**TECHNICAL UNIVERSITY OF MOLDOVA**
**FACULTY OF COMPUTERS, INFORMATICS AND MICROELECTRONICS**
**DEPARTMENT OF SOFTWARE ENGINEERING AND AUTOMATICS**

# Laboratory work nr. 0:

## Introduction to IoT Application Development Environment - Button-led

**Brinza Cristian** st. gr. FAF-212
**Moraru Dumitru** lect. univ

**Chisinau 2024**

**The task of the laboratory work:**

To design an application based on an MCU that would change the state of an LED upon detecting a button press.

- Use CamelCase coding conventions;
- Declare port variable like intended;
- The functionalities for each peripheral device (LED, button, LCD, keypad) should be implemented in separate files for the purpose of reuse in future projects.

**Main functions/methods used to execute the task:**

- In the **LED** Class:
  - **Constructor (LED(int pin)):** Initializes an **LED** object on a specified pin. It sets the pin mode to OUTPUT and initializes the internal state to represent the LED being off.
  - **on():** Sets the LED's state to **HIGH**, turning the LED on.
  - **off():** Sets the LED's state to **LOW**, turning the LED off.
  - **toggle():** Changes the LED's state from ON to OFF or from OFF to ON based on its current state. This function is crucial for toggling the LED each time the button is pressed and released.
- In the **Button** Class:
  - **Constructor (Button(int pin, unsigned long debounceDelay)):** Initializes a **Button** object on a specified pin with a debounce delay. It sets the pin mode to **INPUT** and initializes variables for managing the debounce logic.
  - **read():** Reads the button's state with debounce logic. It ensures that the button's state change is stable for a defined duration (**debounceDelay**) before considering it a valid press or release. This method returns the current debounced state of the button (either **HIGH** or **LOW**).
- In the **main.ino** Sketch:
  - **setup():** A function called once when the sketch starts. It's typically used to initialize variables, pin modes, or other libraries. In this refactored code, it might appear empty or contain initialization code for other components not shown in the example.
  - **loop():** The main program loop which runs continuously after **setup().** It reads the button state using **button.read()** and toggles the **LED** state with **led.toggle()** when a button press is detected. This function embodies the core logic of responding to user input with visual feedback.

**Block Diagram:**

The diagram is consisting of the following main blocks:

- **Main Program (main.ino):** The central point that initializes and controls other components (LED and Button) and manages the program flow[1].
- **LED Class:** Represents the LED control logic, including turning the LED on, off, and toggling its state.
- **Button Class:** Handles the debounce logic and state reading of a button press.
- **Arduino Board:** The physical layer where the LED and Button are connected.

Here is the diagram itself:

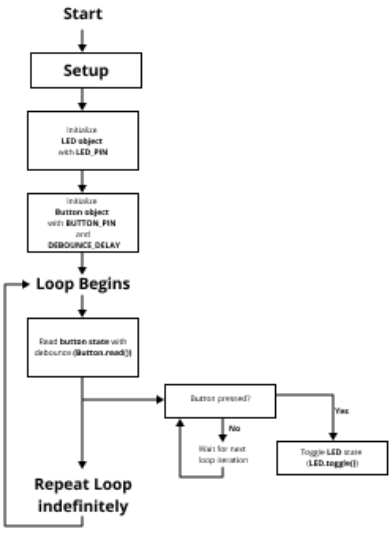


**Figure 1** Program schema [1].

**Simulated or real assembled electrical schematic diagram[2] :**

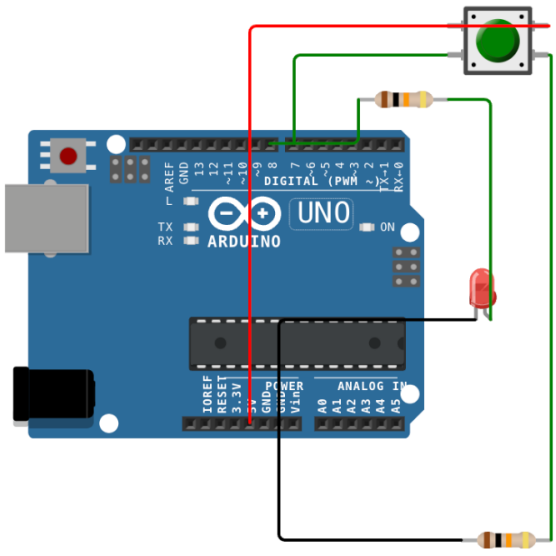


**Figure 2** Phisical board schema [2].

**Code:**

main.ino:

```
#include "LEDControl.h" // Include the LED control header file
#include "ButtonControl.h" // Include the button control header file

void setup() {
  setupLED(); // Initialize LED pin mode
  setupButton(); // Initialize button pin mode
}

void loop() {
  if (checkButton()) { // Check if the button was pressed
    toggleLED(); // Toggle the LED state
  }
}
```

