Complex of Technical Schools in Grudziadz

Automation and Robotics Laboratory

Laboratory instruction:

Introduction to Arduino

M.Sc. Eng. Marcin Jabłoński

I. What is Arduino ...

Arduino - a programming platform for embedded systems based on a simple Open Hardware project for microcontrollers mounted on a single printed circuit, with built-in input / output support and a standardized programming language

What exactly is **Arduino** ? I will try to describe it in the **simplest words** :
Arduino is a board with a microcontroller and all additional electronics that helps / facilitates the programming and use of Uc in various projects. I have already written about what a microcontroller is (link to the article: "Microcontroller ..."). The heart of the system is the AVR microcontroller from **Atmel** (specifically ATmega8, ATmega168, ATmega328, and ATmega1280 and ATmega 2560). Most of the boards contain a 5V voltage ***regulator*** , 16 MHz quartz ***resonator , input / output pins*** and additional elements improving work (e.g. PWM pins). The microcontroller in the platform is pre-programmed with a bootloader, which simplifies the transfer of the program to the flash memory (this is the "hard drive" of our microcontroller) of the chip, compared to other devices where an external programmer is most often needed.
The easiest way to program Arduino is with the help of a specially prepared program: **Arduino IDE** . It is a cross-platform application written in Java.
The environment is designed to be friendly to hobbyists and non-developers on a daily basis. The IDE includes a code editor with functions such as syntax highlighting or automatic code indentation, and allows you to compile and upload the program to the Arduino board. Usually, there is no need to edit the Makefile files (it is a file that allows us to upload the program to the microcontroller - you don't need to know anything about it yet) or run programs from the command line, which makes the work much easier.
Officially, Arduino has released **several versions of** its platform, so that you can choose the right one for you. Below is a list of available models:

1. Arduino serial, programmed via DB9 serial interface and using the ATmega8 chip
2. Arduino Extreme, programmed via the USB interface and provided with the chip ATmega8
3. Arduino Mini, a miniature version of Arduino that uses the ATmega168 surface-mounted chip, or Atmega328
4. Arduino Nano, an even smaller version of Arduino, powered by USB with a surface-mounted chip ATmega168
5. Arduino LilyPad, a minimalist design using a surface mounted chip ATmega168
6. Arduino NG, programmed via the USB interface and using the chip ATmega8
7. Arduino NG plus, programmed via the USB interface and equipped with the ATmega168 chip
8. Arduino BT, programmed wirelessly via the Bluetooth interface using the ATmega168 chip
9. Arduino Diecimila, with USB interface and Atmega168 chip in the housing DIL28
10. Arduino Duemilanove ("2009"), using Atmega168 (Atmega328 in newer versions) powered from an external power supply or via USB

11. Arduino Mega, equipped with a surface-mounted ATmega1280 chip, thanks to which it has more memory and pins inputs / outputs.
12. Arduino Uno, version with programmable USB interface and chip Atmega328
13. Arduino Leonardo, version with the ATmega32u4 chip, which is the heart of Arduino and is responsible for communication USB
14. Arduino Ethernet, version with ATmega328 chip, Ethernet interface instead of USB and card reader microSD
15. Arduino Yún, version with ATmega32u4 chip and built-in chip WiFi
16. Arduino Esplora, version with Atmega32U4 chip, board designed as a base for the controller console
17. Arduino Robot, version made of two round boards, each equipped with an ATmega32u4 processor

I would consider the most popular versions of the Arduino: **UNO, Leonardo, Mega, Nano, Mini** - I would classify them as the most famous.
However, the original tiles are not everything, the Internet "tears up" the mass of clones, that is, systems identical or similar to the official version. The main difference is the price;
Original price:
~ 150 PLN Price of the maple: ~ 50 PLN… And the components. Usually, inferior quality parts are used for the production of cheap clones.

II. The first simple program on Arduino ...

Connecting the LED to a microcontroller and its actuation is usually one of the first tasks for adepts to the art of microcontroller programming. And rightly so, because harnessing the potential of modern microcontrollers should start with the simplest challenges.

