AKADEMIA GÓRNICZO-HUTNICZA

Dokumentacja do projektu

Biblioteka do obsługi sensorów i komunikacji UART z mikrokontrolerem FRDM-KL05Z

z przedmiotu

Języki programowania obiektowego

Elektronika i Telekomunikacja, semestr 5

Mirosław Baca

grupa zajęciowa - piątek 9:45

Prowadzący ćwiczenia: Jakub Zimnol

1. Opis projektu

Projekt stanowi bibliotekę komunikacyjną pomiędzy mikrokontrolerem na płytce ewaluacyjnej FRDM-KL05Z, a komputerem PC z systemem Windows. Kompilacja kodu na komputer mogłaby odbyć się również w innym systemie operacyjnym, wymagałoby to zmienienia biblioteki do obsługi UART (jest to jedyna zewnętrzna biblioteka użyta w tym projekcie). Przy użyciu komend w terminalu PC mamy możliwość przesłania ich do mikrokontrolera, który sprawdza ich poprawność i wykonuje dane polecenie. Płytka może odsyłać z powrotem na strumień UART dane z wbudowanego miernika temperatury, akcelerometru lub suwaka dotyku oraz sterować diodami RGB na podstawie otrzymanych komend.

2. Struktura projektu

W repozytorium znajdują się osobne pliki do kompilacji i wgrania do pamięci mikrokontrolera oraz te do kompilacji aplikacji terminalowej na PC.

Piki wspólne dla MCU i PC:

- **CommunicationModuleBase.hpp** – Bazowa klasa abstrakcyjna zawierająca definicje komend oraz innych części wspólnych interfejsu komunikacyjnego, z której dziedziczą osobno klasy stworzone w projektach MCU i PC.

Część dla MCU:

- **main.cpp** - Główna pętla programu na mikrokontrolerze. Inicjalizuje peryferia i obsługuje komendy otrzymane z UART odsyłając odpowiedź na UART lub wykonując zadane polecenie.
- **Uart.hpp / Uart.cpp** – Plik z klasą implementującą obsługę transmisji danych oraz obsługę odbioru danych przy pomocy przerwań na interfejsie UART.
- **BoardSupport.hpp / BoardSupport.cpp** - Moduły obsługi peryferiów (TSI, ADC, LED, I2C).
- **CommunicationModuleMCU.hpp / CommunicationModuleMCU.cpp -** Klasa implementująca obsługę komend i buforowanie danych UART na mikrokontrolerze.

Część dla MCU:

- **main.cpp** - Aplikacja terminalowa do wysyłania komend do mikrokontrolera przez UART, oraz demonstrująca funkcjonalności klasy Accelerometer.
- **AccelerometerClass.hpp / AccelerometerClass.cpp** – Klasa implementująca obiekt przechowujący dane przyspieszenia w 3 osiach oraz metody związane parsowaniem danych z akcelerometru i wyświetlaniem wyników w formacie czytelnym dla użytkownika.
- **CommunicationModulePC.hpp / CommunicationModule.cpp** - Klasa przetwarzająca komendy oraz zarządzająca komunikacją UART na komputerze.
- **SerialPort.hpp / SerialPort.cpp** – Zewnętrzna biblioteka implementująca obsługę UART w systemie Windows. Źródło https://github.com/manashmandal/SerialPort

3. Uruchomienie projektu

- **MCU:**
  Na podstawie plików z repozytorium należy stworzyć projekt w IDE kompatybilnym z FRDM-KL05Z, w tym przypadku jest to Keil uVision5, z którego pliki projektu również są w repozytorium. Następnie należy skompilować pliki oraz utworzyć obraz pamięci dla tego mikrokontrolera i wgrać go przy pomocy programatora J-Link.

- **PC:**
  Należy skompilować wszystkie pliki z folderu *PC Files*, można wykorzystać również przygotowany do tego celu plik CMake. Pliki zewnętrznej biblioteki SerialPort również są zawarte w tym folderze.
- **Testy:**
  Tak zrealizowane kroki powinny pozwolić na komunikację między tymi dwoma programami. W celach debugowania UART można skorzystać również z takich narzędzi jak Termite. Domyślna wartość BaudRate to 9600 ale moduły są tak zrealizowane, że można ją zmienić przy inicjalizacji.

4. Komendy i klasy

Dostępne komendy:

ping - Sprawdza połączenie między MCU a PC. W odpowiedzi MCU zwraca "PONG".
reset - Resetuje mikrokontroler.
readinfo - Zwraca informacje o MCU takie jak Device Name oraz UID.

readtemp - Odczytuje aktualną temperaturę z czujnika ADC i zwraca wynik w stopniach Celsjusza.

readtouch - Odczytuje wartość suwaka dotykowego i przesyła ją w skali 0-100 przez UART.
settouch - Uruchamia ponowną kalibrację modułu TSI – suwaka dotykowego.

setledcolorred - Ustawia diode RGB na kolor czerwony.
setledcolorgreen - Ustawia diode RGB na kolor zielony.
setledcolorblue - Ustawia diode RGB na kolor niebieski.

readaccel - Odczytuje surowe dane z akcelerometru (6 bajtów, po 2 na każdą oś) i transmituje po UART, komputer następnie obrabia dane i wyświetla w formacie zmiennoprzecinkowym.

Dodatkowe informacje na temat klas:

- Dziedziczenie:
  CommunicationModuleBase -> CommunicationModuleMCU I CommunicationModulePC
- Konstruktory i destruktory:
  W klasie CommunicationModulePC został dodany konstruktor domyślny oraz parametryczny, oba powodują poprawne inicjalizowanie komunikacji, oraz destruktor, który zamyka port UART.
- Przeciążenie operatorów:
  W klasie Accelerometer przeciążono operatory arytmetyczne oraz porównania.
  UWAGA: Dla porównań <, <=, >, >= wykonano porównywanie długości wektora przyspieszenia w przestrzeni trójwymiarowej ($magnituda = \sqrt{x^2 + y^2 + z^2}$)

5. Podsumowanie

Istotną i czasochłonną kwestią okazała się optymalizacja pod względem zużycia pamięci na mikrokontrolerze, gdyż jest on ograniczony pamięcią RAM (4 KB) i Flash (32 KB). Dane z akcelerometru musiały zostać przesłane w surowym formacie, gdyż typy float albo double zajmują bardzo dużo miejsca. Wszędzie, gdzie to możliwe zastosowano najkrótsze możliwe typy danych do konkretnych zadań (uint8_t, int16_t itd.), a bufory do odbioru danych zdefiniowano jako statyczne co zminimalizowało dynamiczne alokacje pamięci.