

**UNIVERSITATEA TEHNICĂ A MOLDOVEI**

**FACULTATEA: CALCULATOARE, INFORMATICĂ ŞI MICROELECTRONICĂ**

**DEPARTAMENTUL: INGINERIA SOFTWARE ȘI AUTOMATICA**

# **Laboratory work NR. 0**

# **Introduction to IoT development**

**Executed by: Alexandra Konjevic, gr. FAF-213**

**Verified: Moraru Dumitru**

**Chișinău – 2024**

**TASK**: Create an application that changes the state of an LED based on a button press.

**OBJECTIVES:**

- It is recommended to use an IDE with Arduino support that allows working with multiple files, for example, Eclipse.

- For validation, it is recommended to use a simulator, such as Proteus.

- The functionalities for each peripheral device (LED, button, LCD, keypad) should be implemented in separate files for the purpose of reusability in future projects.

- Use camelCase coding conventions.

**IMPLEMENTATION:**

Arduino, an open-source platform renowned for its accessibility in electronics, serves as a gateway for both beginners and experienced enthusiasts to bring their electronic ideas to life. At the core of many Arduino projects lie fundamental components like buttons and LEDs, seamlessly integrated into the design for user interaction and visual feedback.

Buttons, or push-button switches, play a pivotal role in enabling user input within electronic circuits. These versatile components possess two primary states - pressed and unpressed. To maintain a stable state when not pressed, buttons are often coupled with pull-up or pull-down resistors. Arduino leverages its digital input pins to detect changes in the button's state, facilitating the creation of responsive and interactive projects.

On the visual front, Light Emitting Diodes (LEDs) take center stage. As semiconductor devices emitting light when a current passes through, LEDs serve as visual indicators, conveying information or signaling status changes. Controlled through Arduino's digital output pins, LEDs provide a dynamic element to electronic projects, allowing for the creation of captivating visual effects and clear feedback mechanisms.

Starting with the necessary circuit, the following is the diagram I made using wokwi:

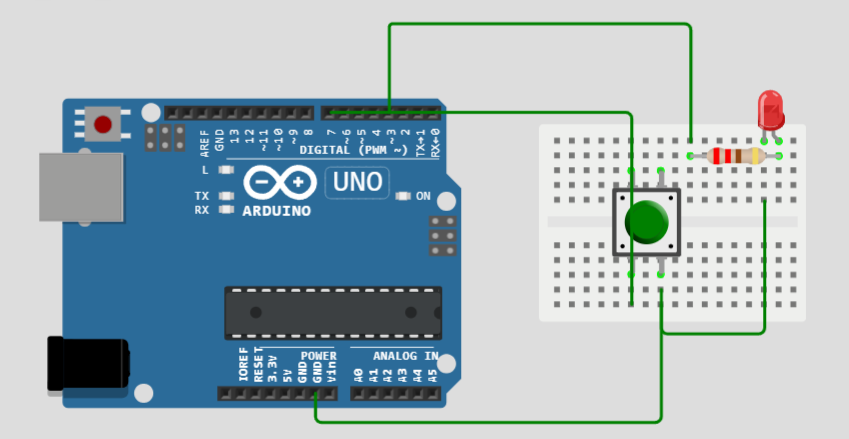


Figure 1. Simulated electrical schema

There we can see all the used component: Arduino UNO R3, a breadboard, a button, a led and a 220 Ω resistor. Next, let’s analyze the code step by step:

1. Definition of the constants and variables:

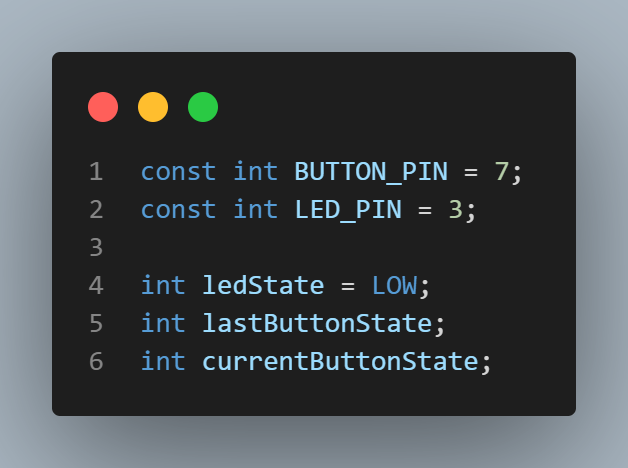


Figure 2. Definition of constants in the program

Constants are defined for the pin numbers connected to the button and LED. `BUTTON\_PIN` is set to pin 7, and `LED\_PIN` is set to pin 3. `ledState` - keeps track of the current state (HIGH or LOW) of the LED on `LED\_PIN`; `lastButtonState` - holds the previous state of the button; `currentButtonState` - holds the current state of the button.

