



MINISTRY OF EDUCATION, CULTURE AND RESEARCH OF THE
REPUBLIC OF MOLDOVA

Technical University of Moldova Faculty of Computers, Informatics and
Microelectronics Department of Software and Automation Engineering

Postoronca Dumitru FAF-233

Report

Laboratory work n.1

on Embedded Systems

Checked by:

A. Martiniuc, *university assistant*
DISA, FCIM, UTM

Chişinău – 2026

1 Domain Analysis

1.1 Application Context and Utilized Technologies

The primary objective of this laboratory work is to establish a communication bridge between a workstation (PC) and an embedded system (MCU) using the **UART (Universal Asynchronous Receiver-Transmitter)** protocol. This interaction is fundamental in embedded engineering for debugging, real-time sensor monitoring, and remote command execution.

The application leverages the standard input/output library (**STDIO**) adapted for microcontroller environments. This approach allows the use of classic C functions, such as `printf()` and `scanf()`, to abstract the hardware registers of the serial port. This provides a human-readable interface for transmitting and receiving text-based commands, moving away from raw byte manipulation to a more sophisticated command-line interaction.

1.2 Hardware and Software Components

Table 1: Hardware and Software Stack

Component		Role and Justification
Microcontroller		The central processing unit (e.g., ATmega328P) that parses the incoming character stream and toggles GPIO pin states.
Serial-to-USB Interface		Facilitates the translation of the MCU's logic levels into USB protocol for communication with the PC terminal.
LED & 220 Ω Resistor		The output peripheral used for visual validation. The resistor ensures current limiting to protect the component.
Neovim	+	The chosen development environment (IDE). Neovim provides high-efficiency editing, while the plugin integrates <code>arduino-cli</code> for compilation.
Arduino-nvim		
Serial Monitor		The terminal emulator used to send the "led on" and "led off" strings and display confirmation responses.

1.3 System Architecture and Solution Justification

The system adopts a **Command Parser architecture**. This solution was chosen to ensure extensibility; by utilizing the **STDIO** library, the code becomes more portable and readable compared to direct manipulation of UART circular buffers.

The data flow is structured as follows:

1. **Input:** The user enters a string command via the Neovim-integrated terminal or a standalone console.
2. **Transmission:** Data is sent asynchronously to the MCU via the TX/RX lines.
3. **Processing:** Functions like `scanf()` or `fgets()` capture the string, which is then passed to a logic module that compares the input against predefined instructions.
4. **Action:** The MCU interacts with the LED driver module to change the physical state of the hardware.

1.4 Relevant Case Study

A real-world application of this concept is the **Command Line Interface (CLI)** found in network infrastructure equipment. Technicians connect via a console port (serial) to input text commands for hardware configuration. Similarly, in industrial automation, this interface is used for field calibration of sensors where a complex graphical user interface (GUI) is not feasible or required.