**TECHNICAL UNIVERSITY OF MOLDOVA**

**FACULTY OF COMPUTERS, INFORMATICS AND MICROELECTRONICS**

**DEPARTMENT OF SOFTWARE ENGINEERING AND AUTOMATION**

**Laboratory Work Nr. 2.2**

**FreeRTOS Operating Systems**

**Done by: Chihai Nichita**

**st. gr. FAF-213**

**Verified by: Moraru Dumitru**

**univ. lect.**

**Chisinau, 2024**

# THE TASKS OF LABORATORY WORK

- Button Led - Changing the LED state upon button press: The LED state changes upon detecting a button press.

- Second Intermittent Led when the LED from the first Task is off. An additional LED will intermittently change its state when the LED from the first task is off.

- Incrementing/decrementing a variable's value upon pressing two buttons, representing the recurrence/time the LED from the second task will be in a certain state. Pressing two buttons will increment or decrement a variable representing the number of occurrences/times the LED from the second task will be in a specific state.

- Idle Task will be used to display program states. The Idle Task will be utilized to display program states, such as showing the LED state and displaying messages upon button presses. An implementation could involve setting a variable upon button press and resetting it during message display, using a producer/consumer mechanism.

# 2 PROGRESS OF THE WORK

For this laboratory work, I utilized the Arduino IDE, a software designed for writing and compiling code for the Arduino development board, along with Proteus, a software used for electronic design simulation.

