UNIVERSITATEA TEHNICĂ A MOLDOVEI
FACULTATEA: CALCULATOARE, INFORMATICĂ ŞI MICROELECTRONICĂ
DEPARTAMENTUL: INGINERIA SOFTWARE ȘI AUTOMATICA

Laboratory work NR. 2.1

Sequential Operating Systems

Executed by: Konjevic Alexandra, gr. FAF-213

Verified: University Lector Moraru Dumitru

Chișinău – 2024

1 TASK OF THE LABORATORY WORK

Implementation of an Application for MCU Running at Least 3 Tasks Sequentially

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 producer-consumer communication between tasks using shared memory
- Find a reasonable interval between tasks
- For a bonus point, implement an extra functional

2 PROGRES OF THE WORK

2.1 Description

In this laboratory work, I created an Arduino sketch that implements a simple scheduler to manage and execute tasks in a cooperative multitasking fashion. The tasks are related to controlling LEDs based on button presses and managing the flickering of a yellow LED. Here's a breakdown of the code:

- **Global Variables:** YELLOW_LED and RED_LED are constants representing the pin numbers for LEDs. TOGGLE_BUTTON, INC_BUTTON, and DEC_BUTTON are constants representing the pin numbers for buttons. Various variables for managing the state of button presses, flickering, and timing intervals.
- **Functions:** pressedButtonTask(): Checks if the toggle button is pressed and prints a message if it has been pressed. checkFlickerYellowTask(): Manages the flickering of the yellow LED. It toggles the state of the isFlicker variable based on a flickering interval. The flickering interval can be adjusted using the increment and decrement buttons. uiTask(): Updates the state of LEDs based on the values of isFlicker and isButtonPressed.
- **Scheduler Class:** Scheduler is a simple cooperative multitasking scheduler class. It has a constructor that takes the maximum number of tasks allowed and dynamically allocates an array for the tasks. addTask(Task task): Adds a task to the scheduler. advanceTick(): Advances the scheduler's internal tick, checks the elapsed time, and executes the current task if the elapsed time exceeds a maximum limit.
- **Setup and Loop:** setup(): Initializes serial communication, sets pin modes, and adds tasks to the scheduler. loop(): Calls advanceTick() on the scheduler in the main loop.

In summary, the code uses a scheduler to organize and execute tasks related to LED control and button presses in an Arduino environment. The tasks are managed in a cooperative multitasking manner to allow for concurrent execution of different functionalities.

2.2 Flow Chart

Next, I will present all the flow chart diagrams for every function in the program, and also, for the Scheduler class.

First of all, above you can see the Scheduler diagram:

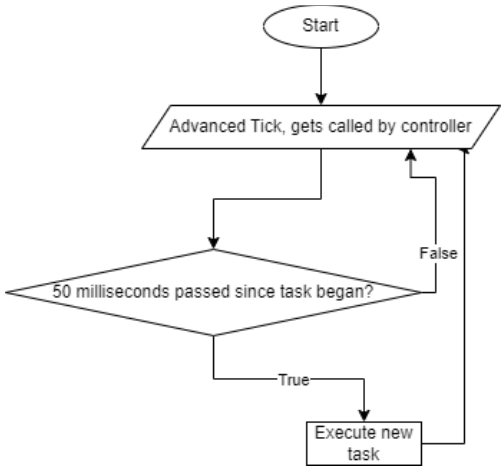

Figure 1. Scheduler loop diagram

The following is the diagram for the flicker LED task

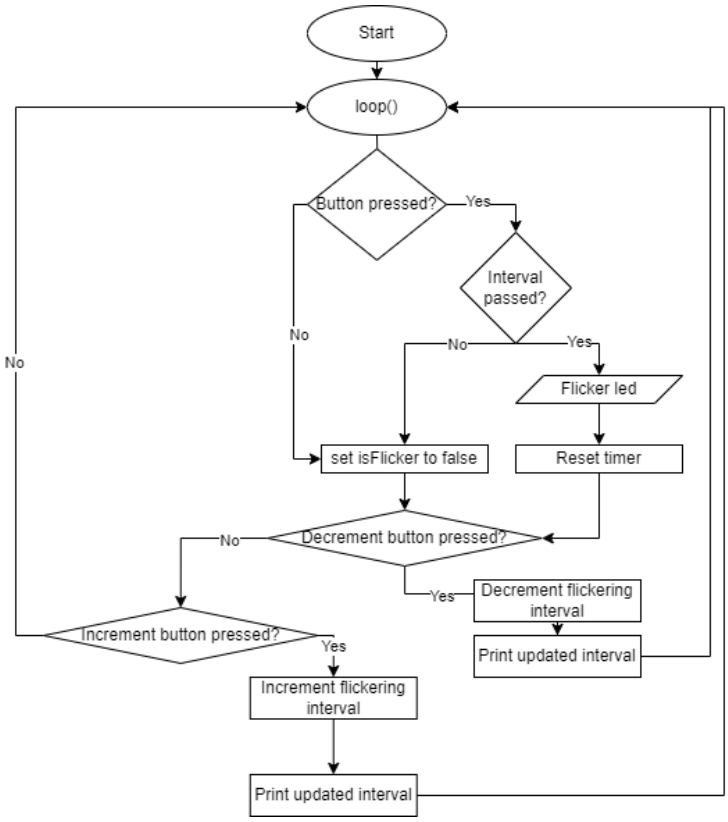

Figure 2. Flicker task diagram

Next, there is the diagram of the function responsible for changing the state of the button that toggles the first led (the red one).

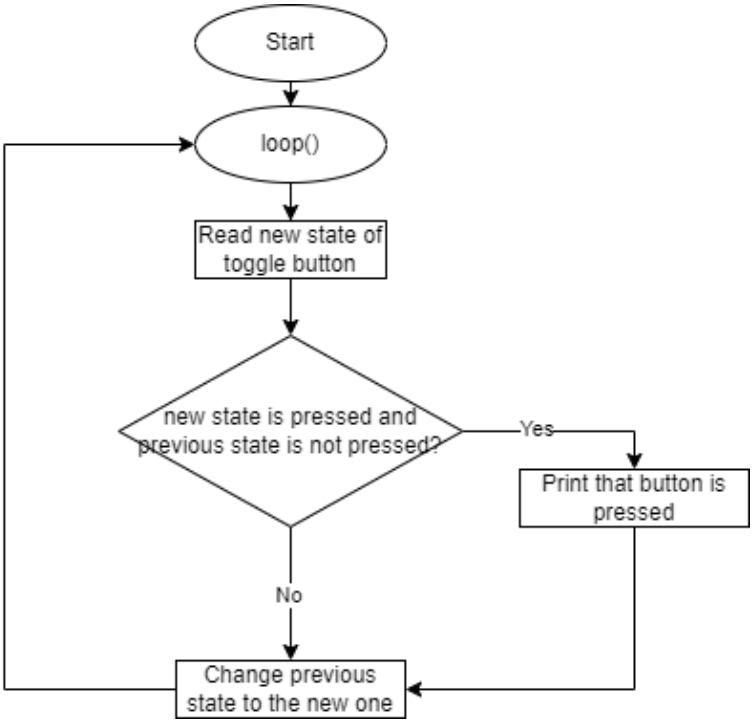

Figure 3. Toggle button task

And next, we have the last diagram - the one that shows the UI task:

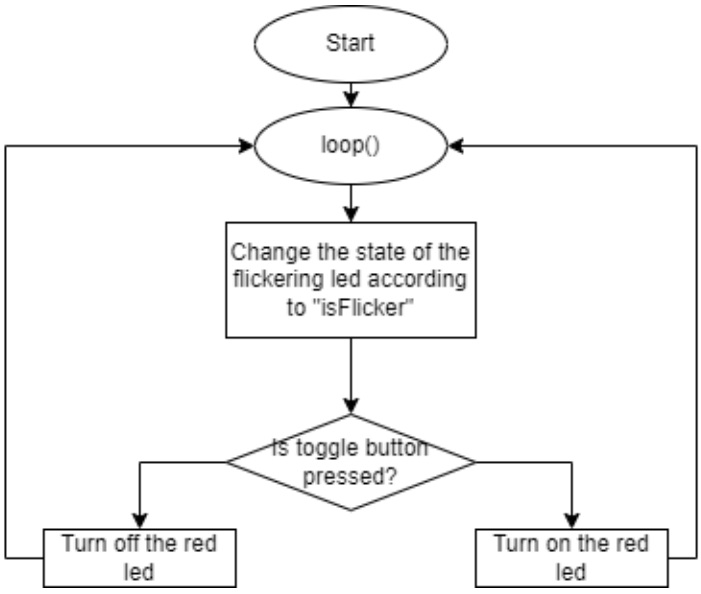

Figure 4. UI task

2.3 Circuit

For achieving the goals set, the components described in section 1.1 needed to be connected to the microcontroller.

Given with the necessary circuitry, the following is the diagram I made using wokwi:

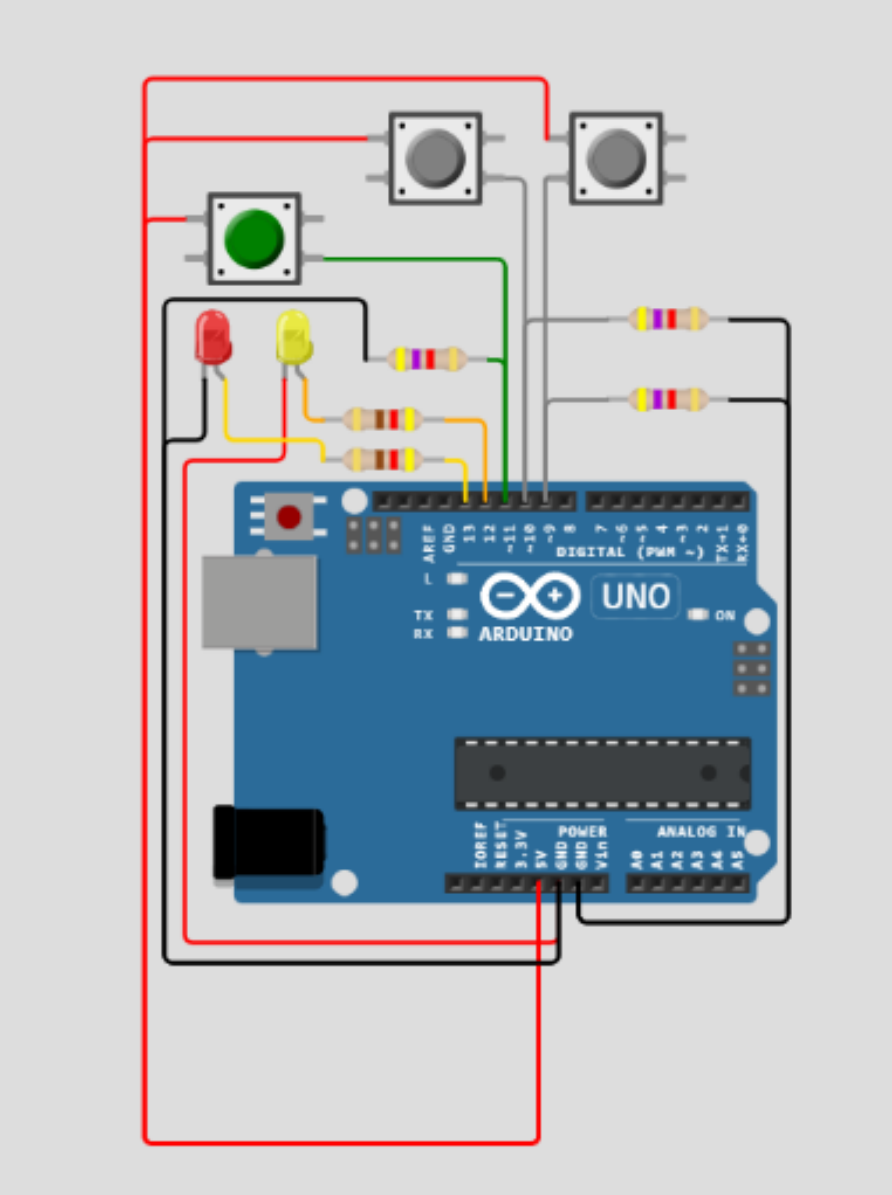

Figure 5. Simulated circuit

2.4 Simulation:

First of all, this is the state of the circuit when only the toggle button is pressed:

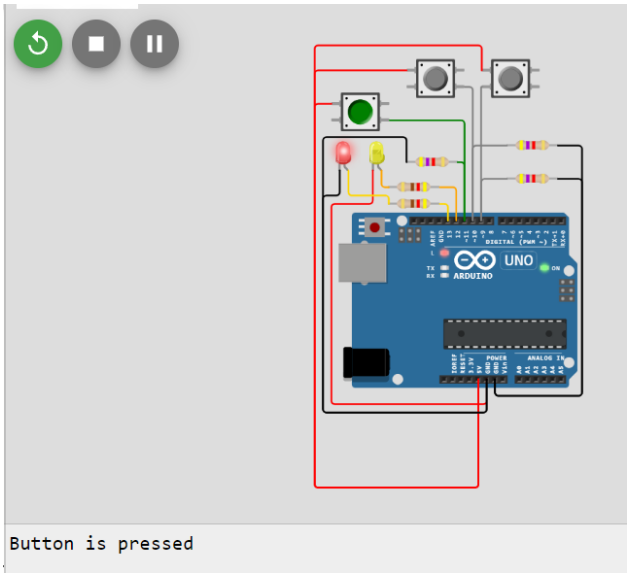

Figure 6. Toggle button pressed

And when the another led is flickering, this is the circuit:

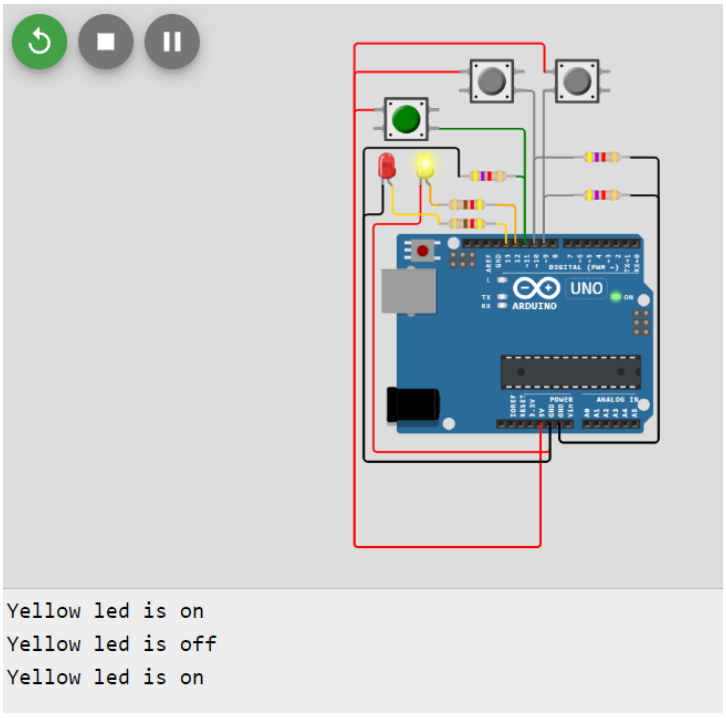

Figure 7. Yellow led flickering

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/