AKADEMIA GÓRNICZO-HUTNICZA

Dokumentacja do projektu

Odbiornik alfabetu Morse’a (z wykorzystaniem czujnika światła)

z przedmiotu

Technika mikroprocesorowa 2

Elektronika i Telekomunikacja, 3 rok

Wiktor Baran

Środa 15:00

prowadzący: dr inż. Mariusz Sokołowski

26.01.2026

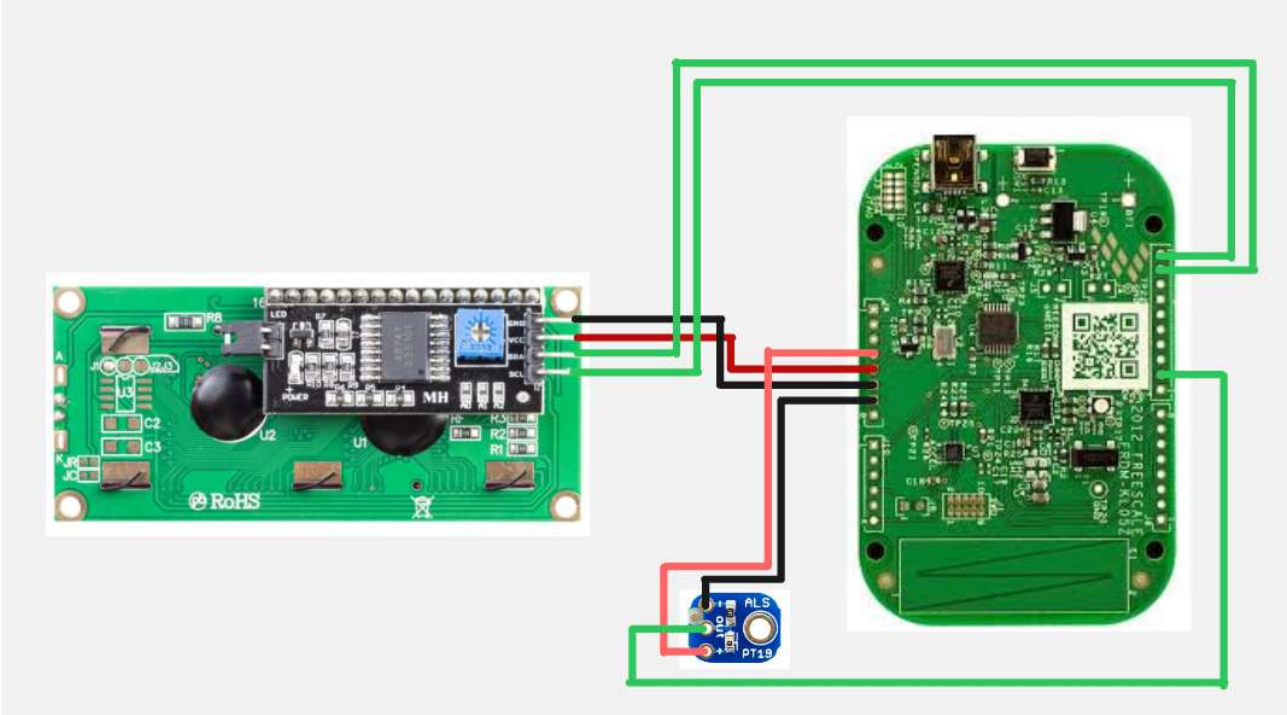
1.Cel projektu

Projekt realizuje optyczny dekoder alfabetu Morse'a na płytce rozwojowej FRDM-KL05Z. System analizuje impulsy świetlne z latarki przy użyciu czujnika światła, klasyfikując je w czasie rzeczywistym jako elementy kodu Morse'a (kropki i kreski). Zdekodowane sekwencje są tłumaczone na znaki, a następnie gromadzone i łączone w spójny wyraz. Oprogramowanie wyposażono w autokalibrację progu światła oraz obsługę błędów transmisji, co zapewnia stabilną pracę w różnych warunkach oświetleniowych. Zdekodowane wiadomości są na bieżąco prezentowane na wyświetlaczu LCD 2x16.

