FreeRTOS 1/3

Wstęp

FreeRTOS jest system operacyjnym czasu rzeczywistego skierowanym do wykorzystania w systemach wbudowanych. W naszym przypadku implementowany będzie na mikrokontrolerze STM32 firmy STmicroelectronics. Rodzina 32-bitowych układów tego producenta zawiera w sobie zawiera serie ekonomiczną, standardową oraz wysokowydajną, nie zabrakło w niej układów typu *LowPower*, jak również układów z wbudowaną komunikacją Bluetooth i WiFi oraz dwurdzeniowych. Na zajęciach wykorzystany zostanie środowisko STM32CubeIDE które posłuży do przygotowania, programowania i debugowania płyty deweloperskiej Nucelo-64 z układem STM32F446.

Przebieg ćwiczenia

1. **Wstęp do pracy z FreeRTOS w środowisku STM32CubeIDE**
    a. Nucelo-64 - płyta ewaluacyjna
    b. STM32CubeIDE - środowisko projektowe
    c. STM32CubeMX – graficzny konfigurator mikrokontrolera
2. **Utworzenie projektu na podstawie pobranego pliku konfiguracyjnego lub domyślnej konfiguracji dla płyty Nucelo-64 z układem STM32F446**
3. **Wyjścia cyfrowe**
   Zadeklaruj funkcje której zadaniem będzie miganie diodą wbudowaną diodą LDX. Funkcja przyjmuje dwa argumenty, czas włączenia i wyłączenia diody. Funkcję wywołaj w głównej pętli programu.
4. **Wejścia cyfrowe**
   Napisz funkcje zmieniającą stan diody na przeciwny za każdym wciśnięciem przycisku. Wykonaj zadanie na przycisku wbudowanym na płytę *Nucelo* oraz na płycie rozszerzeń. Funkcję wywołaj w głównej pętli programu.
5. **Debugowanie**
   Rozbuduj funkcję z poprzedniego zadania o zapis wartości aktualnego stanu przycisku w zmiennej typu uint8_t. Uruchom program w trybie *Debug*, następnie sprawdź aktualną wartość za pomocą narzędzia *Live Expression*.
6. **Opóźnienia**
   Rozbuduj funkcje z poprzedniego zadania o każdorazową zmianę częstotliwości migania diody LDX po naciśnięciu przycisku. Częstotliwość ma zostać wybrana z następującego zbioru: 10Hz, 1Hz, 0.25Hz. Do migania wykorzystuj przygotowaną wcześniej funkcję z zadania 3. Reakcja na przycisk ma być natychmiastowa. Funkcję wywołaj w głównej pętli programu.
7. **Przerwania**
   Napisz program który w momencie pojawienia się zbocza narastającego na przycisku BX ma zmienić pin PAX na stan wysoki, w przypadku zbocza opadającego wyjście PAX ma mieć stan niski. Stan po inicjalizacji na wyjściu PAX ma być wysoki.
8. **Przetwornik analogowo-cyfrowy**
   Napisz funkcje której zadaniem jest odczyt napięcia na fotorezystorze wbudowanym w płytę rozszerzeń. Sprawdź wykorzystaną rozdzielczość przetwornika ADC a następnie zmapuj odczyty do skali 0-100%. Zwracaną wartość zapisz w zmiennej globalnej typu uint8_t i sprawdź poprawność działania w trybie *Debug*.
9. **Przetwornik analogowo-cyfrowy i wyjście cyfrowe**
   Wykorzystaj poprzednie opracowania i przygotuj program który na podstawie czujnika oświetlenia (fotorezystor) będzie dostosowywał płynnie jasność świecenia diody.

Funkcje

Funkcje do wykorzystania na zajęciach przedstawione zostały poniżej, każda z nich została szczegółowo opisana w zamieszczonej nocie katalogowej.

- void **HAL_GPIO_WritePin** (GPIO_TypeDef * GPIOx, uint16_t GPIO_Pin, GPIO_PinState PinState)
- void **HAL_GPIO_TogglePin** (GPIO_TypeDef * GPIOx, uint16_t GPIO_Pin)
- GPIO_PinState **HAL_GPIO_ReadPin** (GPIO_TypeDef * GPIOx, uint16_t GPIO_Pin)
- void **HAL_Delay** (uint32_t Delay)
- uint32_t **HAL_GetTick** (void)
- void **HAL_GPIO_EXTI_Callback** (uint16_t GPIO_Pin)
- HAL_StatusTypeDef **HAL_ADC_Start** (ADC_HandleTypeDef * hadc)
- HAL_StatusTypeDef **HAL_ADC_PollForConversion** (ADC_HandleTypeDef * hadc, uint32_t Timeout)
- uint32_t **HAL_ADC_GetValue** (ADC_HandleTypeDef * hadc)

Materiały dodatkowe

- Biblioteka HAL dla STM32F4 Description of STM32F4 HAL and low-layer drivers - User manual
- Schemat płyty rozszerzeń KA-Nucleo-Multisensor (PL) (kamamilabs.com)

STM32CubeMX