**TECHNICAL UNIVERSITY OF MOLDOVA**

**FACULTY OF COMPUTERS INFORMATICS AND MICROELECTRONICS**

**DEPARTMENT OF SOFTWARE ENGINEERING AND AUTOMATION**

**Laboratory work 1**

**Subject: User Interaction - Serial STDIO**

Done by: Nichita CHIHAI

st. gr. FAF-213

Verified by: Dumitru MORARU,

university lecturer

**Chişinău – 2024**

**1 THE TASK OF THE LABORATORY WORK**

1. Configure the application to work with the STDIO library through the serial interface for text exchange via the terminal.

2. Design an application based on the MCU that would receive commands from the terminal through the serial interface to set the state of an LED.

- "led on" for turning on and

- "led off" for turning off.

- The system should respond with text messages confirming the command.

- For text exchange via the terminal, use the STDIO library.

**2 PROGRESS OF THE WORK**

For this laboratory work, I utilized the Arduino IDE, a software designed for writing and compiling code for the Arduino development board, along with Proteus, a software used for electronic design simulation.

**2.1 Main functions description**

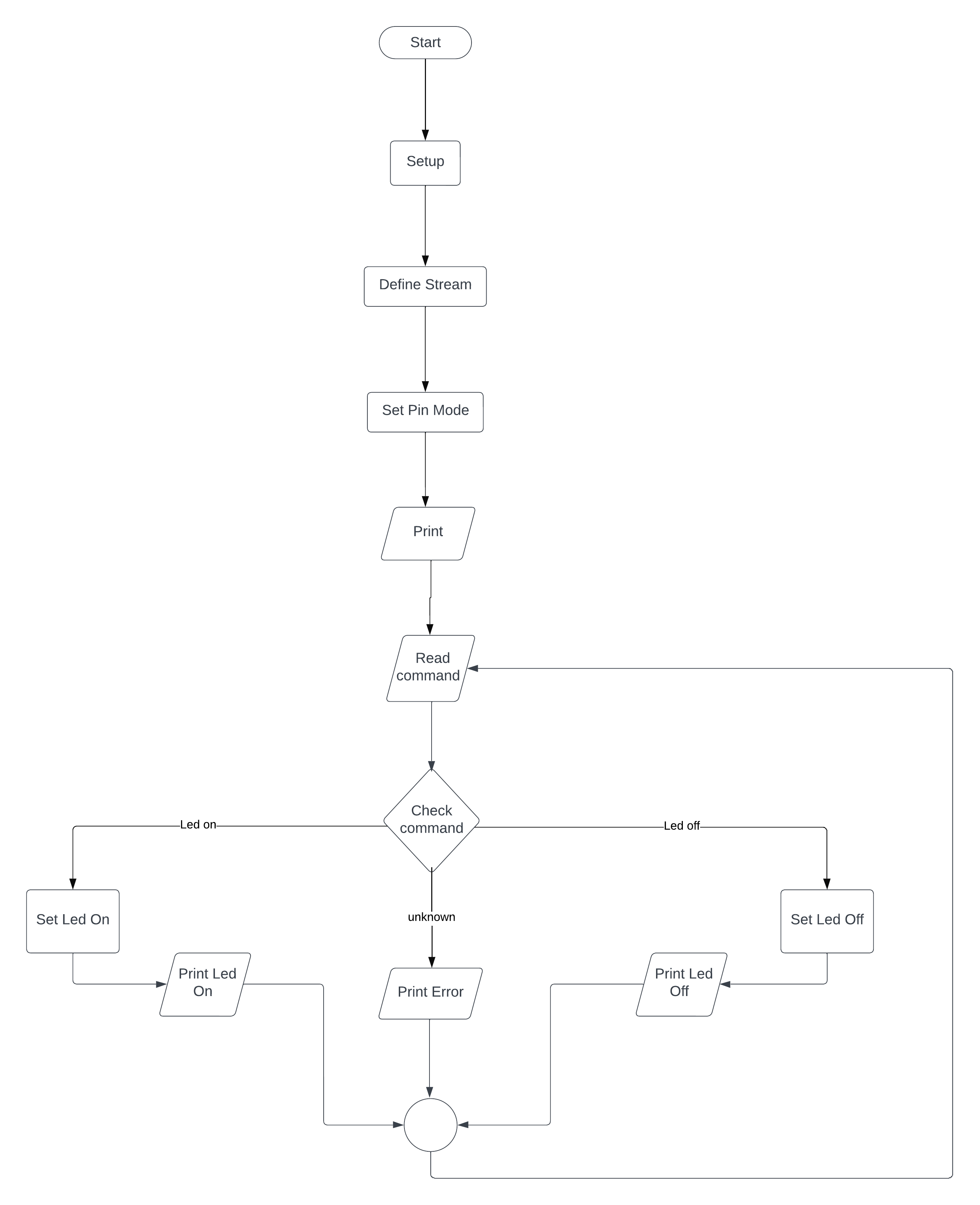
My Arduino code utilizes a serial communication interface to control the state of an LED based on user input.