2.Realizacja sprzętowa

Układ został zrealizowany na platformie rozwojowej FRDM-KL05Z z mikrokontrolerem ARM Cortex-M0+. Interakcja z użytkownikiem odbywa się poprzez wyświetlacz LCD 2x16 sterowany układem HD44780U, został on podłączony do płytki poprzez 8-bitowy ekspander I2C ,dzięki któremu zredukowano potrzebną liczbę połączeń sygnałowych. Rolę interfejsu wejściowego pełni moduł analogowego czujnika światła ALS-PT19 (oparty na fototranzystorze), podłączony do wejścia przetwornika ADC mikrokontrolera. Element ten umożliwia ciągły pomiar natężenia oświetlenia, co pozwala na detekcję impulsów świetlnych ze źródła światła oraz realizację procedury autokalibracji względem tła.

Poniższy schemat ideowy ilustruje sposób integracji modułów zewnętrznych z płytką rozwojową:



3.Omówienie struktury i działania oprogramowania

Architektura aplikacji opiera się na modelu pętli nieskończonej ,w której program ciągle analizuje sygnał analogowy pochodzący z czujnika światła oraz interpretuje go jako sekwencje kodową. Działanie systemu można podzielić na trzy główne etapy:

3.1 Inicjalizacja i autokalibracja

Po uruchomieniu program konfiguruje niezbędne moduły sprzętowe, a następnie przechodzi w tryb adaptacji do otoczenia. Program dokonuje pomiaru natężenia światła tła w pomieszczeniu. Na tej podstawie wyznaczany jest dynamiczny próg detekcji (prog_tla). Dzięki temu urządzenie potrafi odróżnić celowy błysk latarki od zwykłego oświetlenia w pokoju, niezależnie od tego, czy jest dzień, czy noc.

3.2 Analiza Czasowa Sygnału

W fazie głównej program nieustannie próbkuje napięcie na czujniku. Gdy wykryty zostanie wzrost jasności powyżej ustalonego progu, system przechodzi do pomiaru czasu trwania impulsu. Pomiar czasu trwania impulsu realizowany jest poprzez licznik PIT pracujący jako stopwatch zliczający impulsy zegara co określony czas.

- Krótki impuls jest interpretowany jako **kropka**
- Długi impuls jest interpretowany jako **kreska**

Zmierzone sygnały są od razu wyświetlane na ekranie oraz gromadzone w buforze pamięci jako ciąg znaków (np. ...---...). Równocześnie mierzony jest czas przerw między błyskami - jeśli przerwa jest długa, system uznaje, że nadawanie bieżącego znaku dobiegło końca. Na zakończenie cyklu przetwarzania znaku, system wykonuje procedurę rekalibracji. Pozwala to na zaktualizowanie wartości odniesienia dla tła i zaadaptowanie urządzenia do ewentualnych zmian oświetlenia w pomieszczeniu przed nadejściem kolejnego sygnału. Dodatkowo, system realizuje cykliczną rekalibrację w stanie spoczynku. Jeżeli przez dłuższy czas nie zostanie wykryta żadna transmisja, urządzenie automatycznie aktualizuje poziom odniesienia tła, co pozwala na bieżącą kompensację zmian oświetlenia otoczenia.

3.3 Dekodowanie i Prezentacja

Program po skompletowaniu sekwencji dekoduje znak, dopisuje go na wyświetlaczu oraz wpisuje go do tablicy kompletującej wyraz. Dzięki temu na ekranie LCD sukcesywnie budowany jest ciąg znaków, umożliwiając użytkownikowi bieżący odczyt transmitowanej wiadomości. Jeżeli po wypisaniu znaku nastąpi dłuższa przerwa program zakończy odbieranie wyrazu i umożliwi odbieranie kolejnego.

