Politechnika Wrocławska

STEROWNIKI ROBOTÓW

Lab 01 - wykorzystanie narzędzi STM32CubeIDE oraz STM32CubeMx do budowy programu mrugającej diody z obsługi przycisku

Wydział i kierunek	W12N, Automatyka i Robotyka
Numer grupy	2
Autor i numery albumów	Michał Markuzel 275417
Termin zajęć	Środa 15:15-16:55
Kod grupy zajeciowej	W12AIR-SI0102G
Prowadzący	dr inż. Wojciech Domski

1 Wprowadzenie

Celem ćwiczenia było zapoznanie się z podstawowymi narzędziami do pracy z mikrokontrolerami STM32 oraz praktyczna implementacja prostych funkcjonalności: migania diody LED oraz obsługi przycisku z eliminacją drgań styków (debouncing). Do realizacji zadania wykorzystano środowisko **STM32CubeIDE** oraz narzędzie **STM32CubeMX**.

2 Opis ćwiczenia

Zadanie obejmowało:

- Konfigurację projektu w **STM32CubeMX** i **STM32CubeIDE**.
- Implementację programu odpowiedzialnego za miganie diodą LED.
- Rozszerzenie funkcjonalności o obsługę przycisku i implementację debouncingu.
- Wgranie oraz testowanie programu na mikrokontrolerze STM32.

Pracowaliśmy na mikrokontrolerze **STM32L476RG**, który znajduje się na płytce ewaluacyjnej **NUCLEO-L476RG**.

3 Realizacja ćwiczenia

3.1 Tworzenie projektu i konfiguracja mikrokontrolera

Projekt został utworzony w **STM32CubeIDE** i skonfigurowany w **STM32CubeMX**. W konfiguracji ustawiono:

- **GPIO**:
  - **PA5 (LED)** jako wyjście dla diody LED.
  - **PC13 (B1)** jako wejście dla przycisku.
- **Clock Configuration**: ustawienia zegara zgodnie z domyślnymi wartościami.
- **Generowanie kodu**: wygenerowano kod inicjalizacyjny dla mikrokontrolera STM32.

3.2 Implementacja migania diodą LED

Kod odpowiedzialny za miganie diodą LED wykorzystuje funkcję `HAL_GPIO_TogglePin()` oraz opóźnienie `HAL_Delay()`:

```
while (1) {
    HAL_GPIO_TogglePin(LED_GPIO_Port, LED_Pin);
    HAL_Delay(500);
}
```

Po wgraniu programu na mikrokontroler dioda poprawnie migała w zaprogramowanym odstępie czasu.

3.3 Obsługa przycisku i implementacja debouncingu

Modyfikacja kodu pozwoliła na sterowanie diodą za pomocą przycisku. Aby wyeliminować drgania styków, zastosowano metodę programową opartą na odczycie wartości z przycisku w określonych odstępach czasu.

```
uint32_t last_time_pressed = HAL_GetTick();
uint32_t curr_time = HAL_GetTick();
uint8_t last_state = 0;

while (1) {
    curr_time = HAL_GetTick();

    if ((HAL_GPIO_ReadPin(B1_GPIO_Port, B1_Pin) == 0) && ((curr_time - last_time_
        last_time_pressed = curr_time;

        if (last_state == 0) {
            HAL_GPIO_TogglePin(LED_GPIO_Port, LED_Pin);
        }
        last_state = 1;
    }

    if (HAL_GPIO_ReadPin(B1_GPIO_Port, B1_Pin) == 1) {
        last_state = 0;
    }
}
```

Po implementacji, dioda poprawnie zmieniała swój stan po pojedynczym naciśnięciu przycisku.

4 Wnioski

Ćwiczenie pozwoliło na praktyczne zaznajomienie się z obsługą GPIO w STM32 oraz działaniem funkcji HAL. Implementacja debouncingu umożliwiła eliminację fałszywych sygnałów wejściowych. Narzędzia **STM32CubeMX** oraz **STM32CubeIDE** okazały się pomocne w szybkim przygotowaniu projektu.

5 Research: Cyfrowe wejścia/wyjścia w STM32

Mikrokontrolery STM32 są wyposażone w piny GPIO (General Purpose Input/Output), które umożliwiają komunikację z urządzeniami zewnętrznymi. Piny te mogą działać w trybach wejść lub wyjść cyfrowych, przyjmując stany logiczne 1 (wysoki) lub 0 (niski). Celem zadania domowego było zapoznanie się z konfiguracją tych pinów oraz ich zastosowaniem w podstawowych aplikacjach cyfrowych.

