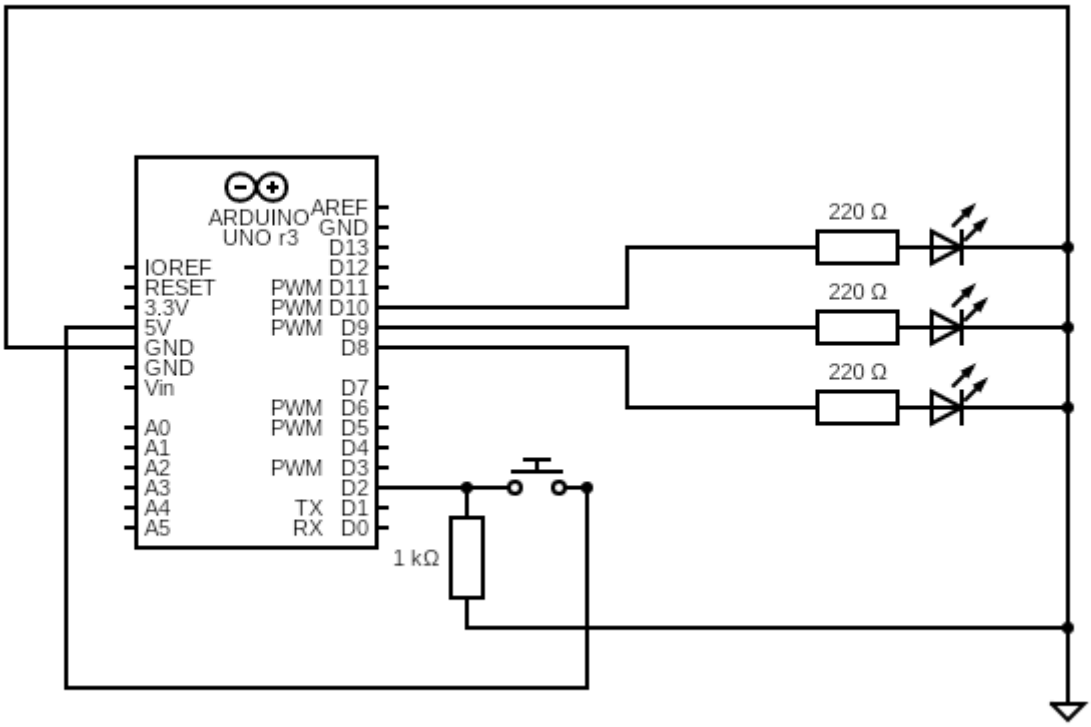
JANNIN Sylvain

H00387879

Msc Robotics

Date of submission: 19 oct 2021

Task 1



Schematic of task 1.

Timer:

I need to have a delay of 2900ms (S+J=29), I call this delay T_{delay}. For this I use Timer1 because this timer is coded on 16 bits and a I choose a prescaler of 1024 which means, *TCCR1B = 0b00000101*. When the delay will be over, I need the timer to be stopped, in this case *TCCR1B = 0*.

The clock frequency is: $f_{clk} = 16 \times 10^6$ Hz. With the prescaler it becomes:

$$f_{pres} = \frac{f_{clk}}{prescalaire} = \frac{16 \times 10^6}{1024} = 15\ 625\ Hz$$

Thanks to the prescaler we have a slower clock, now we calculate the value used to calculate the *TCNT1* value to have a delay of 2,9 seconds.

$$N = T_{delay} \times f_{pres} = 2{,}9 \times 15\ 625 = 45312.5$$

Since we only use integer number, I round N to the closet integer: N = 45313. Furthermore, this value is good since I use a timer coded on 16 bits. In the end, we want to count to 45313 with our timer with an overflow of 2^{16}.

$$TCNT1 = 2^{16} - 1 - N = 65536\ -1\ -\ 45313 =\ 20\ 222$$

For the assembly we can't write this value directly, we have to separate this into two different registers: *TCNT1L* and *TCNT1H*. For this we write this value in binary and then we separate the low and the high value:

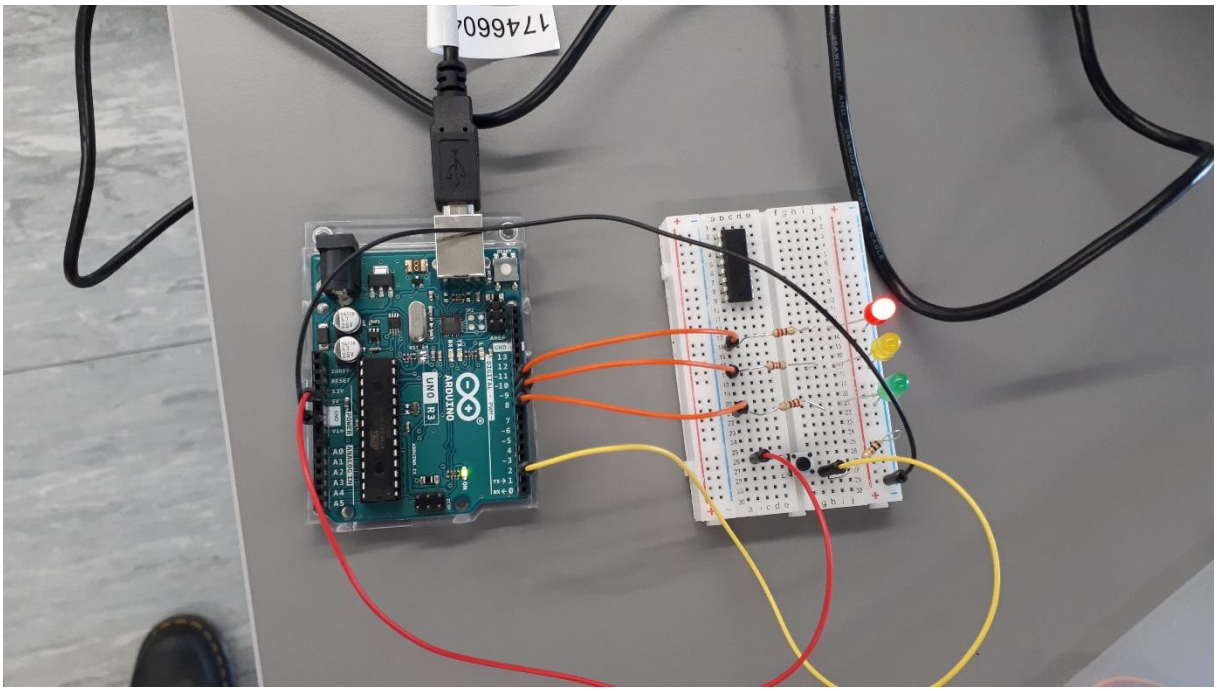
$$TCNT1 = 0d20\ 222 = 0b0100\ 1110\ 1111\ 1110$$

$$TCNT1L = 0b1111\ 1110$$

$$TCNT1H = 0b0100\ 1110$$

To summarize:

- *TCNT1 = 20 222*
  - *TCNT1L = 0b1111 1110*
  - *TCNT1H = 0b0100 1110*
- *TCCR1B:*
  - On: *TCCR1B = 0b00000101*
  - Off: *TCCR1B = 0*