4.Instrukcja obsługi

4.1 Uruchomienie

Przed podłączeniem zasilania należy ustawić czujnik światła w stabilnej pozycji, kierując go bezpośrednio w stronę planowanego źródła sygnału (latarki). Po uruchomieniu urządzenia system automatycznie zweryfikuje warunki oświetleniowe. W przypadku wykrycia zbyt silnego światła tła, wyświetlony zostanie komunikat **„Za jasno”** – należy wówczas zmienić ustawienie czujnika lub zaciemnić stanowisko. Jeżeli warunki są odpowiednie (lub po skorygowaniu pozycji), na wyświetlaczu pojawi się komunikat „**Gotowy...”**, co sygnalizuje pomyślną kalibrację i pełną gotowość systemu do odbierania sygnałów.

4.2 Nadawanie

System charakteryzuje się pełną parametryzacją kryteriów czasowych, co pozwala na dostosowanie szybkości dekodowania do preferencji użytkownika (zmiany należy dokonać w kodzie programu). Kluczowe parametry zdefiniowano jako zmienne globalne:

- **jednostkaczasu_ms** (domyślnie: 1000 ms) – określa bazowy czas trwania kropki.
- **tolerancja_ms** (domyślnie: 500 ms) – definiuje dopuszczalny margines błędu dla nadawanych sygnałów.

Algorytm automatycznie skaluje pozostałe wartości względem czasu bazowego. Czas trwania kreski jest obliczany jako trzykrotność czasu kropki (3 * jednostkaczasu_ms) z uwzględnieniem zdefiniowanej tolerancji.

W trakcie nadawania, system na bieżąco wyświetla rozpoznane elementy (kropka/kreska) w górnym wierszu wyświetlacza. Separacja poszczególnych znaków oraz wyrazów realizowana jest poprzez pomiar czasu trwania stanu niskiego (braku oświetlenia):

1. **Koniec znaku:** Przerwa trwająca dłużej niż 3 * jednostkaczasu_ms zamyka proces zbierania sekwencji (kropek/kresek) i powoduje zdekodowanie znaku.
2. **Koniec wyrazu:** Wykrycie kolejnej przerwy po zdekodowaniu znaku o długości 3 * jednostkaczasu_ms jest interpretowane jako koniec słowa, co skutkuje finalizacją wyświetlanego wyrazu i przygotowaniem bufora na nową wiadomość.

4.3 Obsługa błędów

System został wyposażony w mechanizmy ciągłego monitorowania poprawności transmisji. W przypadku wykrycia anomalii czasowych, logicznych lub środowiskowych, proces dekodowania jest przerywany, a na wyświetlaczu

prezentowany jest stosowny komunikat diagnostyczny. Po odebraniu błędów czasowych, dekodowania oraz przepełnienia możliwe jest nadawanie sygnału od razu po ujrzeniu komunikatu o błędzie.

Możliwe błędy:

- **Błędy czasowe (Err1 [czas ms] ms, Err2 [czas ms] ms):**

  Występują, gdy czas trwania impulsu świetlnego nie mieści się w zdefiniowanych oknach tolerancji dla kropki (jednostkaczasu_ms +/- tolerancja_ms) ani kreski (3*jednostkaczasu_ms +/- tolerancja_ms). W przypadku wystąpienia nic nie zostaje dodane do sekwencji odebranych znaków, użytkownik ma możliwość powtórzenia źle nadanej kreski lub kropki.

- **Błędy dekodowania (Brak znaku):** Pojawiają się, gdy odebrana sekwencja kropek i kresek jest poprawna czasowo, ale nie odpowiada żadnemu symbolowi w zaimplementowanym słowniku alfabetu Morse'a. W przypadku wystąpienia żaden znak nie zostaje dodany do budowanego słowa ,a użytkownik ma możliwość powtórnego nadania tego znaku.

