MINISTRY OF EDUCATION, CULTURE AND RESEARCH
OF THE REPUBLIC OF MOLDOVA
Technical University of Moldova
Faculty of Computers, Informatics and Microelectronics
Department of Software Engineering and Automation

Iamandii Ion, FAF-233

Report

Laboratory work n.0

of Embedded Systems

Checked by:
Martîniuc Alexei, *university lector*
DISA, FCIM, UTM

Chișinău – 2026

1. Domain Analysis

1.1 Technologies Used and Application Context

This laboratory introduces the development environment for IoT applications using a Microcontroller Unit and the Arduino framework. IoT systems commonly rely on embedded hardware combined with lightweight software to interact with physical components such as sensors, buttons, LEDs, and displays.

The application developed in this laboratory demonstrates a basic interaction between input and output peripherals — a button (input) and an LED (output). This represents a fundamental building block for more complex IoT systems such as smart home devices, industrial automation modules, or wearable electronics.

1.2 Hardware Components

- MCU (Microcontroller Unit) – central processing unit of the embedded system responsible for executing program logic.
- LED (Light Emitting Diode) – output device used to visually indicate system state.
- Push Button – input device used to trigger state changes.
- Resistors and wiring – supporting electronic components for proper circuit operation.
- Simulation Environment (Wokwi) – used instead of a physical board for validation.

1.3 Software Components

- Arduino Framework
- C++ Programming Language
- VS Code IDE with Arduino support
- Simulator (Wokwi) for testing firmware behavior without physical hardwar

1.4 System Architecture

The system follows a modular architecture, separating each hardware peripheral into its own software module. This approach improves maintainability, readability, and reusability in future projects.

The architecture consists of:

- Main control logic (`main.cpp`)
- LED module (`Led.h`, `Led.cpp`)
- Button module (`Button.h`, `Button.cpp`)

1.5 Case Study

A simple button-LED system is analogous to real-world IoT scenarios such as:

- Smart light switches
- Alarm activation buttons
- Machine start/stop controls

2. Design

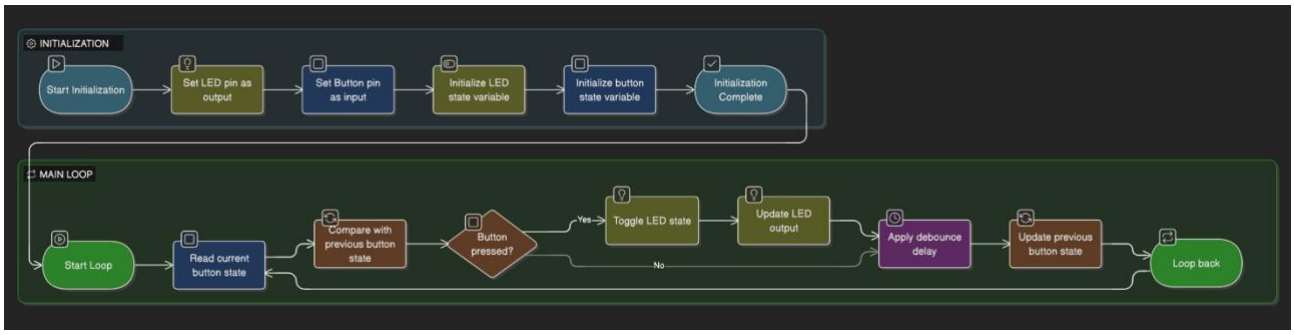
2.1 Architectural description

The system contains:

- One MCU
- One input peripheral (Button)
- One output peripheral (LED)

The MCU continuously reads the button state and toggles the LED state when a press is detected.

2.2 Flowchart



2.3 Project Structure

project/
├── main.cpp
├── Led.h
├── Led.cpp
├── Button.h
├── Button.cpp

2.4 Modular implementation

Header Files (.h):

- Button.h
- Led.h

Source Files (.cpp):