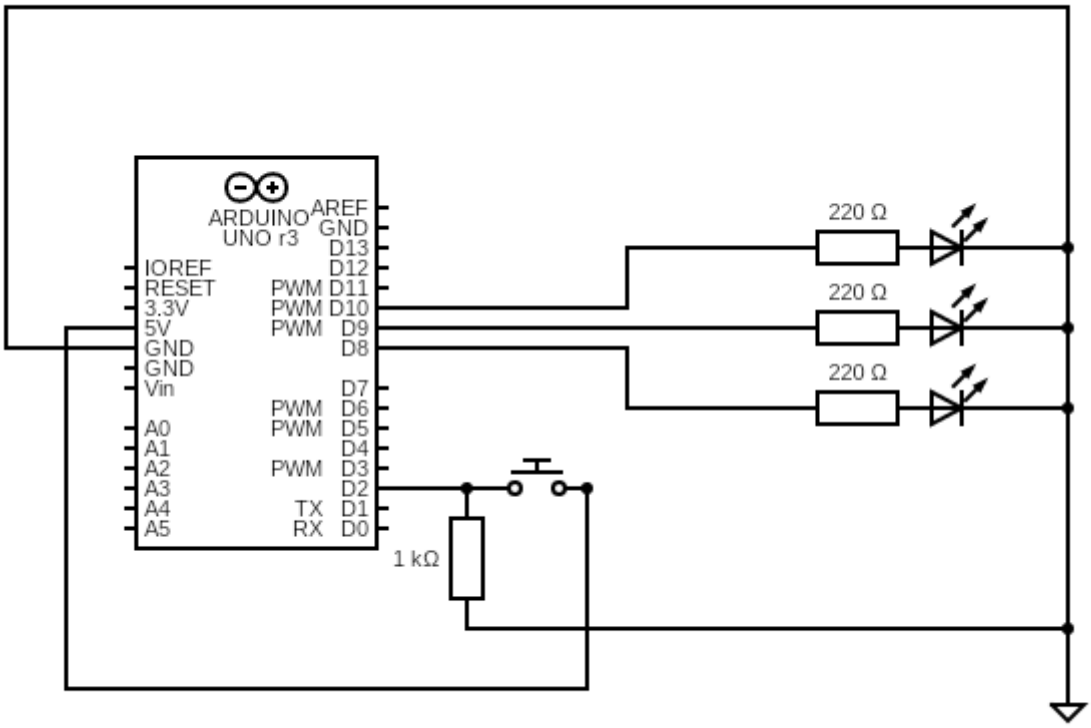


Pictures of the circuit.

Code:

```
; lab1_task1__JANNIN_Sylvain.asm
;
; Created: 13/10/2021 14:56:39
; Author : sylva
;

//Main program

LDI R16,0b111    ; 0xFF
OUT DDRB,R16    ;Set PB as an output
SBIS PIND,2             ;check switch
RJMP PC-1               ;Jump this line and execute following instructions
SBI PORTB,0             ;Switch on LED1
CALL my_delay
CBI PORTB,0             ;Switch off LED1
SBI PORTB,1             ;Switch on LED2
CALL my_delay
CBI PORTB,1             ;Switch off LED2
SBI PORTB,2             ;Switch on LED3
CALL my_delay
CBI PORTB,2             ;Switch off LED3


; We have to load the 20 222 value, 0d20 222 = 0b0100 1110 1111 1110
my_delay: LDI R21, 0b11111110; 1111 1110 --> the low part
                LDI R22, 0b01001110; 1111 1110 --> the high part
                STS TCNT1L,R21 ;load timer1
                STS TCNT1H, R22;
                LDI R20,0x00
                STS TCCR1A,R20 ; normal mode
                LDI R20,0b101;
                STS TCCR1B,R20 ; prescaler of 1024

AGAIN:  SBIS TIFR1,TOV1 ;  skip next line if TOV1 is set
                RJMP AGAIN
                LDI R20,0x00
                STS TCCR1B,R20 ;timer stop
                LDI R20,0x01
                STS TIFR1,R20 ;clear flag TOV1 in the
```

Task2



Schematic of task 1.

Timer:

I need to have a delay of 2900ms (S+J=29), I call this delay T_{delay}. For this I use Timer1 because this timer is coded on 16 bits and a I choose a prescaler of 1024 which means, *TCCR1B = 0b00000101*. When the delay will be over, I need the timer to be stopped, in this case *TCCR1B = 0*.

The clock frequency is: $f_{clk} = 16 \times 10^6$ Hz. With the prescaler it becomes:

$$f_{pres} = \frac{f_{clk}}{prescalaire} = \frac{16 \times 10^6}{1024} = 15\,625\ Hz$$

Thanks to the prescaler we have a slower clock, now we calculate the value used to calculate the *TCNT1* value to have a delay of 2,9 seconds.

$$N = T_{delay} \times f_{pres} = 2{,}9 \times 15\,625 = 45312.5$$

Since we only use integer number, I round N to the closet integer: N = 45313. Furthermore, this value is good since I use a timer coded on 16 bits. In the end, we want to count to 45313 with our timer with an overflow of 2^{16}.