The article Arduino programming language - the only right choice presents the most frequently quoted example of the first microcontroller program, i.e. the control of the LED diode attached to the microcontroller's input / output line. The test setup for this example is shown in **photo 1** . **Listing 1** shows a program that causes the LED attached to the line to go on and off every 1 second 13.

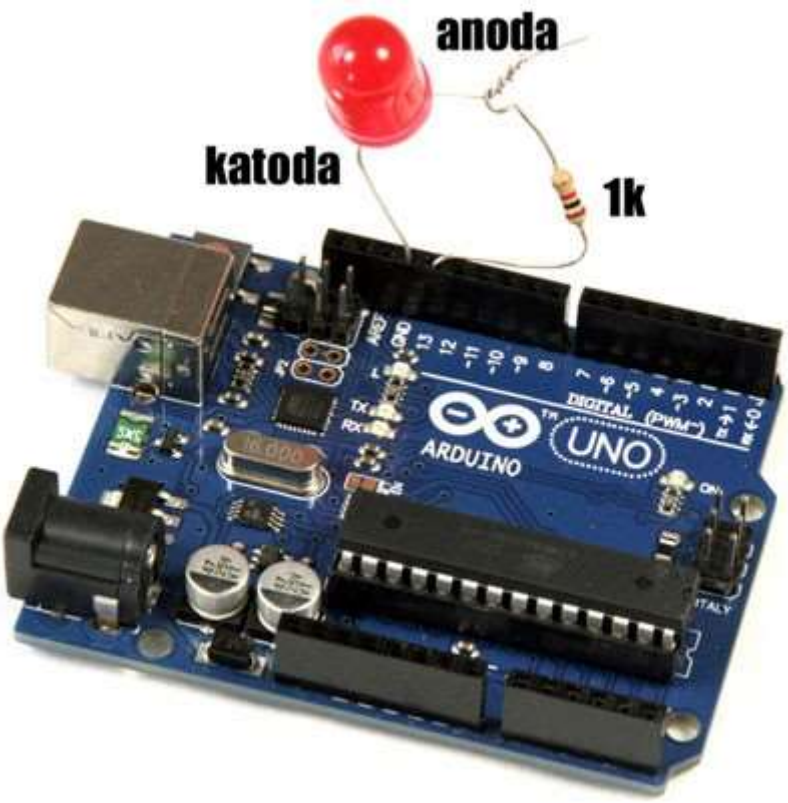

Photo 1. Attaching the LED to the Arduino Uno kit

Listing 1. The LED flashing program for the circuit shown in Fig. 1

```
const int LED = 13;         // an LED is attached to the line 13

void setup ()               // function configuring
{
  pinMode (LED, OUTPUT);    // set the LED line as output
}

void loop ()                // loop infinite
{
  digitalWrite (LED, HIGH); // set the bit
  delay (1000);             // 1 s delay (1000 ms)
  digitalWrite (LED, LOW); // reset the bit
  delay (1000);             // 1 s delay (1000 ms)
}
```

Now we will set ourselves a slightly more ambitious task - 8 LEDs will be attached to the microcontroller and we will get the effect of a moving light point on them. The electrical diagram of our hardware platform is shown in **Figure 2** , while **Figure 3** shows how to make the connections. The drawings, i.e. simplified technical documentation of the project, were made using the fritzing program. The assembled system is shown in **photo 4** . During the assembly, the following were used:

– Arduino Uno kit with ATmega328 microcontroller (AVR family of the company Atmel)

– USB cable for programming and powering the Arduino Uno (eg. CAB_USB_AB)

– contact plate (e.g. PPS0400)

- 8 LEDs (e.g. LED-AL-30R-D00300-60)

– 8 resistors 680 Ohm

– 9 wires from the kit CAB_M-M (65-Rainbow)