LEDControl.h:

```


#ifndef LEDControl_h
#define LEDControl_h

#include <Arduino.h>

#define LED_PIN 8 // Define the LED pin number

byte ledState = LOW; // Variable to store the current state of the LED

// Function to initialize the LED pin
void setupLED() {
  pinMode(LED_PIN, OUTPUT); // Set the LED pin as an output
}

// Function to toggle the LED state
void toggleLED() {
  ledState = !ledState; // Toggle the state
  digitalWrite(LED_PIN, ledState); // Apply the new state to the LED
}

#endif
```

ButtonControl.h:

```
```

```
#ifndef ButtonControl_h
#define ButtonControl_h

#include <Arduino.h>

#define BUTTON_PIN 7 // Define the button pin number

byte lastButtonState = LOW; // Variable to store the last button state, initialized
to HIGH assuming pull-up resistor

// Function to initialize the button pin
void setupButton() {
  pinMode(BUTTON_PIN, INPUT); // Set the button pin as an input
}

// Function to check the button state without debouncing
bool checkButton() {
  byte currentButtonState = digitalRead(BUTTON_PIN); // Read the current state of
the button
  if (currentButtonState != lastButtonState) {
    lastButtonState = currentButtonState; // Update the last button state
    // Check if the button is pressed (assuming active low configuration)
    if (currentButtonState == LOW) {
      return true; // Button press detected
    }
  }
  return false; // No button press detected
}

#endif
```

Code link: https://wokwi.com/projects/388599535614645249

**Screenshots of the simulation execution:**

Here we have a screenshot of the WOKWI simulation[2] :

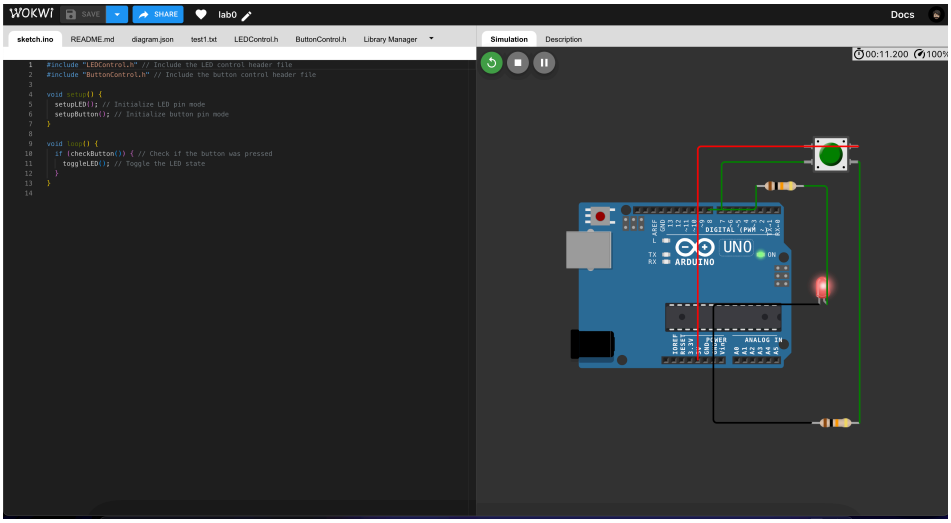


**Figure 3** Screenshot of the simulation [3].

## CONCLUSION

The lab work was all about making an LED light turn on and off when we press a button, which was really cool to see in action. We learned to write our code in a neat way that makes it easy to read and fix if something goes wrong. Also, figuring out how to make sure the button press didn't mess up because of tiny electrical glitches was a big win. We used an Arduino board, which was super helpful because there's a lot of help out there if we got stuck. This project was a great lesson in how to keep everything organized, especially when working with code and electronics, setting us up nicely for more complicated projects down the road.

The task provided a practical experience in embedded system programming, emphasizing the importance of CamelCase coding conventions and modular code organization for peripheral device functionalities.

The laboratory work offered a comprehensive experience that extended beyond the mere toggling of an LED. It fostered an appreciation for the intricacies of embedded system design, from the meticulous organization of code to the thoughtful consideration of hardware-software interactions. The skills and insights gained from this project are invaluable, providing a strong foundation for future endeavors in the field of embedded systems.

**BIBLIOGRAPHY**


**1** EDUCBA: *Introduction to Embedded Systems.* Cursuri electronice online ©2020 [citat 02.02.2021] Available: https://www.educba.com/what-is-embedded-systems/

**2** SPARKFUN: *Driver motor.* Magazin online, ©2020 [citat 10.03.2020]. Available: https://www.sparkfun.com/datasheets/ Robotics/TB6612FNG.pdf.

**3** ARDUINO: *Arduino UNO.* Site-ul oficial al modulelor Arduino, ©2020 [citat 30.03.2020]. Available: https://www.arduino.cc/.

**4** EDUCBA: Introduction to Embedded Systems. Cursuri electronice online ¬©2020 [citat 02.02.2021] Available: https://www.educba.com/what-is-embedded-systems/

**5** SPARKFUN: Driver motor. Magazin online, ¬©2020 [citat 10.03.2020]. Available: https://www.sparkfun.com/datasheets/ Robotics/TB6612FNG.pdf

**6** ARDUINO: Arduino UNO. Site-ul oficial al modulelor Arduino, ¬©2020 [citat 30.03.2020]. Available: https://www.arduino.cc/