

**UNIVERSITATEA TEHNICĂ A MOLDOVEI**

**FACULTATEA: CALCULATOARE, INFORMATICĂ ŞI MICROELECTRONICĂ**

**DEPARTAMENTUL: INGINERIA SOFTWARE ȘI AUTOMATICA**

# **Laboratory work NR. 2.2**

# **Sequential Operating Systems. FreeRTOS**

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

**Verified: Moraru Dumitru**

**Chișinău – 2024**

**1 TASK OF THE LABORATORY WORK**

Creating an application for MCU (Microcontroller Unit) that will run at least 3 tasks using FreeRTOS.

This project aims to develop an application for a microcontroller (MCU) that executes at least three tasks sequentially. The tasks are as follows:

1. Button LED: Changes the LED state upon detecting a button press.

2. Intermittent LED: Flashes the LED while the LED from the first task is off.

3. Variable Increment/Decrement: Increments or decrements a variable upon pressing two buttons. This variable represents the number of repetitions or the time the LED from the second task will be in a certain state.

4. Idle Task: Displays program statuses, such as LED state and a message upon detecting button presses. One implementation could be to set a variable when a button is pressed and reset it when displaying the message, using a provider/consumer mechanism.

The objectives are the following:

* Implement communication between tasks as provider-consumer
* Follow the principles presented in the Sequential Systems course for recurrence and offset
* The reporting task for Sequential with use of STDIO printf() to LCD will be run in  
  FreeRTOS - a separate task.

# **2 PROGRES OF THE WORK**

**2.1 Description**

This program is an example of a multitasking application for a microcontroller unit (MCU) using the FreeRTOS (Real-Time Operating System) library. FreeRTOS allows for the creation of multiple tasks that can run concurrently, enabling efficient utilization of the MCU's resources.

The program consists of several tasks:

Button Task (button\_task):

Monitors the state of a button (LED\_BUTTON).

Signals when the button is pressed using a semaphore (any\_button\_pressed).

Signals UI update using another semaphore (update\_ui).

Flicker Task (flicker\_task):

Controls the flickering state of an LED (RED\_LED) when the button is not pressed.

Adjusts flicker interval based on button presses.

Signals UI update upon button presses.

UI Task (idle\_task):

Handles LCD display and user interface.

Displays messages upon button presses.

Updates UI elements based on semaphore signals.

UI LEDs Task (leds\_control\_task):

Controls the state of LEDs (YELLOW\_LED and RED\_LED) based on button presses and flicker state.

The tasks communicate with each other using semaphores (any\_button\_pressed and update\_ui). Semaphores are synchronization mechanisms used to signal events between tasks in a multitasking environment. In this program, semaphores are employed to notify tasks about button presses and UI updates. When a button is pressed, the corresponding semaphore is given, indicating to other tasks that a button event has occurred. Similarly, when UI elements need to be updated, the update\_ui semaphore is given to trigger UI-related tasks.

FreeRTOS is chosen for its efficiency in managing multiple tasks in real-time embedded systems. It provides features like task scheduling, synchronization primitives (such as semaphores), and memory management tailored for resource-constrained environments. Semaphores, in particular, are essential for coordinating access to shared resources and signaling events between tasks without busy-waiting or excessive CPU usage.

**2.2 Flow Chart**

## 2.3 Circuit

**2.4 Simulation:**

# **CONCLUSION**

# This project demonstrates the implementation of a multi-tasking application on an MCU. The application includes multiple tasks that interact with each other and with the user. The project can be further extended by adding more tasks, implementing a more complex user interface, and using advanced programming techniques.

**BIBLIOGRAPHY**

1. Arduino: Arduino UNO R3. Arduino official site, ©2024 [quote 2024]. Access link: <https://docs.arduino.cc/hardware/uno-rev3/>
2. Serial Monitor: Using the Serial Monitor tool. Arduino official site, ©2024 [quote 2024]. Access link: <https://docs.arduino.cc/software/ide-v2/tutorials/ide-v2-serial-monitor/>
3. Buttons in Arduino. Arduino official site, ©2024 [quote 2024]. Access link: https://docs.arduino.cc/built-in-examples/digital/Button/