$$TCNT1 = 2^{16} - 1 - N = 65536\ -1\ -\ 45313 =\ 20\,222$$

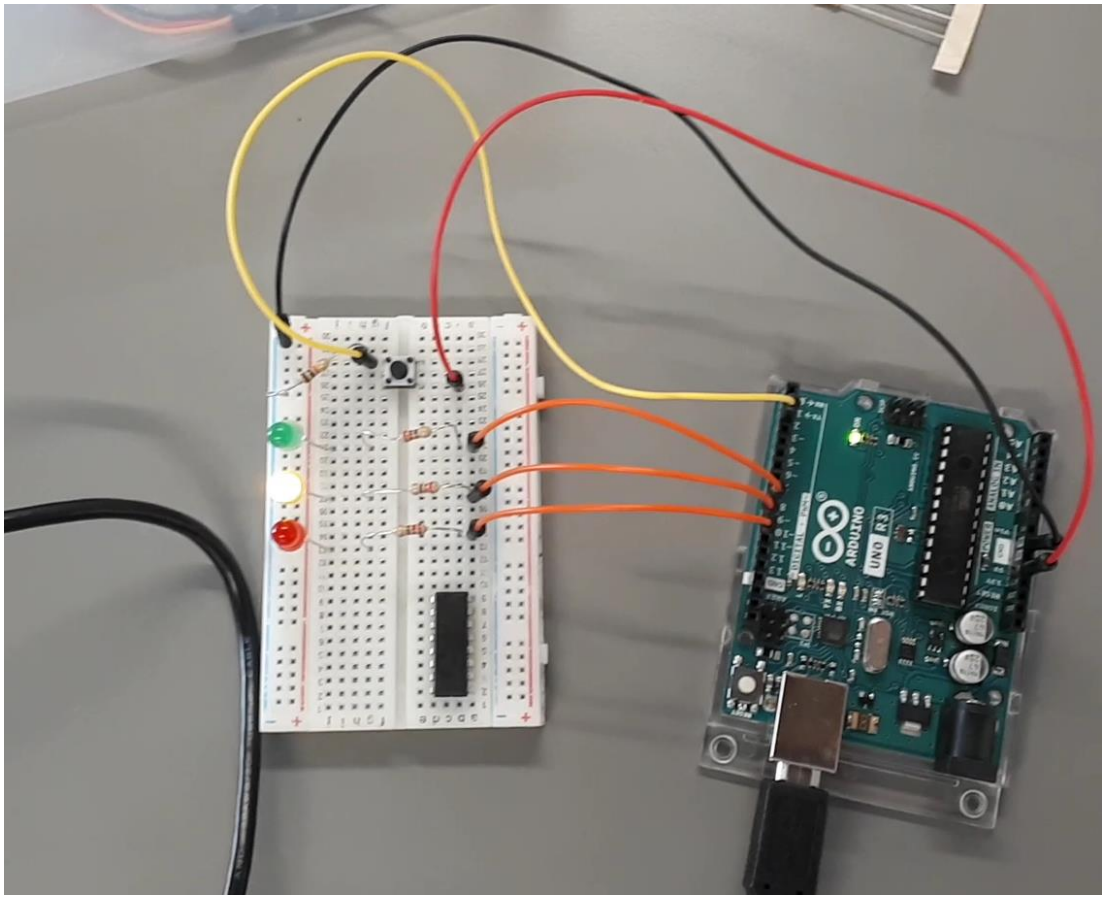
Interruption:

I use the external interrupt 0 which is connected to PD2 and I want the interrupt to be triggered on falling edge so *EICRA = 0b010*. I also need to enable the external interrupt, for this we need to assign *EIMSK* a value:

$$EIMSK = (1 << INT0)$$

To summarize:

- *TCNT1 = 20 222*
- *TCCR1B:*
  - On: *TCCR1B = 0b00000101*
  - Off: *TCCR1B = 0*
- *EICRA = 0b010*
- *EIMSK = (1 << INT0)*