**Table 1** Arduino Uno used functions.

|  |  |
| --- | --- |
| **Function** | **Description** |
| int my\_putChar(char ch, FILE \* f) | This function is a custom implementation of the character output function. It is designed to be used with the fdevopen function to redirect the standard output (stdout) to the Serial communication. In my case, it sends a character (ch) to the Serial port using Serial.write. |
| char my\_getChar(FILE \* f) | This function is a custom implementation of the character input function. It is used with fdevopen to redirect the standard input (stdin) to the Serial communication. It waits until a character is available in the Serial buffer and then reads and returns it using Serial.read. |
| void setup() | Initializes the Serial communication with the specified baud rate.  Uses fdevopen to set up custom character input and output functions.  Sets the pinMode for the LED pin as OUTPUT.  Prints an initial message prompting the user to enter 'LED\_ON' or 'LED\_OFF'. |
| void loop() | Declares a character array command to store the user input.  Uses scanf to read user input into the command array.  Compares the input with predefined commands 'LED\_ON' and 'LED\_OFF'.  If a match is found, it changes the state of the LED accordingly and prints a corresponding message.  If no match is found, it prints an error message. |

**2.2 Block Diagram**

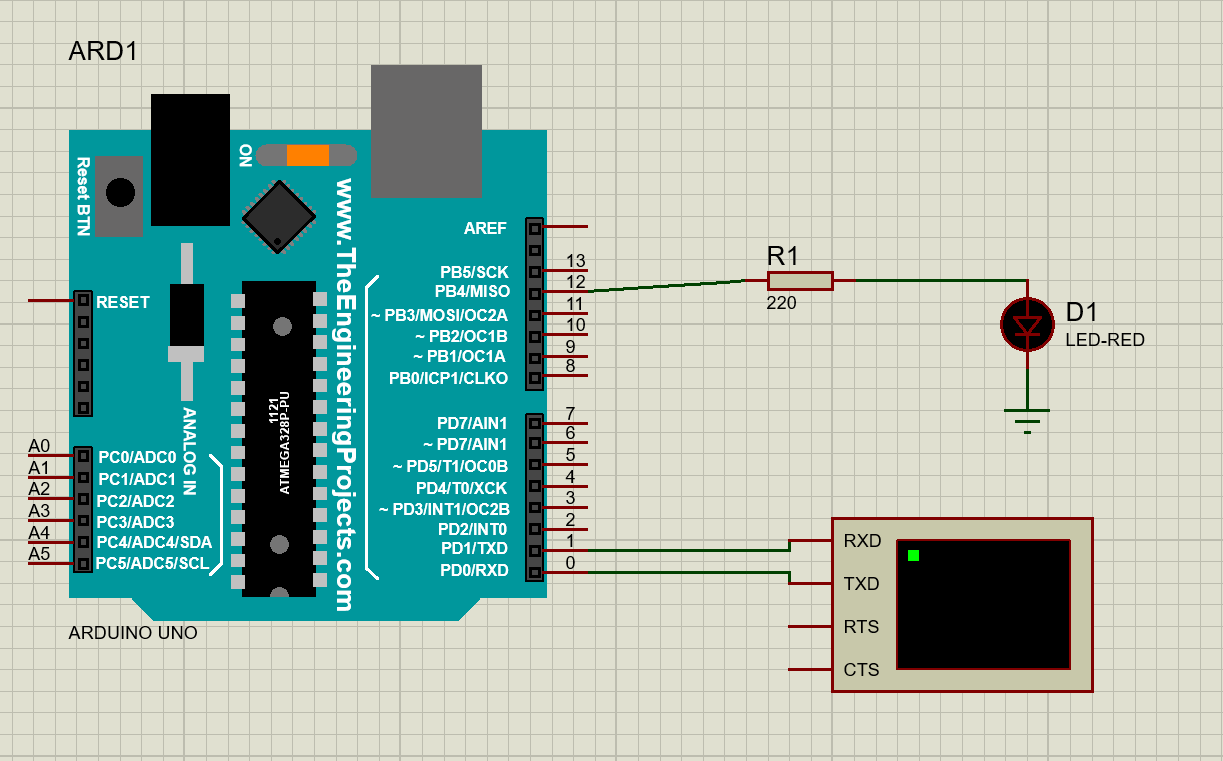
This block diagram provides a visual representation of the main components and their interactions for my Arduino program.



