Ministry of Education of Republic of Moldova

Technical University of Moldova

Faculty of Computers, Informatics and Microelectronics

Software Engineering Department

Embedded Systems

Laboratory work work #1.1

User interaction via Serial Communication. Led

During the preparation of this report, the author used various AI Tools such as ChatGPT, Gemini, Claude to generate/augment the content, generate the code and structure the directory. The resulting information was reviewed, validated, and adjusted to meet the requirements of the laboratory assignment.

Author: Timur Cravtov
std. gr. FAF-231

Verified by:
Alexei Martîniuc
asist. univ

Chișinău, 2026

Contents

1 Objective of the work

2 System design

2.1 System architecture

The system is designed to control an LED connected to an Arduino board via serial communication. The schema of the system architecture is presented in Figure 1. The main components of the system include:

- **Arduino Board:** The microcontroller that interfaces with the LED and handles serial communication.
- **LED:** The output device that will be turned on or off based on commands received via serial communication.
- **Serial Communication Interface:** Facilitates the exchange of commands between the user and the Arduino board.
- **User Input:** The user sends commands ("led on" or "led off") through a serial terminal to control the LED state.

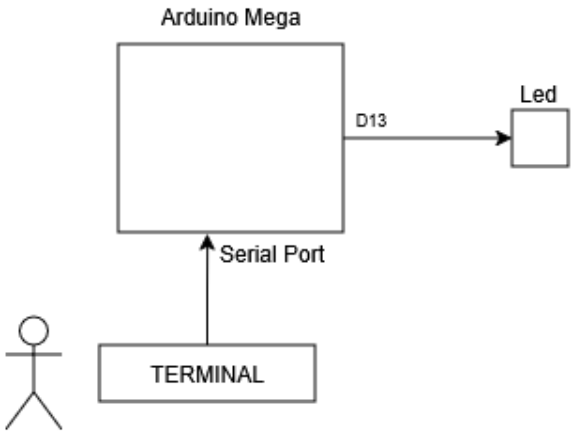


Figure 1: System Architecture Diagram

2.2 Flowchart of the program

The flowchart in Figure 2 illustrates the logic of the program running on the Arduino board. The program continuously listens for user input via serial communication and processes commands to control the LED state.