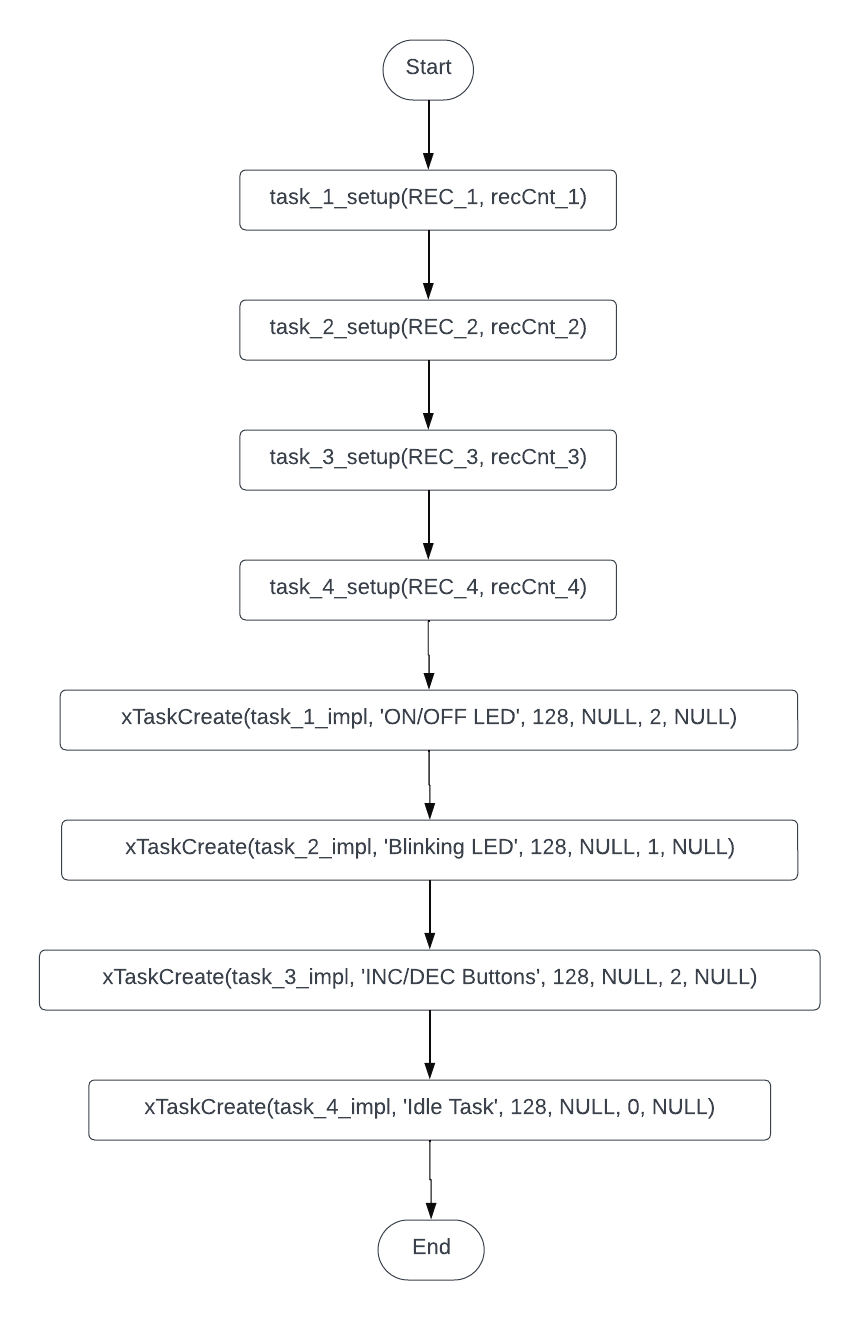
**2.1 Main function description**

**Table 1 Used functions**

|  |  |
| --- | --- |
| **Function** | **Description** |
| setup() | The main setup function provides an order for the setup for each task, invoking their respective setup functions and creating task instances with specific parameters using ‘xTaskCreate’. |
| loop() | The main loop function (`loop`) is deliberately left empty. In FreeRTOS-based programs, the tasks handle the execution, rendering the main loop inactive. This design allows for concurrent execution of tasks with varying priorities, fostering a multitasking environment on the Arduino platform. Each task serves a distinct purpose, such as controlling LEDs, managing button inputs, or acting as an idle task. |
| xTaskCreate | The main purpose of the xTaskCreate is to assign to each task a name, stack size, priority, and a distinct implementation function. The priorities differ, with `task\_1\_impl` having a priority of 2, `task\_2\_impl` a priority of 1, and `task\_3\_impl` a priority of 2, while `task\_4\_impl` is assigned a priority of 0, denoting an "Idle Task". |