- Button.cpp
- Led.cpp
- Main.cpp

3. Results

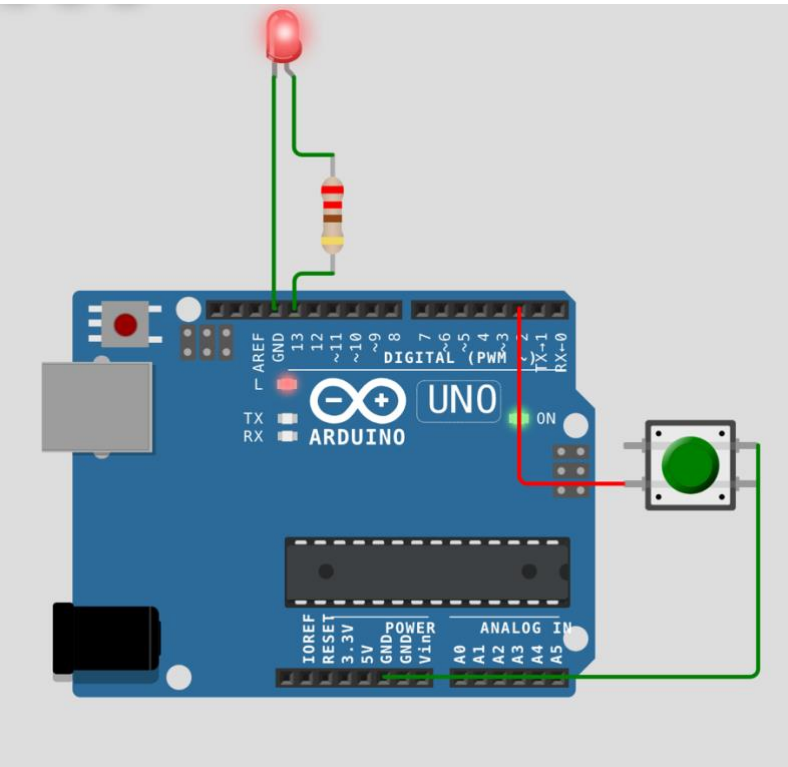


Fig 1 – button was pressed, light on

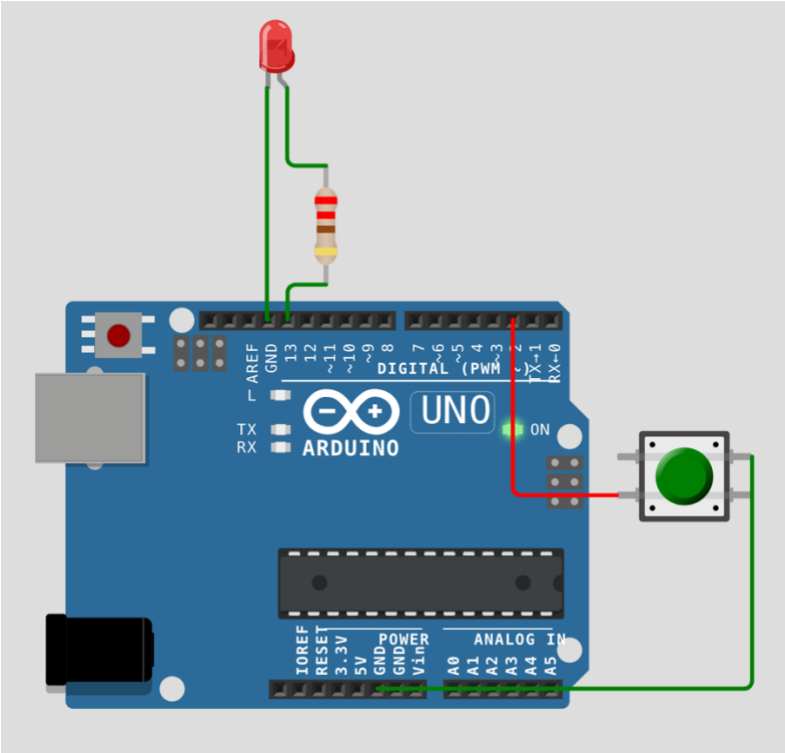


Fig 2 – button was pressed twice, light off

4. Conclusions

The system successfully detects button presses and toggles LED state reliably. The modular design improves scalability and maintainability.

Limitations:

- No advanced debounce filtering.
- No multi-input handling.
- Limited functionality.

Possible improvements:

- Add LCD or keypad.
- Implement interrupts instead of polling.
- Add wireless IoT communication (Wi-Fi / Bluetooth).
- Improve debounce logic.

This laboratory demonstrates core embedded principles used in smart devices, industrial controllers, and consumer electronics.

5. AI usage

During the preparation of this report, the author used ChatGPT to assist in structuring and consolidating content. The resulting information was reviewed, validated, and adjusted according to laboratory requirements.

6. Bibliography

- Arduino Official Documentation: https://www.arduino.cc/
- Flowchart generator: https://www.eraser.io/ai/flowchart-generator
- Arduino simulator: https://wokwi.com/

7. Source code

Github link: https://github.com/ion190/embbeded-systems-labs/tree/main/lab0/src

Main.cpp file:

```
#include "Led.h"
#include "Button.h"

// pins
Led led(13);
Button button(2);

bool lastState = false;

void setup() {
  led.begin();
  button.begin();
}

void loop() {
  bool currentState = button.isPressed();

  if (currentState && !lastState) {
    led.toggle();
```

```
    delay(200); // debounce
  }

  lastState = currentState;
}
```

Button.h file:

```
#ifndef BUTTON_H
#define BUTTON_H

class Button {
  private:
    int pin;

  public:
    Button(int buttonPin);
    void begin();
    bool isPressed();
};

#endif
```

Button.cpp file:

```
#include <Arduino.h>
#include "Button.h"

Button::Button(int buttonPin) {
  pin = buttonPin;
}

void Button::begin() {
  pinMode(pin, INPUT_PULLUP);
}

bool Button::isPressed() {
  return digitalRead(pin) == LOW;
}
```

Led.h file:

```
#ifndef LED_H
#define LED_H

class Led {
  private:
    int pin;

  public:
    Led(int ledPin);
    void begin();
    void on();
    void off();
    void toggle();
};

#endif
```

Led.cpp file:

```
#include <Arduino.h>
#include "Led.h"

Led::Led(int ledPin) {
  pin = ledPin;
}

void Led::begin() {
  pinMode(pin, OUTPUT);
}

void Led::on() {
  digitalWrite(pin, HIGH);
}
```

```
void Led::off() {
  digitalWrite(pin, LOW);
}

void Led::toggle() {
  digitalWrite(pin, !digitalRead(pin));
}
```