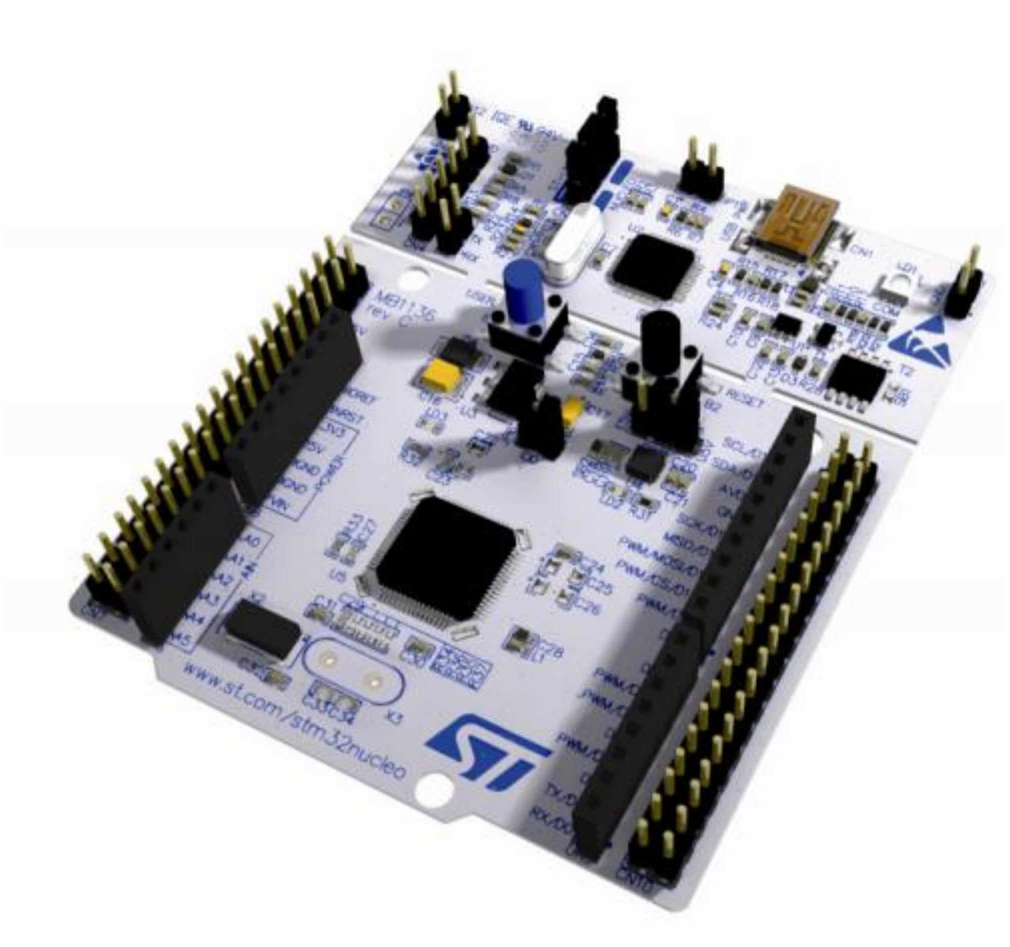
Technical Report

Assignment 1 – GPIO



Course title: Embedded Systems Semester 3

Teacher: Rene Bakx, Robert Verheijen

Date 11.09.2024

Author: Simeon Stoilov

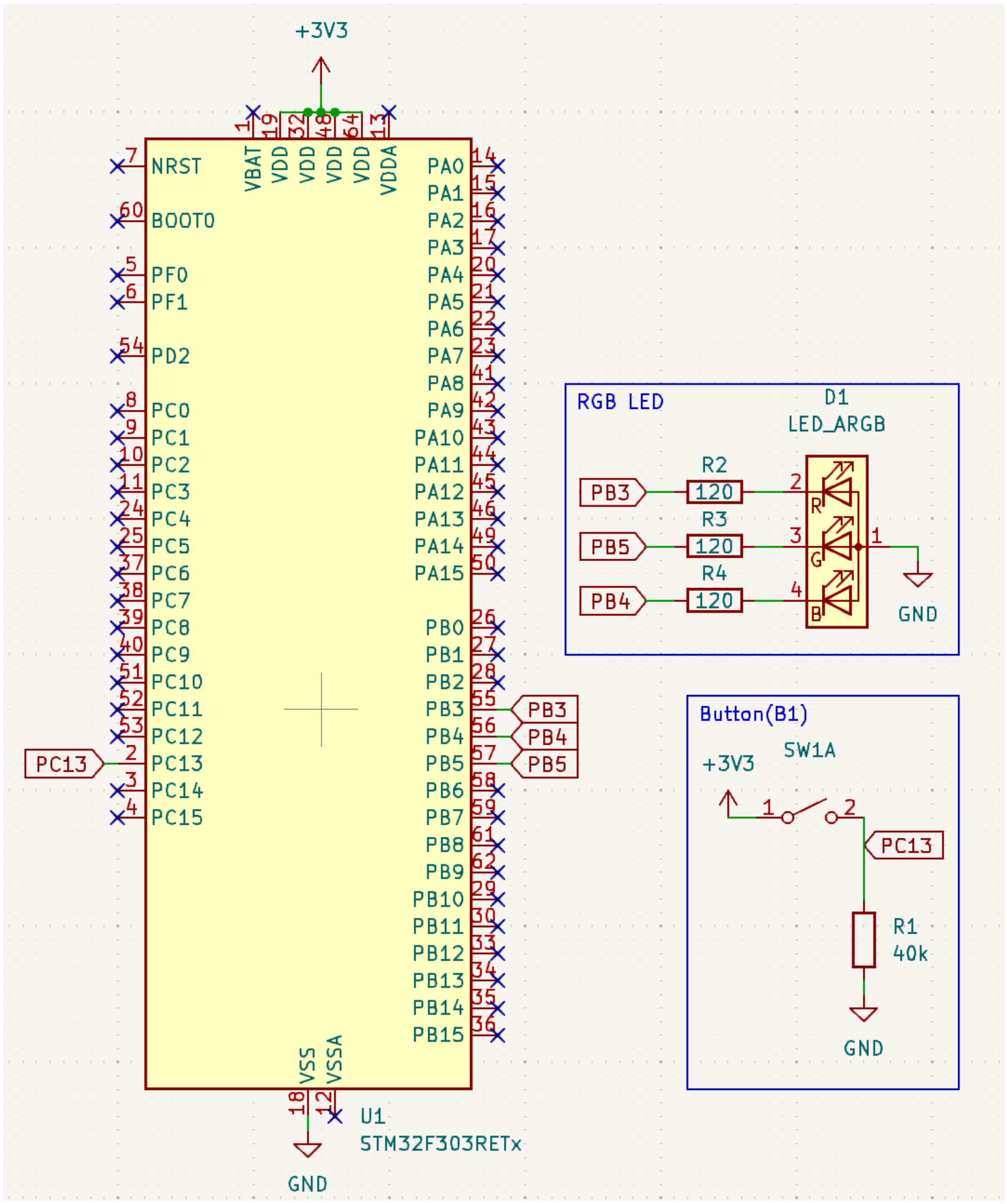
Version: 1.0

Abstract

In this technical report is documented the development of a “Simeon Says” Game using the STM32 Nucleo-F303RE MCU. The game uses a RGB LED and a push button for the user to interact with the system. The game generates a random sequence of LED blinks, which the player must replicate by pressing the button the same amount of times the LED blinked. The project utilizes the CMSIS