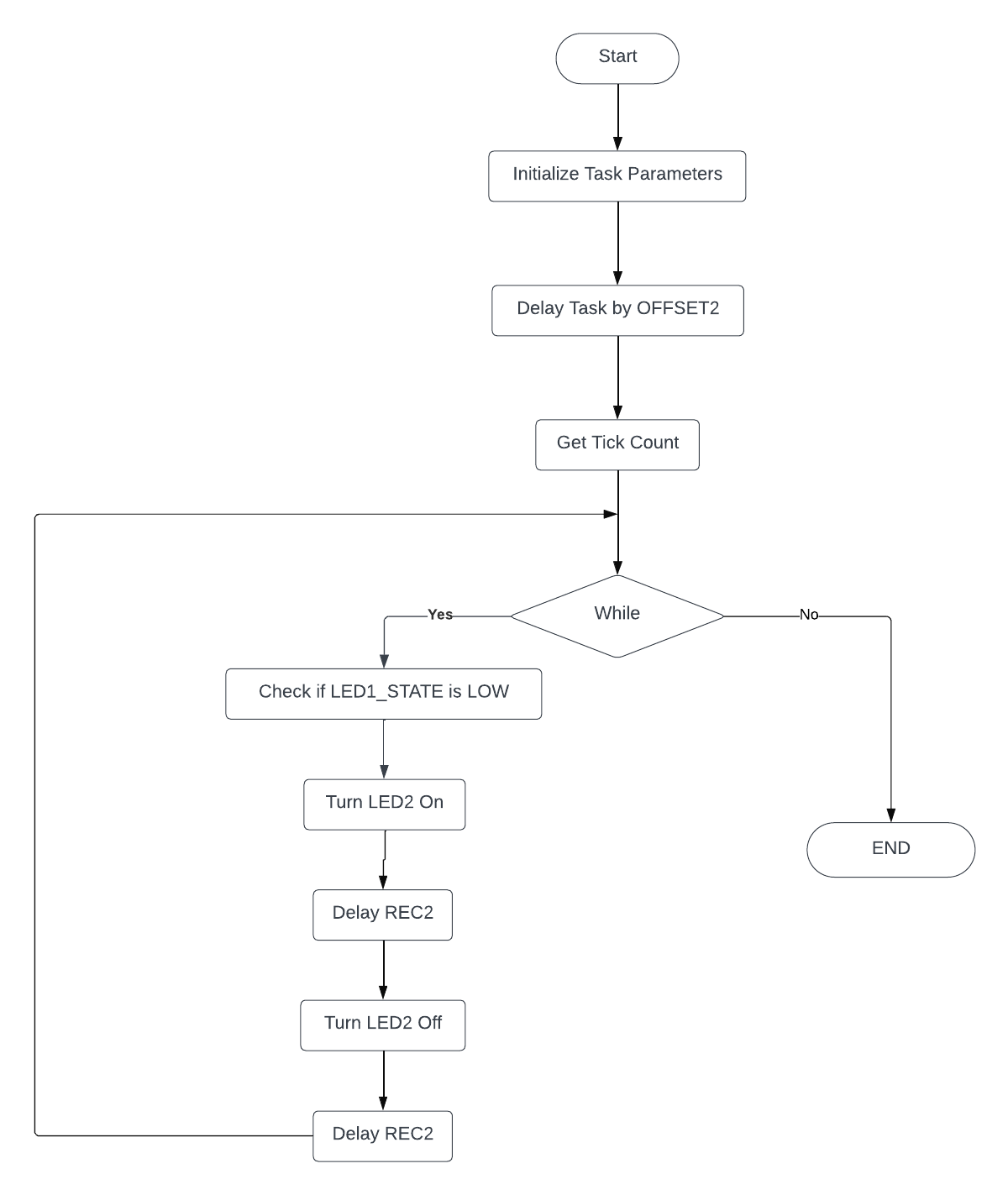
**2.2 Block diagram**

This block diagram provides a visual representation of the main components and their interactions for my Arduino program.