5.1 Piny cyfrowe w STM32

W mikrokontrolerach STM32 piny GPIO mogą pełnić funkcje wejść lub wyjść cyfrowych. Oznacza to, że mogą przyjmować tylko dwa stany logiczne: stan wysoki (logiczna 1) i stan niski (logiczna 0)

5.2 Tryby konfiguracji pinów cyfrowych

Piny GPIO w mikrokontrolerach STM32 mogą być konfigurowane w różnych trybach, w zależności od ich roli w systemie. Tryby te zależą od tego, czy pin jest używany jako wejście, wyjście, czy pełni funkcję alternatywną.

5.3 Wejście (Input)

Dla pinów skonfigurowanych jako wejścia cyfrowe dostępne są trzy podstawowe tryby:

- **Input Floating**: Pin znajduje się w stanie „zawieszenia" (tzw. wysokiej impedancji), czyli nie ma określonego poziomu napięcia. Pin może przyjmować stan wysoki lub niski, zależnie od podłączonego zewnętrznego źródła. Jest to przydatne, gdy pin nie jest podciągany ani do Vcc, ani do GND.
- **Input Pull-up**: Pin jest podciągany do napięcia Vcc za pomocą wewnętrznego rezystora. W tym przypadku pin ma domyślnie stan wysoki, chyba że zewnętrzne urządzenie obniży go do stanu niskiego.
- **Input Pull-down**: Pin jest podciągany do masy (GND) za pomocą wewnętrznego rezystora. Pin ma domyślnie stan niski, chyba że zewnętrzne urządzenie podniesie go do stanu wysokiego.

5.4 Wyjście (Output)

Piny GPIO skonfigurowane jako wyjścia cyfrowe mogą działać w dwóch głównych trybach:

- **Output Push-Pull**: Jest to standardowy tryb wyjścia, w którym pin może generować zarówno stan wysoki (Vcc), jak i niski (GND). Pin działa wtedy jako aktywne źródło lub odbiornik prądu, co zapewnia stabilne poziomy logiczne zarówno dla stanu wysokiego, jak i niskiego.
- **Output Open-Drain**: W tym trybie pin może generować tylko stan niski lub pozostawać w stanie wysokiej impedancji. Aby pin mógł generować stan wysoki, potrzebuje zewnętrznego rezystora podciągającego do Vcc. Ten tryb jest często stosowany w systemach, gdzie wiele urządzeń współdzieli ten sam pin, na przykład w komunikacji I2C.

5.5 Funkcje alternatywne pinów GPIO

Piny GPIO mikrokontrolerów STM32 mogą pełnić różne funkcje alternatywne, które są przypisane do nich w zależności od konfiguracji. Funkcje te są często związane z interfejsami komunikacyjnymi lub innymi peryferiami mikrokontrolera. Przykładowe funkcje alternatywne dostępne w STM32:

- **USART (Universal Synchronous Asynchronous Receiver Transmitter)**: Używane do komunikacji szeregowej, zarówno w trybie synchronicznym, jak i asynchronicznym. Pin może pełnić funkcję odbioru danych (RX) lub nadawania danych (TX).
- **SPI (Serial Peripheral Interface)**: Interfejs szeregowy, który może obejmować piny do przesyłania danych (MOSI), odbierania danych (MISO), wybierania urządzenia (CS) i zegara (SCK).
- **I2C (Inter-Integrated Circuit)**: Interfejs szeregowy, który wymaga dwóch linii - danych (SDA) i zegara (SCL). Piny GPIO mogą pełnić funkcję komunikacji I2C.

- **PWM (Pulse Width Modulation)**: Piny GPIO mogą pełnić funkcję wyjść PWM, co pozwala na generowanie sygnałów o zmiennym wypełnieniu, wykorzystywanych w sterowaniu silnikami, podświetleniem LED, itp.
- **ADC (Analog-to-Digital Converter)**: Piny mogą pełnić funkcję wejścia analogowego, umożliwiając pomiar napięcia zewnętrznego przy pomocy przetwornika analogowo-cyfrowego.
- **DAC (Digital-to-Analog Converter)**: Niektóre piny mogą pełnić funkcję wyjścia analogowego, umożliwiając generowanie sygnałów analogowych.
- **External Interrupts (EXTI)**: Piny GPIO mogą pełnić funkcję wyzwalania przerwań zewnętrznych, reagując na zmiany stanów logicznych.
- **TIM (Timers)**: Piny GPIO mogą być używane do generowania sygnałów zegarowych lub jako wejścia/wyjścia dla timerów, wykorzystywanych m.in. do generowania opóźnień, liczenia impulsów, generowania sygnałów PWM.