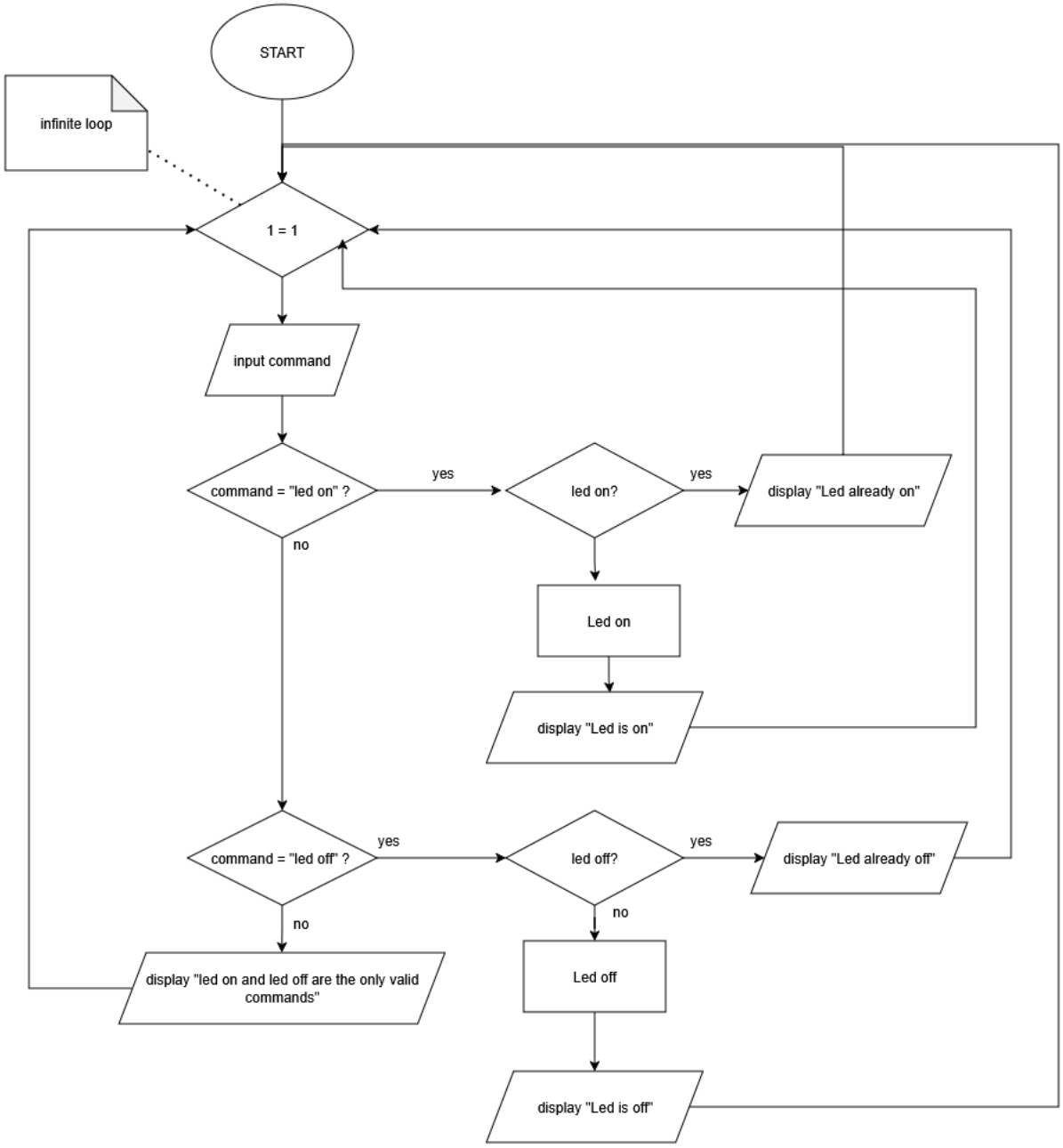


Figure 2: Program Flowchart

In this flowchart, there is an infinite loop that waits for user input. When a command is received, it checks if the command is "led on" or "led off" and turns the LED on or off accordingly. If the command is unrecognized, it prompts the user to enter a valid command.

2.3 Electronic circuit

Figure 3 illustrates the electronic circuit designed for controlling an LED using an Arduino board. The circuit consists of an LED connected to a digital output pin of the Arduino through a current-limiting resistor. The ground pin of the Arduino is connected

to the cathode of the LED, while the anode is connected to the resistor, which in turn is connected to the designated digital pin.

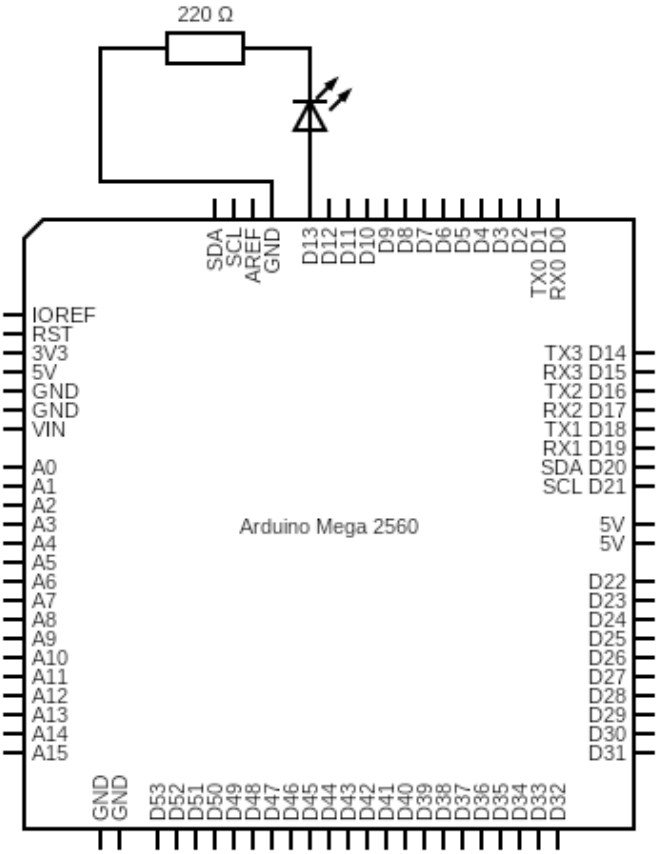


Figure 3: Circuit diagram of the LED control system

Since the Arduino board operates at 5V, a resistor value of 220Ω is chosen to limit the current through the LED to a safe level, preventing damage to both the LED and the Arduino pin.

Following the Om's law, the current flowing through the LED can be calculated as:

$$I = \frac{V_{supply} - V_{LED}}{R} = \frac{5V - 2V}{220\Omega} \approx 13.64mA$$

which is within the safe operating limits for both the LED and the Arduino pin.