**Figure 1 Block diagram**

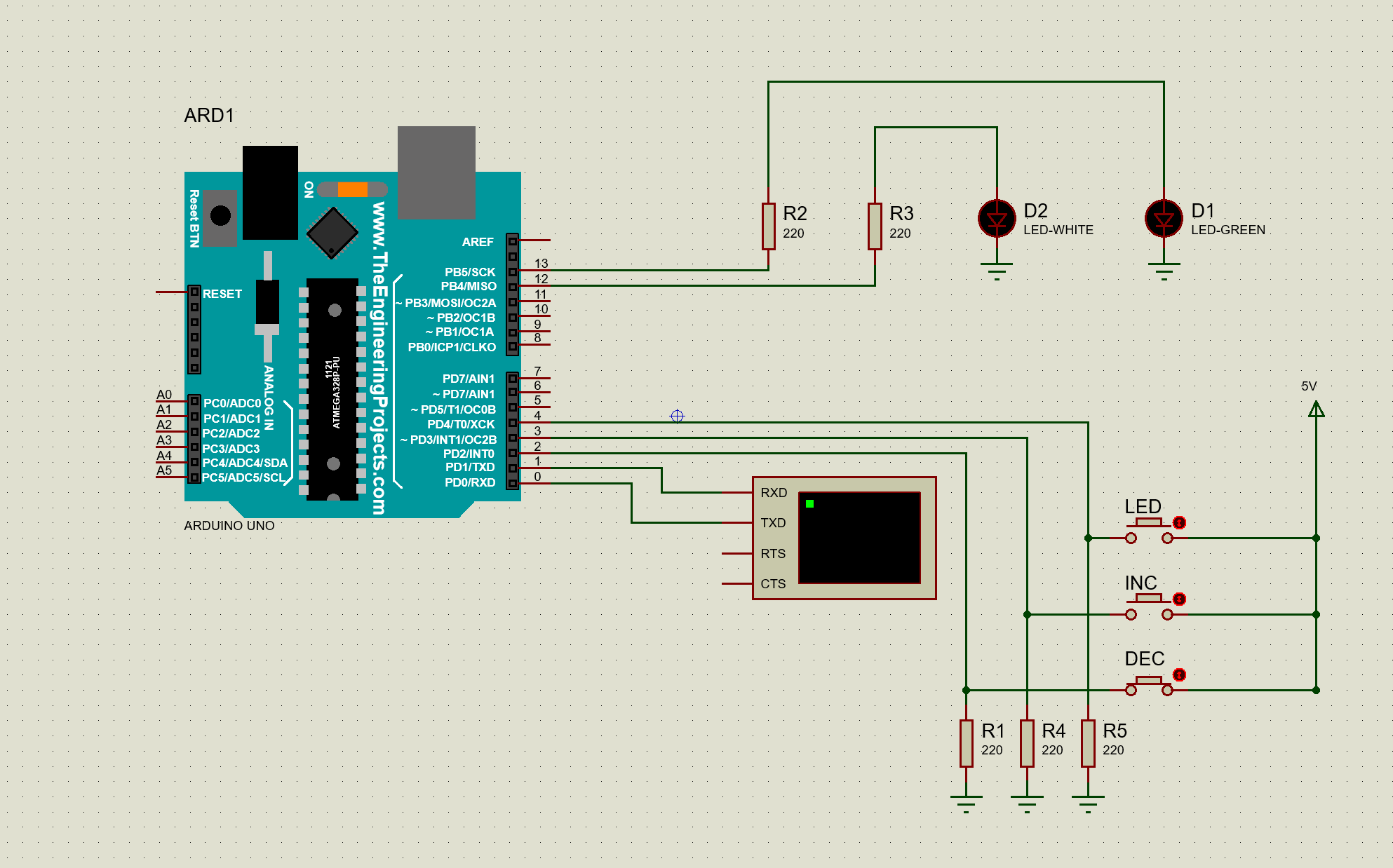
This block diagram provides one of the tasks implementations, in this case task 1, logic and use of certain freeRTOS functions [[1](#one)].

****

**Figure 2 Block diagram**

**2.3 Simulated schematic**

This is the simulated Arduino Uno scheme assembled in Proteus.