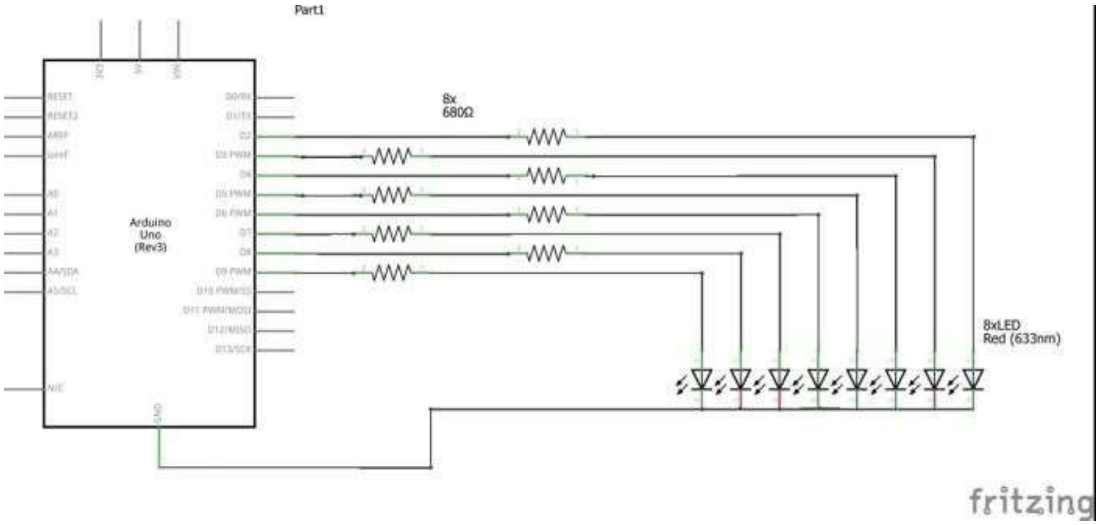

Fig. 2. Connecting 8 LEDs to Arduino Uno - wiring diagram

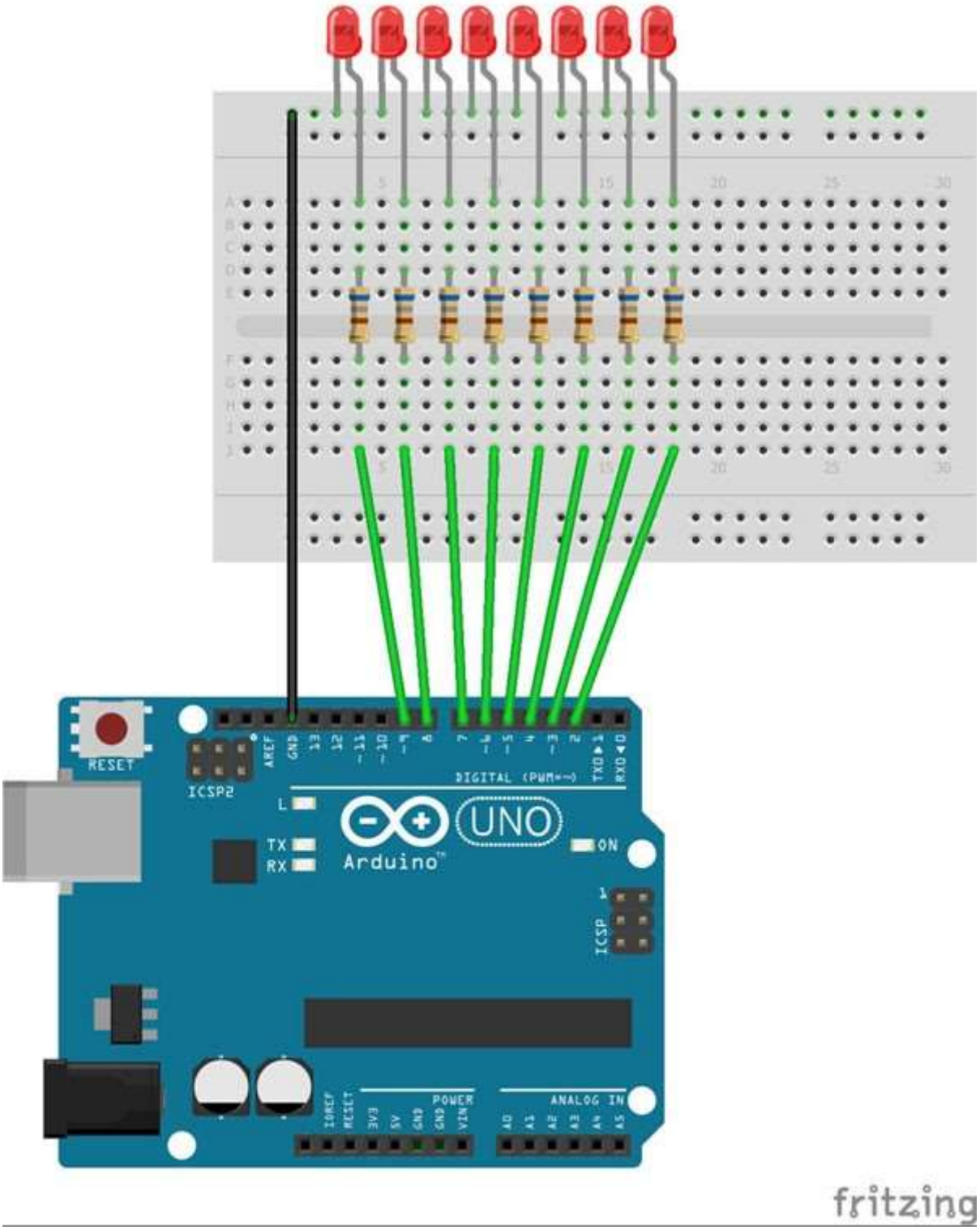