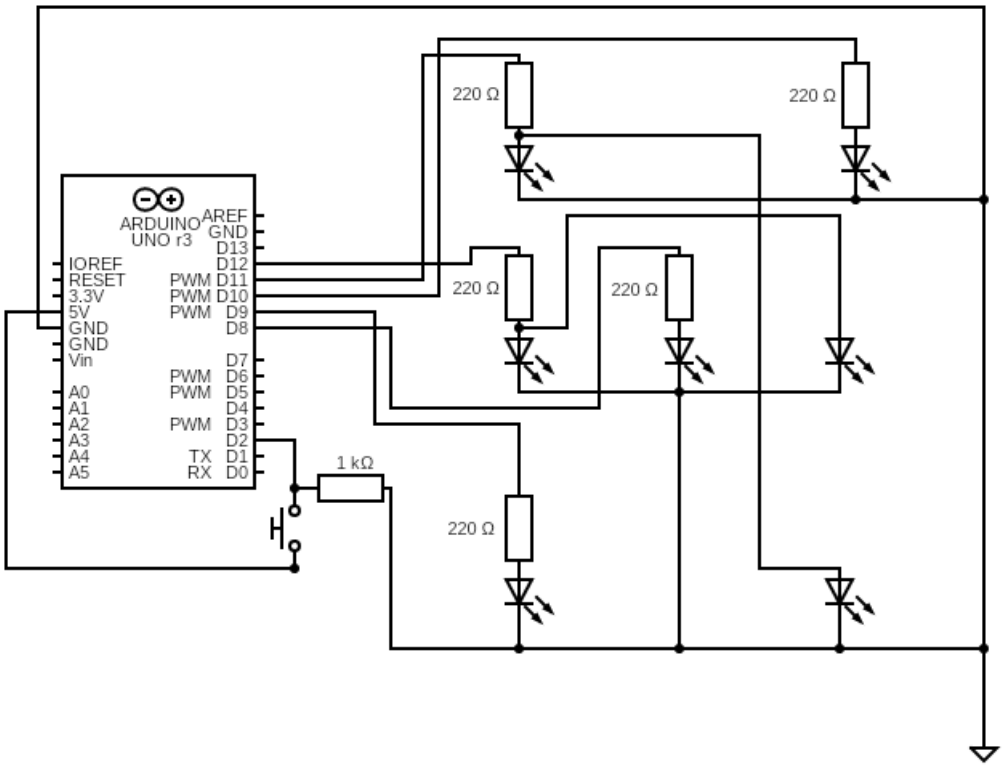


Pictures of the circuit.

Code:

```
/*
 * lab1_task2__JANNIN_Sylvain.c
 *
 * Created: 30/09/2021 09:00:37
 * Author : sylva
 */

#include <avr/io.h>
#include <avr/interrupt.h>
#define F_CPU 16000000UL

void my_delay()
{
        int n = 45313 ; // value that depends on the "n" value // slide ??
        TCNT1  = 65536 -1 - n; // TCNT1 = TCNT1H + TCNT1L
        TCCR1A = 0x00;
        TCCR1B |= (1<<CS10); // no need to set CS11 to 0 because it is 0 by default
        TCCR1B |= (1<<CS12); // prescaler of 1024

        while((TIFR1&(1<<TOV1))==0)
        {}

        TCCR1B = 0; //timer stops
        TIFR1 = (1<<TOV1);
}


ISR (INT0_vect)
{
        PORTB = 0b0000001; //turn on 1 led
        my_delay();                // wait 2.9seconds

        PORTB = 0b0000010;      //turn on one other LED and turn off The first one
        my_delay();

        PORTB = 0b0000100;
        my_delay();

        PORTB = 0x00; // turn off all the LED
}



int main(void)
{
        //# ------------- Define in and outputs ------------- #
        DDRB |=1<<PB0 | 1<<PB1 | 1<<PB2; // set PB0, PB1 and PB2 as output
        DDRD &=~(1<<PD2);       //set PD2 as input because it has the INT0 interrupt


        //# ------------- Define register ------------- #
        //Define External Interupt
        EIMSK = (1 << INT0); //enable external interrupt 0 // does it also work if I write : "EIMSK =
0b01;" ?
        EICRA = 0b010; // interrupts on falling edge
        sei(); // Global Interrupt enable

    while (1)
    {
    }
}
```

Task 3



Schematic of task 3.