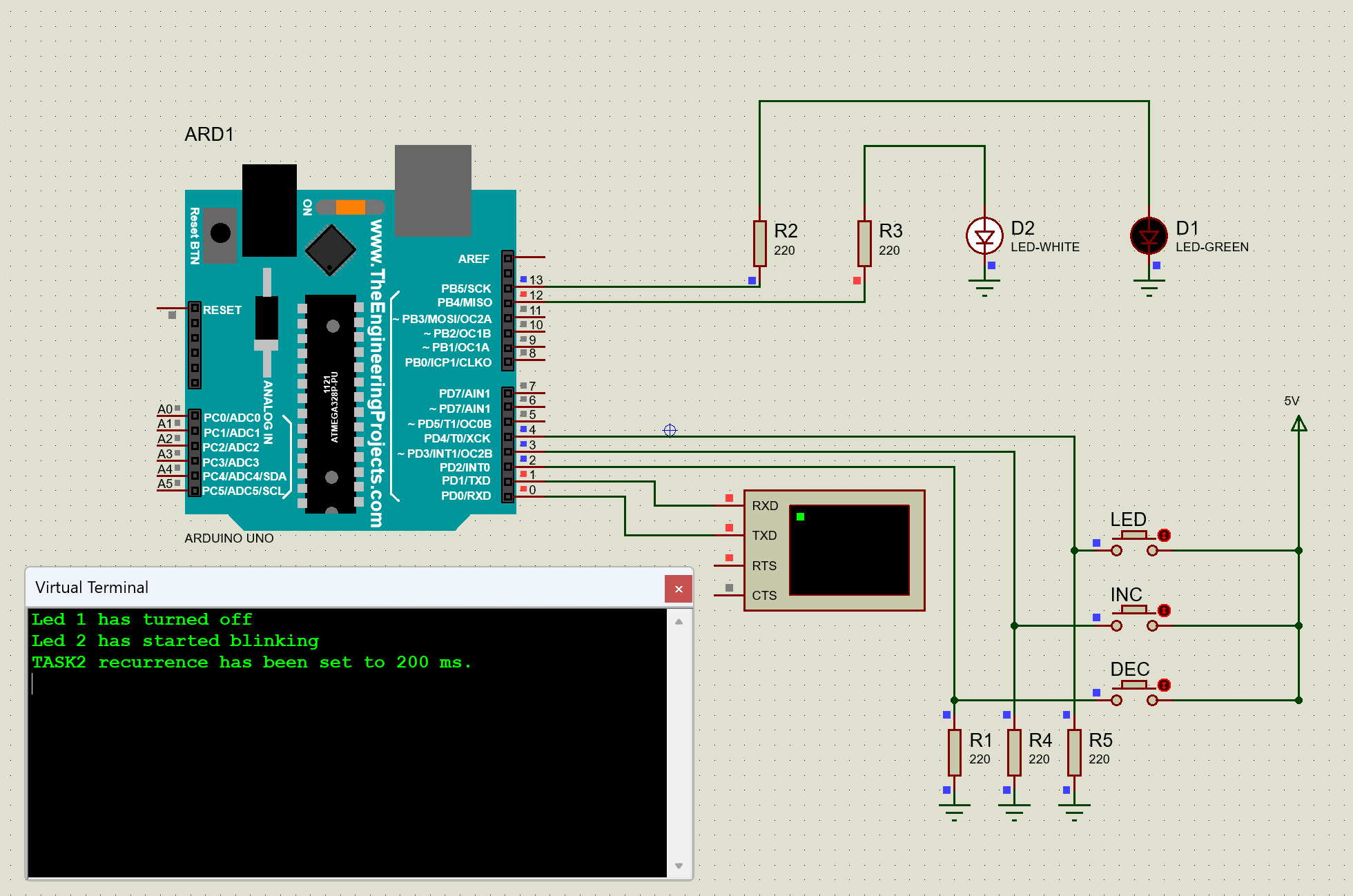


**Figure 3 Electrical schematic**

This circuit is composed of several elements such as: Arduino Uno, five resistors, two LEDs, three push buttons, a power source and a virtual terminal. In order to complete this laboratory task I assembled this circuit which meets all the requirements that were set at the beginning [[2](#two)].