**Figure 1** STDIO and LED program [1].

**2.3 Simulated schematic**

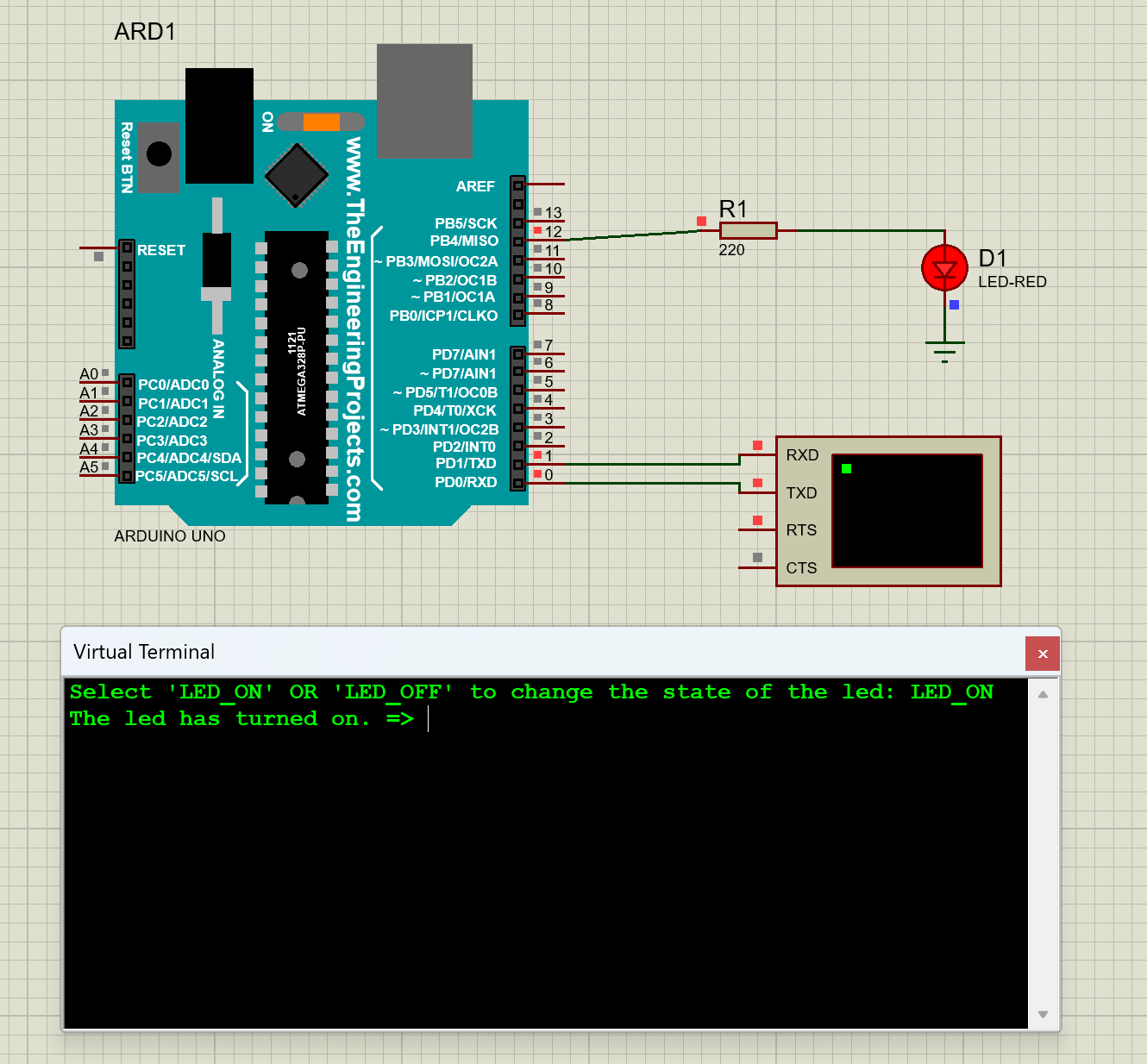
This is the simulated Arduino Uno scheme assembled in Proteus, here are represented 4 elements: Arduino Uno with its microcontroller, a red led, resistor and a virtual terminal.