Fig. 3. Connecting 8 LEDs to Arduino Uno - making connections

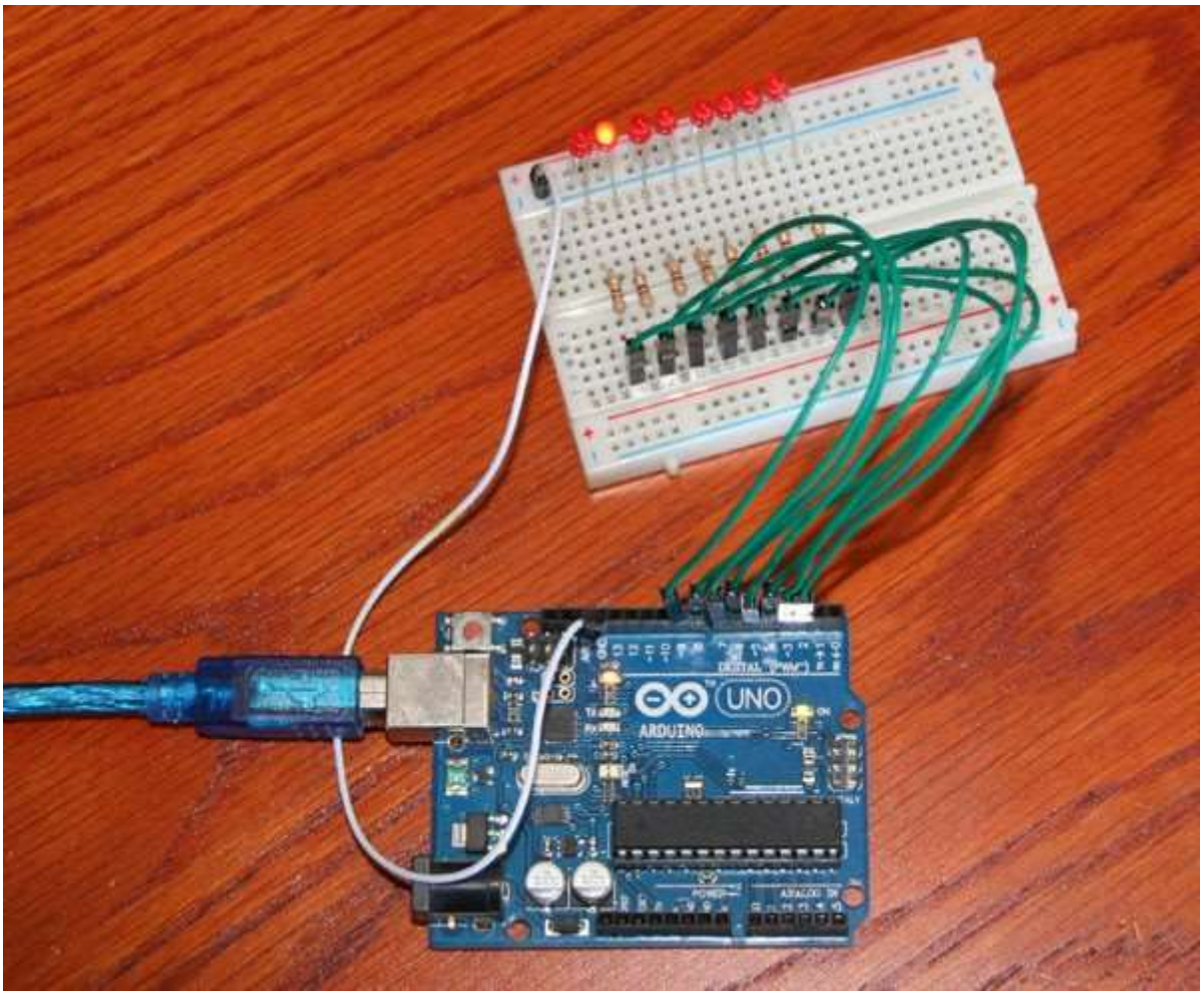

Photo 4. View of the assembled system

The circuit shown in Figure 2 can also be built a little easier (but also more expensive) with the use of the KAmodLED8 module shown in **photo 5** .