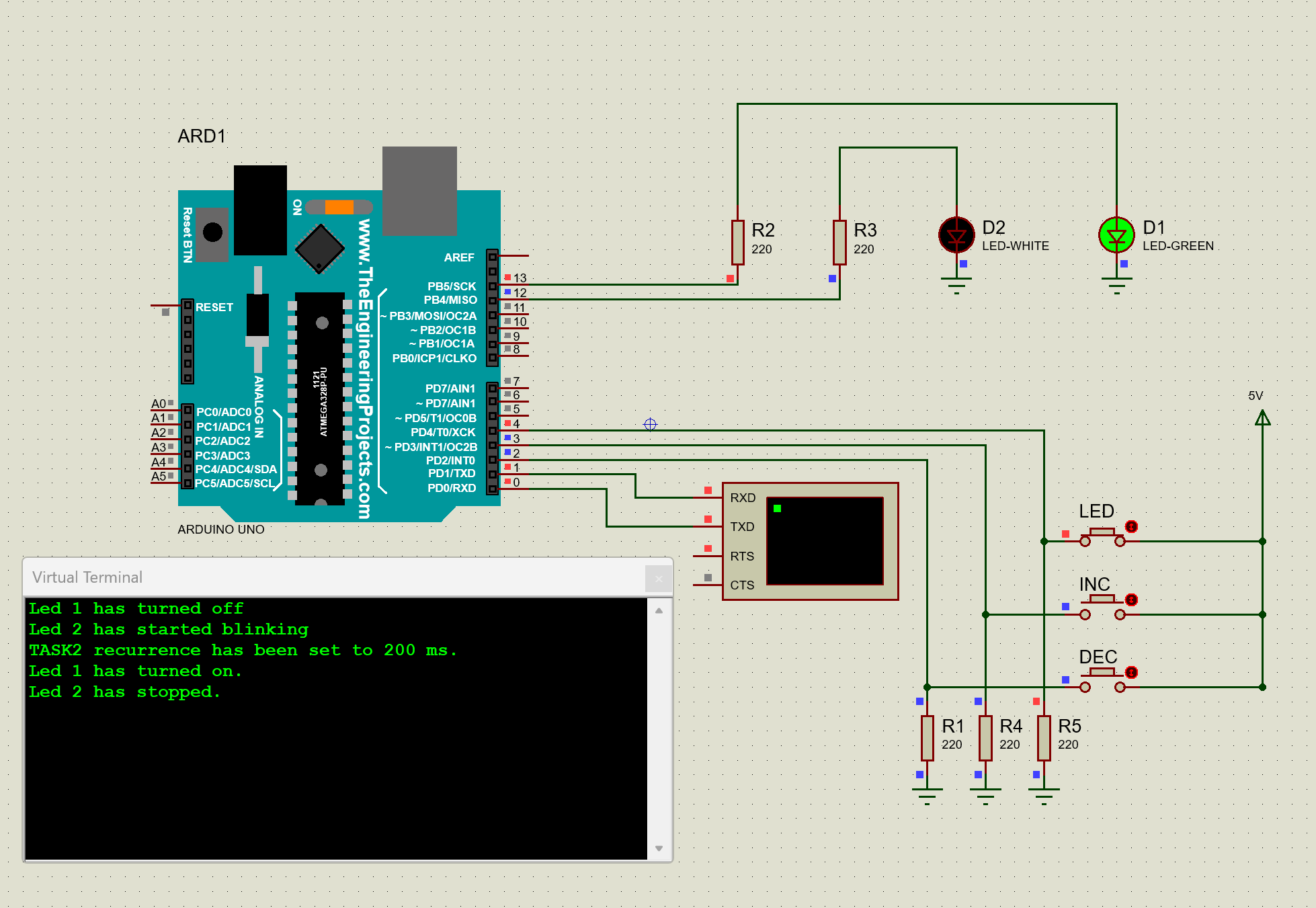
**2.4 Photos of running simulation**

This picture is an example of the circuit when the first led is turned off and the second one is blinking.