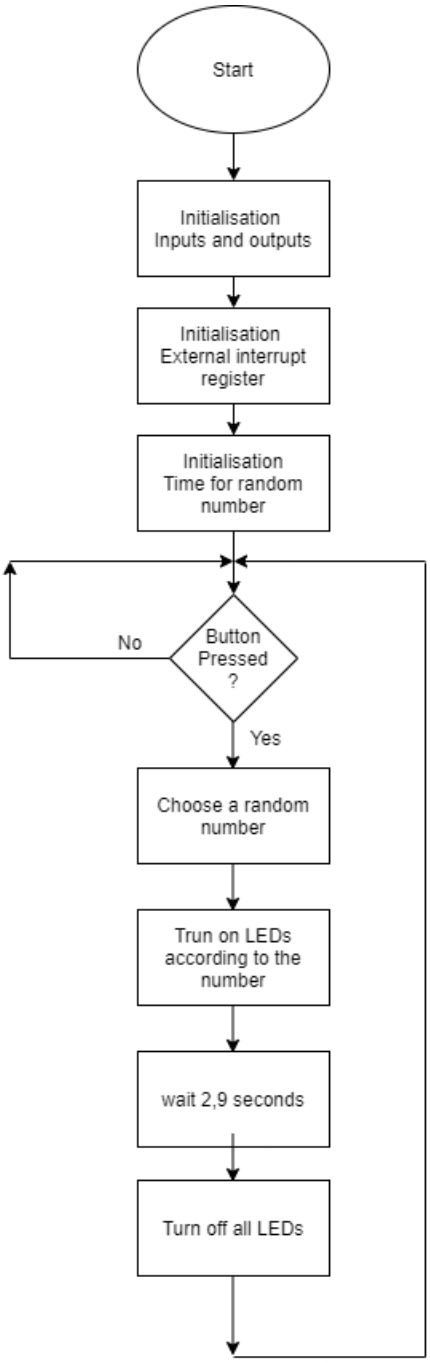
Summary of the program:

To implement the dice, I reuse the function I created in task 2 for the delay and for in external interruption. I also generate a random number from 1 to 6 thanks to time.h which is used to know which LED the program turns on or not.

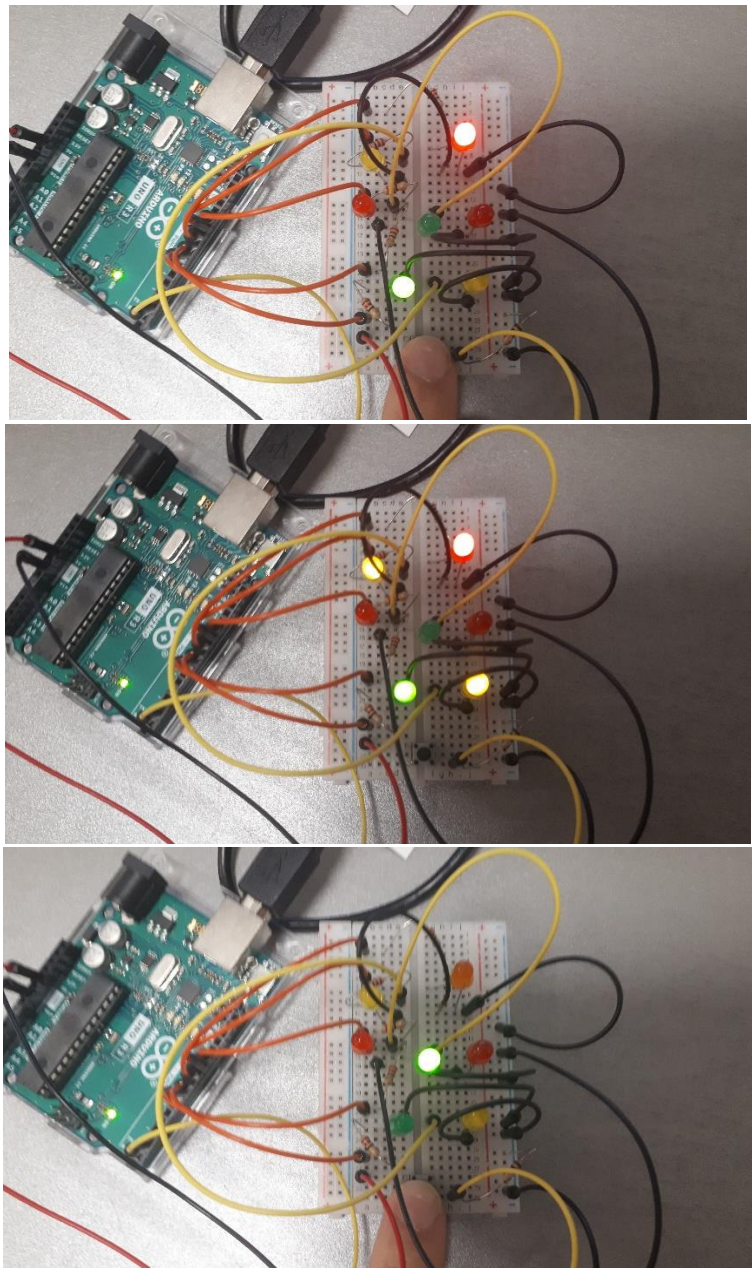
Like in task 2 there is a button associated to the external interruption, when the user released the button, it creates an interruption which activates the dice. There is a pull-down resistor linked to the button to have a better signal.