framework to directly manipulate the registers of the GPIO, without relying on higher level libraries. Successful completion of the game is indicated by a green light at the end of the round, while a failure is indicated by a red light.

Contents

Connection Diagram



Microcontroller connections

Button

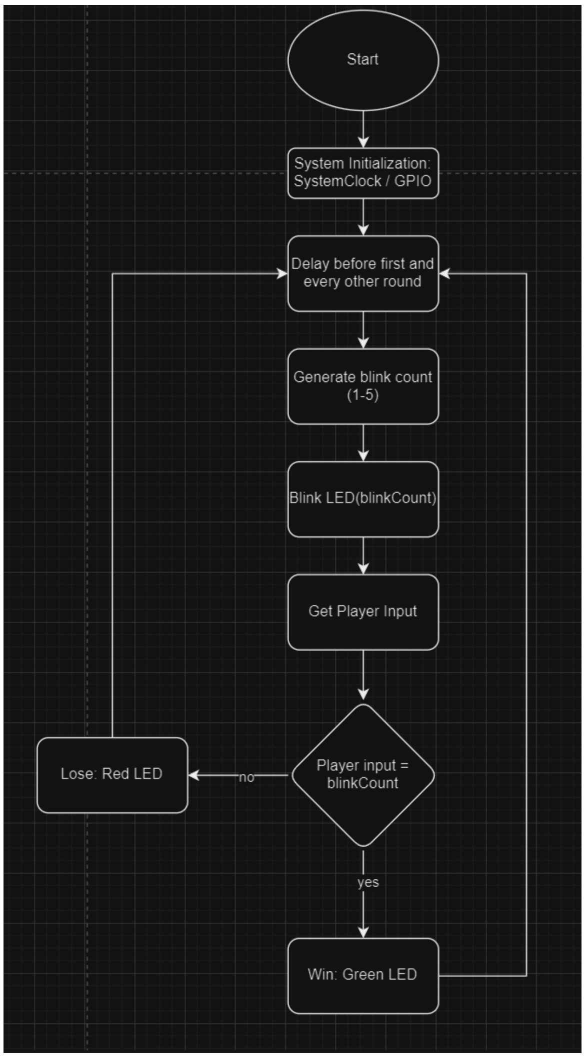
For this system the internal button is used, it is connected to pin PC13 and uses an internal pull-down resistor to detect a button press. One of the pins of the button is connected to VDD (3.3V) and the other is connected to pin PC13, the pull-down resistor (25-55kΩ, typical 40k) and to GND(VSS).

