Table of Contents

1. STM32 Control of build-in LEDs with buttons 2
1.1. Prerequisites 2
1.2. Background 2
1.2.1. LED connection 2
1.2.2. Buttons connection 3
1.2.3. Interrupts 3
1.2.4. Tasks to be implemented 3

1. STM32 Control of build-in LEDs with buttons

This exercise presents how to control three LEDs with buttons mounted on the Nucleo STM32WB55 board.

1.1. Prerequisites

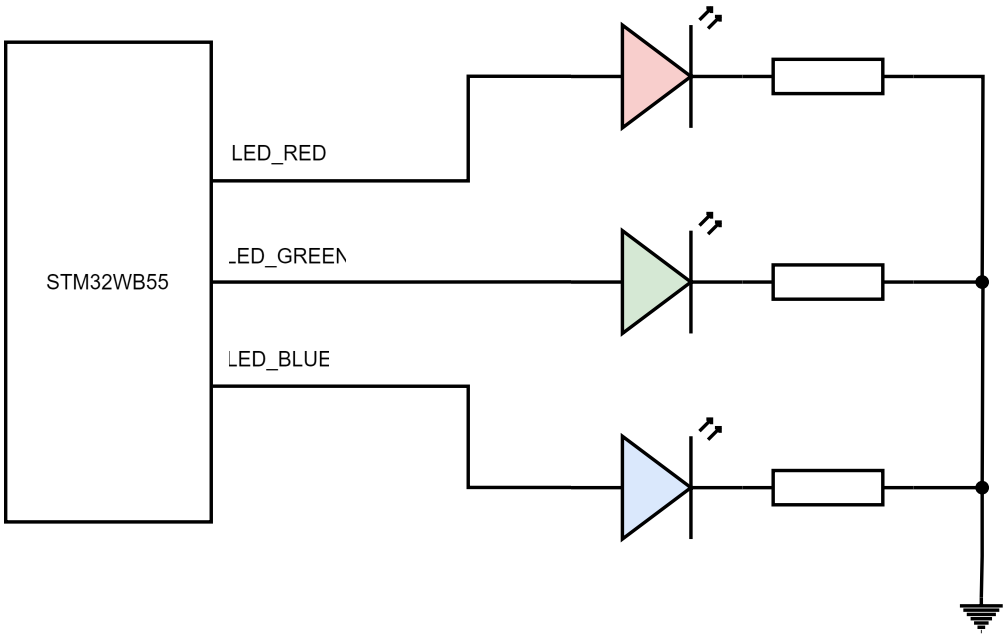
A good understanding of the digital GPIO pins operation and principles of interrupts is necessary. We use the Arduino Core STM32 framework (stm32duino)[1)] for STM32WB55. During this laboratory, no additional external libraries will be required for the project.

1.2. Background

LEDs and buttons are electrically connected to microcontroller's pins. Below we will show how they are connected because it determines their behaviour depending on the software.

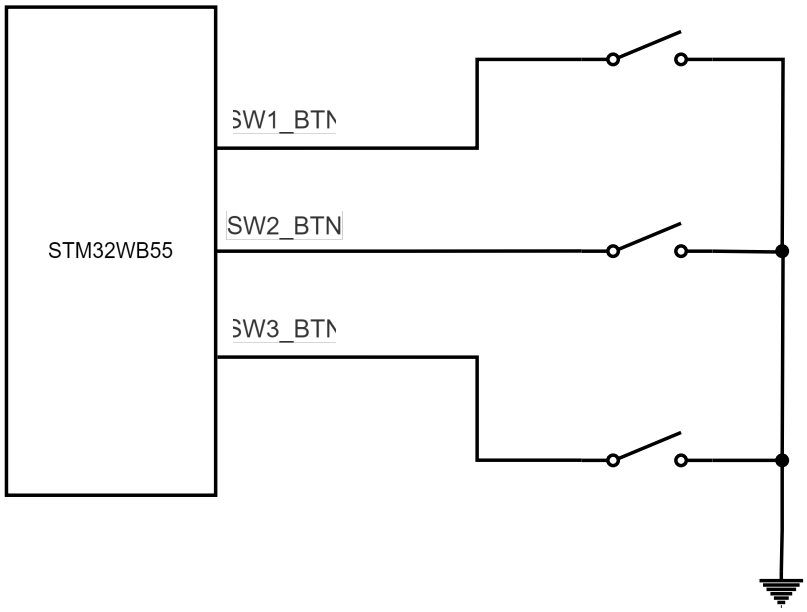
1.2.1. LED connection

LEDs are electrically connected to microcontroller's pins. LEDs have 3 different colours (Red, Green, and Blue). The red LED is connected to the pin PB1, the green LED to the pin PB0, and the blue LED to the pin PB5. LED is lit when the pin state is “1”, and dark when the pin state is “0”. LEDs are easily accessible in the software as digital pins named LED_RED, LED_GREEN, and LED_BLUE respectively. Each LED can be controlled with the Arduino standard DigitalWrite() function to switch it on or off.



1.2.2. Buttons connection

The buttons on board are named SW1, SW2, and SW3, and are connected to pins PC4, PD0, and PD1 respectively. In the software, the names of the pushbuttons are SW1_BTN, SW2_BTN, and SW3_BTN. Pushbuttons are easily accessible in the software by their names using the Arduino function DigitalRead(). Buttons are connected to the ground, so the reading gives the “0” state result if the button is pressed and the “1” state otherwise.



1.2.3. Interrupts

The button can control LED in a simple way with a pair of functions digitalRead() and digitalWrite(). The drawback of this approach is the dependency of the reading and writing time on the current flow of the program. If the program calls the Delay() function it's impossible to perform other actions, including controlling the LED state. The asynchronous approach would allow us to use the Delay() or other blocking functions together with the instant reaction on the button press. This can be achieved with interrupts. It is possible to call the interrupt handler function as the callback attached to the button press event. To define the attachment you can use the attachInterrupt() function. It requires three parameters: pin number, address of the callback function, and event type. Attaching switch 1 with the SW1_callback function which is called every time the button is pressed can look like in the following line.

```
attachInterrupt (SW1_BTN, SW1_callback_f, FALLING);
```

1.2.4. Tasks to be implemented

1. Write the program that will switch the red LED for one second, then the green LED for one second, and the blue LED for one second repeatedly. Use the Delay() function. This part is to be familiar with the environment, GPIO controlling, and the STM32WB55 board.
2. Write the program that lights the blue LED when the SW1 button is pressed, green RED when SW2 is pressed and red LED when SW3 is pressed.
3. Write the program that controls the LEDs with interrupts. One LED should blink periodically using the Delay() function, and the second LED should be controlled asynchronously with two buttons. One

button press should turn the LED on, second button should turn the LED off.

1)

“stm32duino”, https://github.com/stm32duino

From:
https://dokuwiki.zmitac.aei.polsl.pl/ - **RAU6 Dokuwiki**

Permanent link:
https://dokuwiki.zmitac.aei.polsl.pl/en:edu:zmitac:macro:embedeed:lab:stm32-gpio?rev=1710690696

Last update: **2024/03/17 16:51**