



**AKADEMIA GÓRNICZO-HUTNICZA**

Dokumentacja do projektu

# **Generyczna biblioteka do obsługi GPIO na mikrokontrolerze**

z przedmiotu

## **Języki Programowania Obiektowego**

Elektronika i Telekomunikacja, 3rok

*Sylwester Ślusarczyk*

Środa 8:00

prowadzący: mgr. inż. Jakub Zimnol

13.01.2026r

# 1. Opis projektu

W ramach projektu została wykonana uniwersalna biblioteka do obsługi GPIO, przeznaczona do zastosowań w systemach wbudowanych. Jej założeniem jest uproszczenie pracy z pinami wejścia i wyjścia poprzez udostępnienie jednolitego interfejsu dla rodzin mikrokontrolerów ARM oraz AVR

Biblioteka wykorzystuje podstawowe mechanizmy programowania obiektowego, co pozwala na ukrycie szczegółów sprzętowych i oddzielenie ich od kodu aplikacji. Umożliwia konfigurację pinów oraz sterowanie ich stanem logicznym w sposób czytelny i łatwy w użyciu.

## 2. Struktura i opis klas

Struktura projektu została oparta na podejściu obiektowym, zastosowano podział na klasy abstrakcyjne oraz klasy implementacyjne, odpowiadające za konkretną platformę sprzętową.

Centralnym elementem biblioteki jest klasa bazowa reprezentująca ogólną koncepcję pinu GPIO. Definiuje ona wspólny interfejs umożliwiający konfigurację pinu oraz operacje odczytu i zapisu stanu logicznego. Na jej podstawie zaimplementowane zostały klasy pochodne, które realizują rzeczywistą obsługę GPIO dla danej platformy sprzętowej. Klasy te odpowiadają za konfigurację rejestrów, ustawianie kierunku pinu oraz zmianę jego stanu logicznego.

**GPIO** – klasa abstrakcyjna opisująca pojedynczy pin GPIO. Definiuje wspólny zestaw operacji (init, ustawienie kierunku, trybu, pull-up, zapis stanu oraz odczyt), które muszą zostać zaimplementowane w klasach pochodnych.

**GPIO\_port < T >** – generyczna klasa reprezentująca port GPIO oparty o rejestry DDRx/PORTx/PINx. Odpowiada za operacje niskopoziomowe na bitach rejestrów (ustawianie kierunku, ustawianie stanu, odczyt, pull-up) oraz walidację numeru pinu zależnie od szerokości rejestru.

**GPIO\_pin<T>** – generyczna klasa reprezentująca pin GPIO, obiekt tej klasy przyjmuje w konstruktorze referencję do obiektu klasy GPIO\_port <T>. Implementuje operacje wysokopoziomowe (setDirection/setPinState/setPinMode/setPullMode/read) na konkretne operacje na rejestrach portu, a w konstruktorze sprawdza poprawność indeksu pinu.

W bibliotece zastosowano mechanizm **conceptów języka C++**, który został wykorzystany do ograniczenia typów używanych w klasach szablonowych. W tym celu zdefiniowano concept **McuType**, którego zadaniem jest weryfikacja na etapie kompilacji, czy dany typ może być użyty jako typ rejestru mikrokontrolera.

### 3. Kompilacja i uruchomienie

Projekt wykorzystuje system budowania **CMake** oraz wymaga kompilatora zgodnego ze standardem **C++20**.

Aby poprawnie zbudować projekt należy w command line w głównym katalogu wykonać następujące polecenia:

```
mkdir build
```

```
cd build
```

```
cmake ..
```

```
make
```

#### **WAŻNE:**

aby przeprowadzić poprawną kompilację oraz uruchomić testy jednostkowe należy również przeprowadzić instalację potrzebnego narzędzia, aby to zrobić wystarczy w command line wpisać:

```
sudo apt install doctest-dev
```

Po przeprowadzeniu kompilacji powstaną dwa pliki wykonywalne:

`./GPIO_Lib_Project` - jest to plik wykonywalny przykładowego działania aplikacji

`./GPIO_Lib_Project_test` – jest to plik wykonywalny przeprowadzonych testów jednostkowych