RGB LED

The RGB LED has 4 pins:

- Pin1 – GND – connected to pin VSS(GND) on the MCU. This is a shared GND line for all three diodes
- Pin2 – Red diode – connected to pin 4 (PB3) on the MCU through resistor R2 (120Ω)
- Pin3 – Green diode – connected to pin 5 (PB5) on the MCU through resistor R3 (120Ω)
- Pin4 – Blue diode – connected to pin 6 (PB4) on the MCU through resistor R4 (120Ω)

Flowchart


Start
System Initialization: SystemClock / GPIO
Delay before first and every other round
Generate blink count (1-5)
Blink LED(blinkCount)
Get Player Input
Player input = blinkCount
no
Lose: Red LED
yes
Win: Green LED

In this flowchart is represented the loop of the game. The first thing to happen after starting the system is the System Initialization where we call the functions to initialize the system clock and the GPIO. After that we get into the loop of the game, to begin we give the user a delay before starting, followed by generating the blink count and executing the blinks using the RGB LED. After the blinks the user is given a few seconds to input the same number of blinks using the push button on the board. Then we check if the input amount equals the generated amount, if so a win condition is executed, if not the lose condition is executed. This loop does not have an end condition and will run until the system is turned off.

Code Explanation

Main

The main program consists of the main loop and calling the initializations from the other modules. It consists of the main function which starts by initializing the system clock configuration and the input and output pins used.

Initialization

The initialization module is essential for the system and it consists of a header and a implementation file.

Header

In the header file we have the declarations of the pins used for the LEDs (output) / Button (input), and the functions – SystemClock_Config(), GPIOx_Init()

Implementation

The implementation file consists of the following functions:

- void GPIOx_Init(void)
  - Enables the clocks for the B/C groups of GPIO
  - Configures the pins used for the RGB LEDs as output
  - Configures the pin used for the button as input and enables the internal pull-down resistor to pull the line low when the button is not pressed and high when it's pressed
- void SystemClock_Config(void)
  - The system clock is configured to ensure the MCU operates at the correct frequency.
  - Sets up the flash memory latency
  - Enables the HIS
  - Configures the APB1 prescaler
  - Activates the PLL to use as the system clock source

Game

The game module contains the logic that runs the Simeon Says Game.

Header

The header file declares the functions used in this module:

- uint32_t getPlayerInput(void);
- void blinkLED(uint32_t);
- void checkGameEnd(uint32_t);
- void simeon_says_game(void);
- void Delay(uint32_t time_ms);

Implementation

- uint32_t getPlayerInput(void);
  - This function keeps gives a 5 second timer for the user to input the presses needed to win the game
  - It uses a loop to count the number of presses until the timer runs out
  - In the loop it checks if the registers value to see if the button is pressed (the value of the register is "high"), this increases the press counter.
  - A simple debounce method is used to ensure that the button presses are valid. It consists of a while loop that does nothing until the button is released
- void blinkLED(uint32_t blinkCount);
  - This function blinks the LEDs at 2Hz for a set amount of times
  - In the loop we manipulate values of the registers for each LED pin to 1 for half a second and then to 0 for half a second.
- void checkGameEnd(uint32_t blinkCount);
  - At the start of this function it calls the getPlayerInput function to get the number of presses and then compares it to the pre generated blink count amount.
  - If the player has matched the inputs to the count the register for the green pin is set to 1 for 2 seconds and after that it is reset to 0
  - If the player has not matched the inputs to the count the register for the red pin is set to 1 for 2 seconds and after that it is reset to 0
- void Delay(uint32_t time_ms)
  - Delay function as in the example Hello World code.
  - Two nested for loops are used in combination with the system clock to count and do nothing, essentially a blocking delay function.
- void simeon_says_game()
  - This function is used to call all of the other functions in the game logic.
  - First the delay function is called to give time for the user to react
  - Second the blink count is generated using the rand function of the stdlib library
  - Third the blinkLED function is called using the blinkCount
  - Fourth the checkGameEnd function is called with the blinkCount

Demo

A demo video of the game will be uploaded with the assignment submission. In the video I showcase the functionality of the game while playing four rounds.

- Round 1 I replicate the sequence by pressing the button 2 times for the 2 blinks and win(green LED on).
- Round2 The game lights up the LED twice but I press it 3 times so I lose(red LED on).

- Round 3 The game lights up the LED three times and so I press it the same amount and win
- Round 4 The LED is lit up once but I don’t press the button at all so I lose