2. Setup:

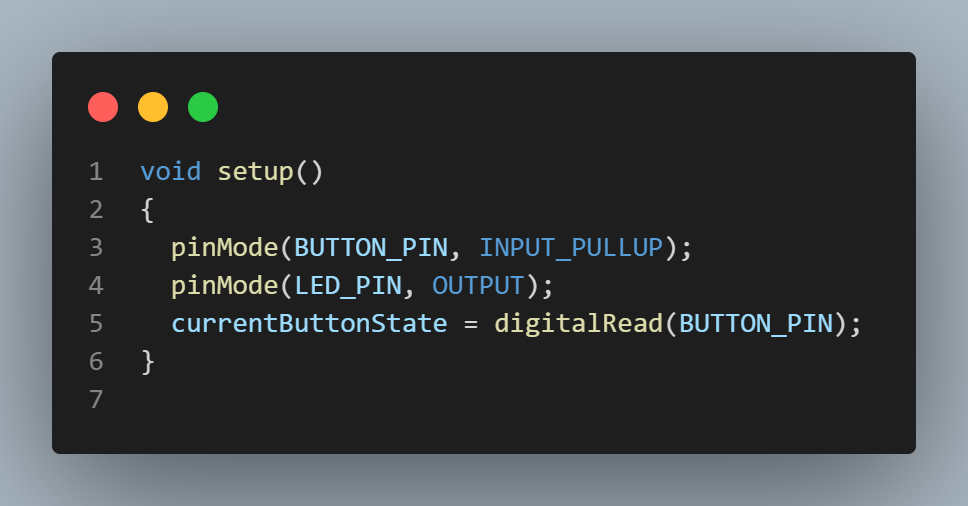


Figure 3. Setup function of the program

First of all, we have the function `pinMode(BUTTON\_PIN, INPUT\_PULLUP)`, which configures the button pin as an input with a pull-up resistor. The pull-up resistor means the pin is normally HIGH and goes LOW when the button is pressed. After that, using the `pinMode(LED\_PIN, OUTPUT)`, I configured the LED pin as an output. The next line, `currentButtonState = digitalRead(BUTTON\_PIN)`, reads and initializes the `currentButtonState` with the initial state of the button.