**Figure 2** STDIO and LED assembly scheme [2].

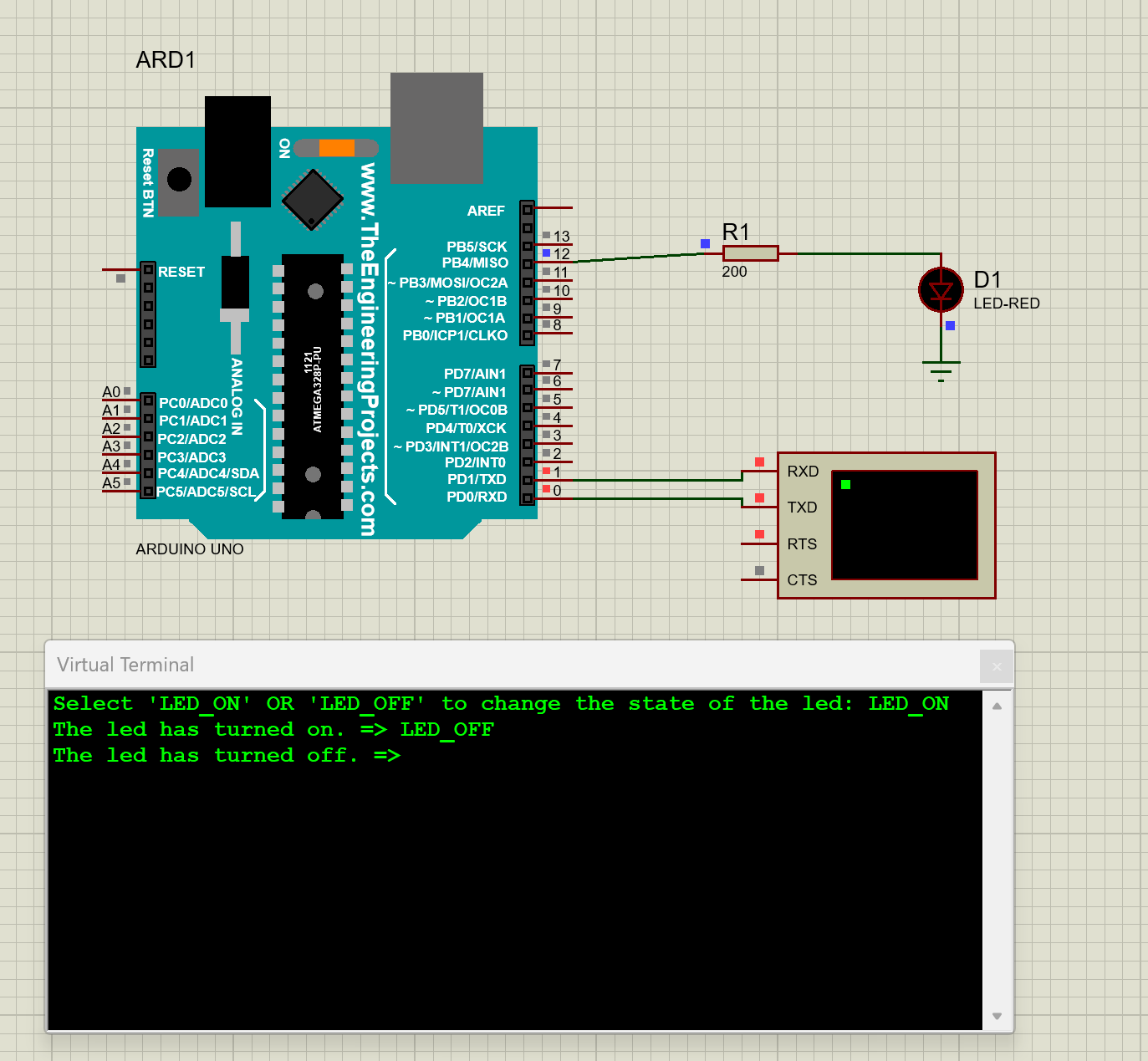
**2.4 Screenshots of running program.**

The first image shows the virtual terminal available, where the user typed ‘LED\_ON’ and the LED turned on.



**Figure 3** Simulation of the scheme when LED is on.

The second image shows the virtual terminal available, where the user typed ‘LED\_OFF’ and the LED turned off.



**Figure 4** Simulation of the scheme when LED is off.

**CONCLUSIONS**

This laboratory work was a great experience for me. I learned a lot of small concepts, rules that helped me realize this project.

First of all, I managed to create a code that would establish communication through the Serial interface, allowing the user to input commands such as ‘LED\_ON’ or ‘LED\_OFF’ to turn the LED on or off, respectively.

Second of all, I was able to create a simple scheme that has helped me simulate my project in real-time. I managed to correctly connect the elements used in my projects, such as Arduino Uno, resistor, led, and the virtual terminal.

Moreover, I built my own custom functions for redirecting the standard input and output for the Serial communication, which enabled me the use of functions like printf and scanf in the Arduino environment.

To sum up, this laboratory work has deepened my understanding of electronic circuits and Arduino development. The project’s success will contribute to my future projects in the field of embedded systems.