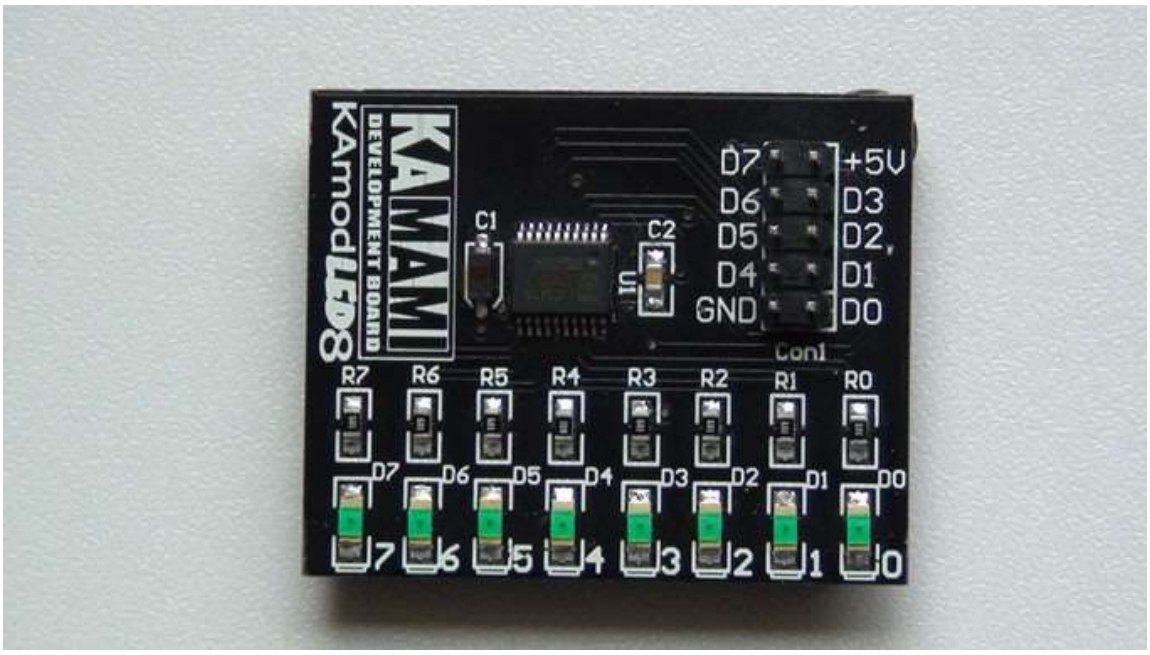

Photo 5. The KAmodLED8 module can also be used in the project implementation

After the circuit is assembled as shown in Figure 3, it should be connected to a personal computer with a USB cable and you can start experimenting with the program. **Listing 2** shows a program that allows you to obtain the effect of a moving light point.

Listing 2. The program controlling the line of LEDs for the circuit shown in Fig. 4 (prog_01.ino)


```
#define LED_GET 2
#define LED_KON 9

void setup ()
{
  int and;
  // lines 2 ... 9 as outputs
  for (i = LED_POCZ; i & lt; = LED_KON; i ++)
    pinMode (i, OUTPUT);
}

void loop ()
{
  int and;
  // moving point
  for (i = LED_POCZ; i & lt; = LED_KON; i ++)
  {
    digitalWrite (i, HIGH);
    delay (200);
    digitalWrite (i, LOW);
twenty

  }

}
```

In the first two lines of the program, using the #define preprocessor directive, two constants were defined: LED_POCZ and LED_KON with the values 2 and 9, respectively. The LEDs are connected to the I / O line of the Arduino Uno with the numbers: 2, 3, 4, 5, 6, 7, 8 and 9. In the setup () function, these lines are set to the output mode using the function:

```
pinMode(i, OUTPUT);
```

The defined constants LED_POCZ and LED_KON correspond to the first and last used line in / out. In the program, access to these lines is realized by means of a for loop, which uses these constants to improve the program readability:

```
for (i = LED_POCZ; i & lt; = LED_KON; i ++)
  pinMode (i, OUTPUT);
```

After setting the required configuration of the I / O line, the desired effect, i.e. a moving light point, is obtained in a loop:

```
for (i = LED_POCZ; i &lt;= LED_KON; i++)
{
  digitalWrite( i, HIGH );
  delay(200);
  digitalWrite( i, LOW );
}
```

In each iteration of the above loop, another LED lights up. Each LED lights up for 200 ms.
By changing the value of the delay () function, you can easily change the speed at which the light point moves.

At this point, it's worth noting that changing the I / O lines to which LEDs are attached basically demolishes our program. Unfortunately, sometimes when designing a printed circuit board, it turns out that changing the order in which these diodes are attached would result in a much easier printed circuit board. There is an easy way to protect yourself from this type of inconvenience. It is enough to "address" the LEDs using a properly prepared table, as shown in **Listing 3** . The leds [] table contains the numbers of the I / O lines to which the LEDs are connected. In the case of the layout shown in Figure 3 the array of leds [0] should look like this:

```
1 int leds[] = {2, 3, 4, 5, 6, 7, 8, 9};
```

In the program, the reference to the first LED in the light bar is done by calling the first array element, i.e. leds [0], which will be replaced during the program operation by the reference to the I / O line number 2.

If for some reason it turns out that the LEDs should be connected in a different way, e.g. by replacing the first LED with the last one (Figure 6), it is enough to change the contents of the leds table [] without modifying the rest of the program:

```
1 int leds[] = {9, 3, 4, 5, 6, 7, 8, 2};
```