3. Loop:

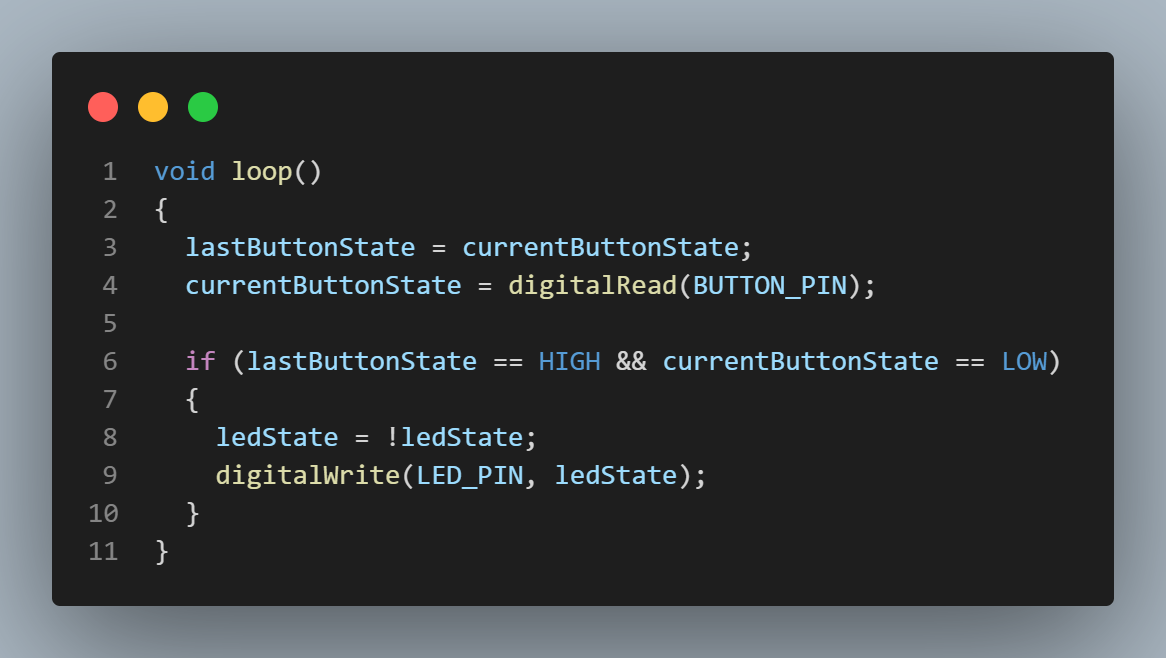


Figure 4. Loop function of the program

Initially, the program updates `lastButtonState` with the previous state of the button and after that, it updates the `currentButtonState` with the current state of the button. The if statement checks if the button has changed from HIGH to LOW (indicating a button press). If the condition is true, it toggles the `ledState` (from HIGH to LOW or vice versa) and updates the LED on `LED\_PIN` accordingly using `digitalWrite`.

**SIMULATION:**

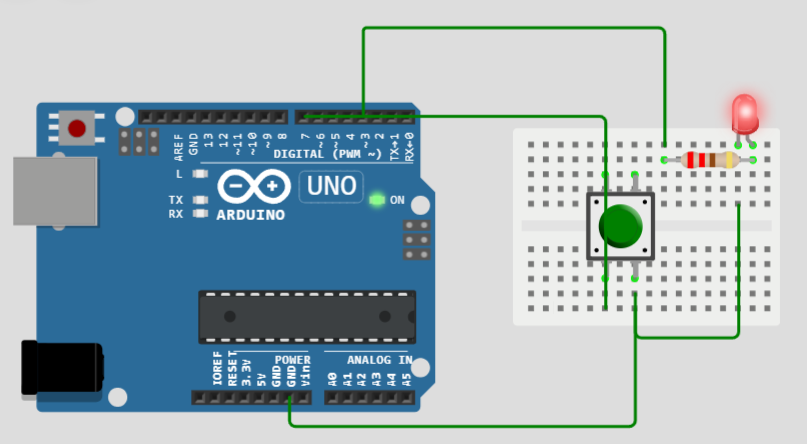


Figure 5. Simulated circuit

**CONCLUSION:**

In this laboratory experiment, we successfully designed, implemented, and simulated a circuit using Arduino to create a simple interactive system. The circuit consisted of a button and an LED, where the LED state toggled each time the button was pressed. The implementation was achieved through careful consideration of digital input and output configurations, utilizing the Arduino programming environment.

The key components of the circuit were the button and LED, connected to specific pins on the Arduino board. The use of a pull-up resistor for the button ensured stable readings when the button was not pressed, preventing floating states and enhancing the reliability of the system.

The programming logic employed in the Arduino code was crucial for monitoring the button state changes and controlling the LED accordingly. The code continuously checked for transitions from a high to a low state on the button pin, indicating a button press. Upon detection of a press, the LED state was toggled, creating a visually perceptible effect.

Simulating the circuit provided valuable insights into the expected behavior of the system. It allowed us to observe the interaction between the button and the LED, confirming that the implemented code and circuit design functioned as intended. The simulation phase facilitated debugging and refinement, ensuring the reliability and stability of the circuit.

This laboratory work not only enhanced our understanding of digital input and output operations with Arduino but also provided practical experience in circuit design and simulation. The ability to create interactive systems using microcontrollers is a fundamental skill in electronics, and this experiment served as a hands-on introduction to the principles of digital control.

In conclusion, the successful creation and simulation of the circuit demonstrated the effective integration of hardware and software components, reinforcing key concepts in embedded systems and Arduino programming. The knowledge gained from this experiment lays a foundation for more complex projects and applications involving microcontroller-based systems.

**BIBLIOGRAPHY**

1. Arduino: Arduino UNO R3. Arduino official site, ©2024 [quote 2024]. Access link: <https://docs.arduino.cc/hardware/uno-rev3/>
2. Buttons in Arduino: How to Wire and Program a Button. Arduino official site, ©2024 [quote 2024]. Access link: https://docs.arduino.cc/built-in-examples/digital/Button/