AKADEMIA GÓRNICZO-HUTNICZA

Dokumentacja do projektu

Biblioteka generyczna do protokołów komunikacyjnych

z przedmiotu

Języki Programowania Obiektowego

EiT 3 rok

Antonia Zdziebko

Środa 13:15

prowadzący: Jakub Zimnol

13.01.2026

1. Opis projektu

Projekt jest biblioteką generyczną do obsługi protokołów komunikacyjnych: UART, I2C i SPI. Została zaprojektowana tak, aby była niezależna od mikrokontrolera. Wykorzystano projektowanie obiektowe, bazując na abstrakcyjnej klasie bazowej Protocol.

Celem projektu jest wspólny interfejs dla różnych protokołów komunikacyjnych oraz oddzielenie logiki protokołów od warstwy sprzętowej (aby była ogólna i uniwersalna). Dzięki temu rozbudowa projektu jest łatwiejsza, a chęć wykorzystania biblioteki w pełnej sprawności sprzętowej można osiągnąć pisząc odpowiedni adapter do mikrokontrolera.

Działanie kodu, zostało przedstawione na mikrokontrolerze ESP32 w formie demonstracyjnej. Nie wykorzystano pełnej implementacji protokołów, lecz użyto logi pozwalające pokazać zastosowanie biblioteki. Przedstawiają dodatkowo mechanizmy kontrolne i konfiguracyjne, a także zarządzenie stanami. Zachowuje to generyczność biblioteki i pokazuje pełne możliwości po zastosowaniu warstwy sprzętowej.

2. Opis klas

Protocol:

Główną klasą abstrakcyjną jest klasa Protocol. Stanowi ona podstawę do konkretnch protokołów komunikacyjnych. Definiuje ona elementy wspólne dla protokołów takich jak UART, I2C oraz SPI, niezależnie od konkretnej implementacji sprzętowej. Zadeklarowano typy opisujące protokoły i ich stan.

Zawiera metody send i recive, które są czysto wirtualne, ponieważ są one wspólną cechą wszystkich protokołów. Te odpowiedzialne za inicjalizacje - nie zostały zawarte, ponieważ są one mocno zależne od konkretnego protokołu w razie chęci pełnego uruchomienia sprzętowego.

UART:

Klasa UART jest klasą dziedziczącą po Protocol. Jest implementacją samego protokołu UART. Zawiera pola konfiguracyjne typowe dla niej jak np. odpowiednie piny. Wykorzystuje zarządzanie swoim stanem, który został zdefiniowany w Protocol. Nadpisuje metody send i recv, tak aby spełniały założenia przesyłu ramek tak jak w protokole UART.

I2C:

Klasa I2C reprezentuje działanie protokołu komunikacyjnego I2C. Tak samo jak UART dziedzicy po Protocol i nadpisuje te same metody. Nie przesyła pierwszego bajtu danych, ponieważ jest on adresem urządzenia. Zarządza konfiguracją i pinami zgodnie z założeniami samego protokołu.

SPI:

Klasa SPI również jest klasą pochodną i realizuje protokół komunikacyjny SPI. Ma zdefiniowane pola potrzebne do przesyłu danych właśnie tym protokołem. Sprawdza poprawność konfiguracji pól i zarządza stanami obiektu.

3. Uruchomienie CMake

Projekt zawiera dwa pliki main, jeden jest przeznaczony do uruchomienia projektu na podstawowym oprogramowaniu. Natomiast drugi jest przeznaczony do uruchomienia na ESP32 za pomocą ESP-IDF.

Wersja desktop:

W katalogu /desktop znajduje sie plik “main_desktop.cpp” i odpowiedni “CMakeLists_desktop.txt”. Aby uruchomić w podstawowym oprogramowaniu należy przejść do katalogu /desktop. Następnie skompilować:

cmake -S . -B build

cmake --build build

./build/protocols_desktop

Aby uruchomić na ESP-IDF należy:

idf.py build

Idf.py flash monitor.