**BIBLIOGRAPHY**

1. AVR-LIBC DOCUMENTATION: *<stdio.h> library.* Input Output facilities [accessed 08.02.2024]. Access link: [avr-libc: <stdio.h>: Standard IO facilities (nongnu.org)](https://www.nongnu.org/avr-libc/user-manual/group__avr__stdio.html)
2. PROTEUS DOCUMENTATION: *Configuring proteus.* Creating circuits. [accessed 09.02.2024]. Access link: [How to Simulate Arduino Projects Using Proteus | Arduino | Maker Pro](https://maker.pro/arduino/projects/how-to-simulate-arduino-projects-using-proteus)

**APPENDIX A: Source code**

#include <Arduino.h>

#include <stdio.h>

#define LED\_PIN 12

#define BAUD\_RATE 9600

#define TEXT\_ON "LED\_ON"

#define TEXT\_OFF "LED\_OFF"

int my\_putChar(char ch, FILE \* f) {

return Serial.write(ch);

}

char my\_getChar(FILE \* f) {

while(!Serial.available()) {

}

return Serial.read();

}

void setup() {

Serial.begin(BAUD\_RATE);

FILE \*my\_stream = fdevopen(my\_putChar, my\_getChar);

stdin = stdout = my\_stream;

pinMode(LED\_PIN, OUTPUT);

printf("Select 'LED\_ON' OR 'LED\_OFF' to change the state of the led: ");

}

void loop() {

char command[64] = "";

scanf("%s", command);

if (strcmp(command, TEXT\_ON) == 0) {

digitalWrite(LED\_PIN, HIGH);

printf("The led has turned on. => ");

}

else if (strcmp(command, TEXT\_OFF) == 0) {

digitalWrite(LED\_PIN, LOW);

printf("The led has turned off. => ");

}

else {

printf("No such command available. ");

}

}

**Listing 1** STDIO and LED program.