****

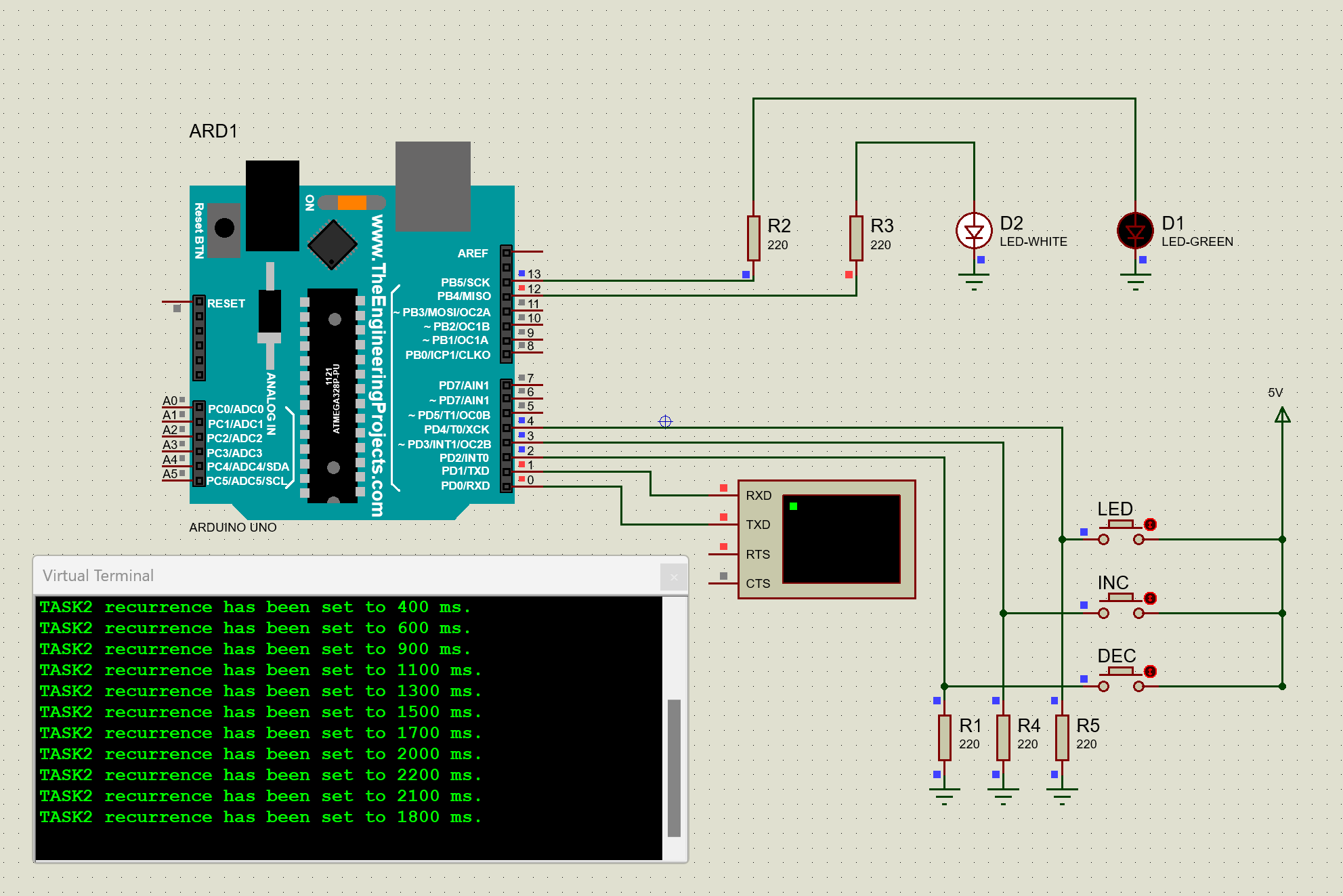
**Figure 4 Blinking LED on**

In the second example I pushed the ‘LED’ button that triggers the first led to turn on/off, in this case it was turned on and the second led that was blinking has stopped doing that. All the actions that have been made are printed in the virtual terminal, which acts as a log.



**Figure 5 Blinking LED off**

This image shows the implementation of task 3 with decrementing and incrementing the recurrence of the blinking led. This means that when we increase the recurrence the blinking led will take more time between phases, thus if we decrease the recurrence the led will blink faster. Also there were implemented boundaries to the upper and lower side, 3000ms and 100ms accordingly.



**Figure 6 Incrementing and decrementing the recurrence**

# CONCLUSION

This laboratory work was a great experience for me. I learned a lot of small concepts, rules that helped me realize this project.

First of all, I used several LEDs, resistors and buttons. I connected them together so that they would work properly. So that when the user interacts with them, it will trigger the right commands behind it.

Second of all, I was able to create a simple scheme that has helped me simulate my project in real-time. I managed to correctly connect the elements used in my project, such as Arduino Uno, five resistors, green and white LED, virtual terminal and a power source.

To sum up, this laboratory work has deepened my understanding of electronic circuits and Arduino development. The project’s success will contribute to my future projects in the field of embedded systems.

# BIBLIOGRAPHY

**1** freeRTOS: vTaskDelay. [online], [accesed 09.03.2024]. Available at: [This page describes the RTOS vTaskDelay() FreeRTOS API function which is part of the RTOS task control API. FreeRTOS is a professional grade, small footprint, open source RTOS for microcontrollers.](https://www.freertos.org/a00127.html)

**2** Proteus Documentation: Information for building circuits in proteus. [online], [accesed at 09.03.2024]. Available at: [How to Simulate Arduino Projects Using Proteus | Arduino | Maker Pro](https://maker.pro/arduino/projects/how-to-simulate-arduino-projects-using-proteus)