

Mikroelektronika w Technice i Medycynie  
Marzec 2023

**Instrukcja do ćwiczeń/kolokwium z mbed\_cz1**

## 1. Praca w środowisku mbed

1.1 Uruchomić na sprzęcie projekt Blinky-example według instrukcji:

<https://developer.arm.com/documentation/102497/1-5/Tutorials/Get-started-with-an-Mbed-OS-Blinky-example>

PRZYCZYM:

- Punkt 2: Wybieramy `mbed-os-example-blinky-baremetal`
- Punkt 4: Zostawiamy defaultowe opcje (project active i )
- Punkt 6:
  - wybieramy DISCO\_F429ZI (w tym i wszystkich nast. projektach)
  - nie łączymy się z targetem (opcja Connect)

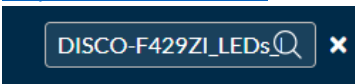
Teraz podłączamy płytke do PC

- Powinniśmy zobaczyć nowy dysk podłączony do PC (DIS\_F429ZI)
- Punkt 7: Naciskamy zawsze cały przycisk a nie opcje „clean build”
  - Jeśli proces kompilacji i linkowania przebiegł poprawnie to zostanie ze strony kompilatora przesłany plik z kodem binarnym programu, który to plik należy zapisać na dysku odpowiadającym płytce (DIS\_F429ZI)
  - Program powinien zacząć działać, tzn. powinna zacząć migać dioda na płytce.

1.2 Dla nabrania wprawy sugeruje się:

- modyfikacje powyższego programu (np. zwiększenie częstotliwości kodu)
- uruchomienie przykładu „DISCO-F429ZI\_LEDs\_Button”

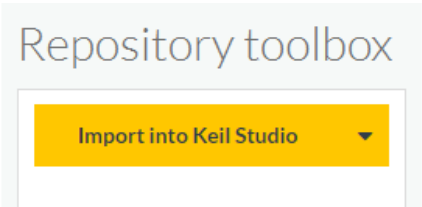
- <https://os.mbed.com/>

- 
- Did you mean: [DISCO-F429ZI\\_LEDs\\_Button](#)

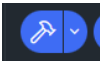
[DISCO-F429ZI\\_LEDs\\_Button - Mbed](#)

os.mbed.com › teams › code › DISCO-F429ZI\_LI

Dec 18, 2015 ... Files at revision 1:b39846a1c

Labeled [Code](#)
- 

UWAGI:

- „Build project”  dotyczy aktywnego projektu. Projekty aktywujemy z menu kontekstowego projektu, opcja „Set Active Project”
- Zdarza się, że po wgraniu pliku na dysk program nie zostanie załadowany prawidłowo (np. ekran pozostaje szary choć nie powinien). W takim przypadku należy
  - Albo nacisnąć przycisk reset i ponownie wgrać plik
  - Albo odłączyć i połączyć z powrotem płytke do PC (najlepiej od strony PC a nie płytki bo duże złącza są bardziej odporne na zużycie niż małe) i ponownie wgrać plik.

## 2. Klawiatura dotykowa

### 2a) Wersja proceduralna

#### Wymagania:

- Działanie jak odpowiedniego pliku bin znajdującym się w załącznikach na stronie www.
- Odświeżanie ekranu i odczyt panelu dotykowego równe 10 Hz
- Rozmiar "przycisków" 80x80
- Numery przycisków: "Font24"
- Kolory:
  - tło: czarny
  - przyciski:
    - ramka: zielony
    - numer tło/cyfra: czerwony/biały
    - wypełnienie wyciśnięty/wciśnięty: niebieski/zielony

#### Wskazówki:

- Sprawdzić działanie przykładowych programów, tj.:
  - „DISCO-F429ZI\_LCD\_demo”
  - „DISCO-F429ZI\_LCDTS\_demo”
- Użyć nast. metod klasy LCD\_DISCO\_F429ZI:
  - Clear, ○  
SetBackColor, ○  
SetFont
  - SetTextColor  
(uwaga: ustawia  
bieżący kolor nie  
tylko tekstu) ○  
DrawRect ○ FillRect
  - DisplayStringAt

## 2b) Wyodrębnienie klas

Cel: Wyodrębnić klasy do pracy z panelem dotykowym i wyświetlaczem.

Wymagania:

- Plik `main.c` powinien wyglądać jak poniżej.
- Implementacja metod powinna znajdować się w plikach `c`.
- Zastosować `"#include guardy"`

```
#include "mbed.h"
#include "Keyboard_Ts.h"
#include "Led_Lcd.h"

int main() {
    LedLcd Led;
    KeyboardTs Keyboard;
    while(1) {
        switch(Keyboard.eRead()) {
            case BUTTON_0:
                Led.On(0);
                break;

            case BUTTON_1:
                Led.On(1);
                break;

            case BUTTON_2:
                Led.On(2);
                break;

            case BUTTON_3:
                Led.On(3);
                break;

            default :
                Led.On(4);
                break;
        }
        wait(0.1);
    }
}
```

## 2c) Parametryzacja obiektów

Cel: Dodać do klas `LedLcd` i `KeyboardTs` możliwość ustawiania numeru kolumny.

Działanie jak odpowiedniego pliku bin znajdującym się w załącznikach na stronie [www](#).

Wymaganie: plik `main.c` powinien mieć zawartość jak w poprzednim punkcie z wyjątkiem tworzenia obiektów, które powinno wyglądać jak poniżej:

```
LedLcd Led(1);  
KeyboardTs Keyboard(2);
```

## 2d-1) Zastosowanie agregacji

Cel: Stworzyć i przetestować klasę `KeyboardTsLcd`, która będzie odpowiedzialna zarówno za obsługę dotyku jak i wyświetlanie klawiatury.

Działanie jak odpowiedniego pliku bin znajdującym się w załącznikach na stronie [www](#).

Wymagania:

- Wyodrębnić klasę `KeyboardTsLcd`, która będzie odpowiedzialna zarówno za obsługę dotyku jak i wyświetlanie klawiatury.
- `main.c` powinien wyglądać następująco:

```
int main()  
{  
    KeyboardTsLcd Keyboard(1);  
    while(1) {  
        Keyboard.eRead();  
        wait(0.1);  
    }  
}
```
- klasa `KeyboardTsLcd` powinna używać obiektów klasy `LedLcd` i `KeyboardTs` z punktu 2c), przy czym nie należy ich, tj. klas, modyfikować.
- wspomniane wyżej obiekty należy tworzyć operatorem `new`, jak poniżej:

```
KeyboardTsLcd::KeyboardTsLcd(unsigned char _ucColumn)  
{  
    pKeyboard    = new KeyboardTs(_ucColumn);  
    pLed         = new LedLcd(_ucColumn);  
};
```
- wyświetlanie klawiatury powinno się odbywać w metodzie `eRead`

## 2d-2) Wielokrotne użycie tej samej klasy

Cel: Zmodyfikować program z poprzedniego punktu tak aby klawiatura sterowała "pozycją Leda".

Działanie jak odpowiedniego pliku bin znajdującym się w załącznikach na stronie [www](#).

Ograniczenie: Nie należy tworzyć nowych klas (nie mylić z obiektami) ani modyfikować istniejących.