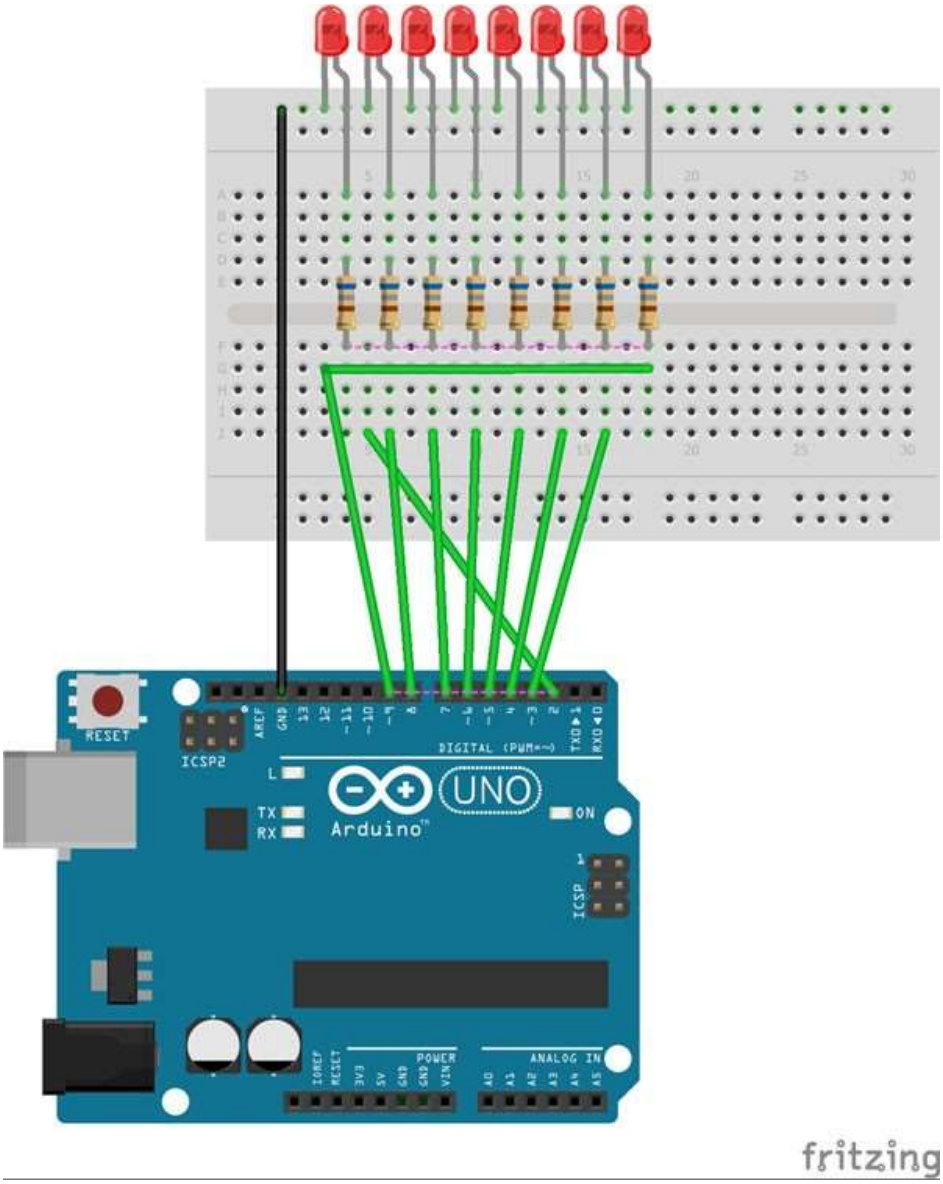

Fig. 6. View of connections. Connecting 8 LEDs to Arduino Uno - making connections

Listing 3. The program controlling the line of LEDs for the circuit shown in Fig. 4 (prog_02.ino)

```
int leds [] = {2, 3, 4, 5, 6, 7, 8, 9};
#define LEDS_SIZE 8

void setup ()
{
  int and;
  // lines from leds as outputs
  for (i = 0; i & lt; LEDS_SIZE; i ++)
    pinMode (leds [i], OUTPUT);
}

void loop ()
{
  int and;
  // moving point
  for (i = 0; i & lt; LEDS_SIZE; i ++)
  {
    digitalWrite (leds [i], HIGH);
    delay (200);
    digitalWrite (leds [i], LOW);


  }

}
```