3 Conclusion

References

[1] GitHub repository `https://github.com/TimurCravtov/EmbeddedSystemsLabs`

[2] Arduino LED Control https://roboticsbackend.com/arduino-led-complete-tutorial/

[3] Circuit Diagram Builder https://www.circuit-diagram.org/docs

A Source Code

Besides this appendix, the source code is available in the GitHub repository [1] with all build instructions.

main.cpp

```cpp
#include <Arduino.h>
#include <stdio.h>
#include <stdlib.h>
#include <led/led.h>
#include <serialio/serialio.h>


// setup LED on pin 13
const uint8_t ledPinNum = 13;
Led led(ledPinNum);

void setup() {

  // start serial communication
  Serial.begin(9600);
  delay(100);


  // the function does what you think it does
  redirectSerialToStdio();
}

bool equals(const char* a, const char* b) {
  return strcmp(a, b) == 0;
}

void loop() {

  // reading command from serial
```

```
30    char buffer[10] = {0};
31    fflush(stdout);
32
33    printf("Waiting for command (led on / led off): \n");
34
35    readLine(buffer, sizeof(buffer));
36
37    if (equals(buffer, "led on")) {
38
39      // printf("Intered command to turn LED ON\n");
40      if (led.isOn()) {
41        printf("LED is already ON\n");
42      } else {
43        printf("Turning LED ON\n");
44        led.on();
45      }
46    }
47
48    else if (equals(buffer, "led off")) {
49
50      // printf("Intered command to turn LED OFF\n");
51      if (!led.isOn()) {
52        printf("LED is already OFF\n");
53      } else {
54        printf("Turning LED OFF\n");
55        led.off();
56      }
57    } else {
58      printf("Unknown command. Use 'led on' or 'led off'.\n");
59    }
60  }
```

serialio.h

```
1  #pragma once
2
3  #include <Arduino.h>
4
5  /// @brief Functions for serial input/output redirection and
       line reading
```

```
6 void readLine(char* buffer, size_t maxLen);
7 void redirectSerialToStdio();
```

serialio.cpp

```
1  #include <Arduino.h>
2  #include <serialio/serialio.h>
3  #include <stdio.h>
4
5  #include <stdlib.h>
6
7  // add a char to serial output
8  int serial_putchar(char c, FILE* stream) {
9    Serial.write(c);
10   return 0;
11 }
12
13 // get a char from serial input
14 int serial_getchar(FILE* stream) {
15   while (!Serial.available() == 0);
16   return Serial.read();
17 }
18
19 // reads a line from stdin into buffer, up to maxLen-1
     characters
20 void readLine(char* buffer, size_t maxLen) {
21   size_t i = 0;
22   while (i < maxLen - 1) {
23     int c = fgetc(stdin);
24     if (c == '\n' || c == '\r' || c == EOF) {
25       break;
26     }
27     buffer[i++] = (char)c;
28   }
29   buffer[i] = '\0';
30 }
31
32 // Helper functions for redirecting Serial to stdio
33 void redirectSerialToStdio() {
34   static FILE uartout;
```

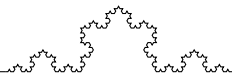
```
35    static FILE uartin;
36
37    // setup the UART streams for input and output
38    fdev_setup_stream(&uartout, serial_putchar, NULL,
          _FDEV_SETUP_WRITE);
39    fdev_setup_stream(&uartin, NULL, serial_getchar,
          _FDEV_SETUP_READ);
40
41    // redirect standard input and output to the UART
42    stdout = &uartout;
43    stdin = &uartin;
44    stderr = &uartout; // never used this, but why not. output
          errors to serial as well
45  }
```

led.h

```
1  #pragma once
2
3  #include <Arduino.h>
4
5  /// @brief A simple LED control class
6  class Led {
7  public:
8    explicit Led(uint8_t pin);
9
10   void on();
11   boolean isOn();
12   void off();
13   void toggle();
14
15 private:
16   uint8_t pin_;
17 };
```

led.cpp

```
1  #include <led/led.h>
2
3  Led::Led(uint8_t pin) : pin_(pin) {
4    pinMode(pin_, OUTPUT);
```



```
5  }
6
7  void Led::on() {
8    digitalWrite(pin_, HIGH);
9  }
10
11 boolean Led::isOn() {
12   return digitalRead(pin_) == HIGH;
13 }
14
15 void Led::off() {
16   digitalWrite(pin_, LOW);
17 }
18
19 void Led::toggle() {
20   digitalWrite(pin_, !digitalRead(pin_));
21 }
```