Once the interruption is activated, the program creates a random number from 1 to 6 which is used in the function *dice_choice*.

In the function *dice_choice*, according to the number given it turns LEDs to show the number for 2.9 seconds. After 2.9 seconds, all the LEDs are turned off and the external interruption is over.



Task 3 flowchart.



Pictures of the circuit with 3 different LEDs configuration.

Code

```
/*
 * lab1_task2__JANNIN_Sylvain.c
 *
 * Created: 30/09/2021 09:00:37
 * Author : sylva
 */

#include <avr/io.h>
#include <avr/interrupt.h>
#include <time.h>
#define F_CPU 16000000UL



void dice_choice(int dice)
{
	switch(dice)
	{
		case(1):
			PORTB = 0b00000001;	// do a 1 with LEDs
			break;

		case(2):
			PORTB = 0b00000110; // do a 2 with LEDs
			break;

		case(3):
			PORTB = 0b00000111; // do a 3 with LEDs
			break;

		case(4):
			PORTB = 0b00001110; // do a 4 with LEDs
			break;

		case(5):
			PORTB = 0b00001111; // do a 5 with LEDs
			break;

		case(6):
			PORTB = 0b00011110; // do a 6 with LEDs
			break;
		default:
			break; // in case the number is not good, do nothing
	}

}


void my_delay()
{
	int n = 45313; // value that depends on the "n" value
	TCNT1  = 65536 -1 - n; // TCNT1 = TCNT1H + TCNT1L
	TCCR1A = 0x00;
	TCCR1B |= (1<<CS10); //// no need to set CS11 to 0 because it is 0 by default
	TCCR1B |= (1<<CS12);

	while((TIFR1&(1<<TOV1))==0)
	{}

	TCCR1B = 0; //timer stops
	TIFR1 = (1<<TOV1);
}

int my_random_dice()
{
	int nb;
	nb = 0;
```

```
        nb = rand () %6; //generate a random number from 0 to 5
        nb = nb +1;
        return (nb);
}

ISR (INT0_vect)
{
        int dice;
        dice = my_random_dice();
        dice_choice(dice);
        my_delay();
        PORTB = 0x00; // reset
}



int main(void)
{
        //# ------------- Define in and outputs ------------- #
        DDRB = 0b00011111;
        DDRD &=~(1<<PD2);       //set PD2 as input because it has the INT0 interrupt


        //# ------------- Define register ------------- #
        //Define External Interupt
        EIMSK = (1 << INT0);
        EICRA = 0b010; // interrupts on falling edge
        sei(); // Global Interrupt enable

        srand (time (NULL));    //initialise time for the random number

    while (1)
    {
    }
}
```