- **Błąd środowiskowy (Za jasno [Napięcie] V):** Aktywowany, gdy napięcie na czujniku w stanie spoczynku przekracza próg nasycenia (2V). Błąd ten blokuje działanie programu ,aż do momentu znalezienia nowego napięcia tła ,które jest mniejsze niż 2V. Znalezione napięcie zostaje nowym progiem nasycenia.

- **Błędy przepełnienia (Za dlugi...):** System chroni bufory pamięci przed nadpisaniem w dwóch przypadkach:

  - Za dlugi kod: Próba nadania więcej niż 5 elementów (kropek/kresek) dla jednego znaku – powoduje anulowanie nadawanego znaku, użytkownik ma możliwość powtórnego nadania go.

  - Za dlugi wyraz: Przekroczenie limitu 16 znaków dla wyświetlanego słowa – powoduje natychmiastowe zakończenie słowa oraz usunięcie z wyświetlacza aktualnie uzbieranego słowa.

- **Błąd krytyczny ADC (Blad ADC):** Pojawia się natychmiast po uruchomieniu urządzenia, jeśli procedura inicjalizacji przetwornika analogowo-cyfrowego zakończy się niepowodzeniem. Jest to błąd sprzętowy, który trwale zatrzymuje pracę systemu (wymagany reset zasilania).

4.4 Dodatkowe informacje

Autokalibracja w stanie spoczynku - W stanie spoczynku (brak sygnału wysokiego) program realizuje cykliczną autokalibrację w interwałach wynoszących 3 * jednostkaczasu_ms. Mechanizm ten ma na celu bieżącą adaptację punktu pracy czujnika do zmieniających się warunków oświetleniowych otoczenia.

Zabezpieczenie Timeout (Stan wysoki > 5 * jednostkaczasu_ms): Jeżeli ciągły czas trwania impulsu świetlnego przekroczy **5 * jednostkaczasu_ms**, program uznaje to za anomalię (np. przypadkowe pozostawienie włączonego źródła światła lub zbyt mocne nasłonecznienie pomieszczenia) i wywołuje kalibracje ,która polega na ustanowieniu nowego progu napięcia tła lub zablokowaniu programu jeśli światło jest zbyt intensywne (powyżej 2V).

5.Opis funkcji

5.1 Funkcja void PIT_Init (void)

```
void PIT_Init(void) {
    SIM->SCGC6 |= SIM_SCGC6_PIT_MASK;
    PIT->MCR &= ~PIT_MCR_MDIS_MASK;
    PIT->CHANNEL[0].LDVAL = ((SystemCoreClock / 2) / 1000) - 1;
    PIT->CHANNEL[0].TCTRL |= PIT_TCTRL_TIE_MASK | PIT_TCTRL_TEN_MASK;
    NVIC_ClearPendingIRQ(PIT_IRQn);
    NVIC_EnableIRQ(PIT_IRQn);
}
```

Funkcja konfiguruje licznik okresowy PIT do pełnienia roli systemowej podstawy czasu, generującej przerwania co 1 ms. Proces inicjalizacji rozpoczyna się od włączenia sygnału taktującego dla modułu w rejestrze SIM oraz wybudzenia go ze stanu uśpienia. Następnie do rejestru LDVAL wpisywana jest wartość obliczona na podstawie częstotliwości zegara

magistrali (SystemCoreClock / 2) podzielonej przez 1000, co definiuje pożądany interwał czasowy. W końcowym etapie funkcja fizycznie uruchamia licznik, zezwala na zgłaszanie żądań przerwań oraz odblokowuje ich obsługę w kontrolerze NVIC.

5.2 Funkcja void PIT_IRQHandler (void)

```
void PIT_IRQHandler(void) {
    if (PIT->CHANNEL[0].TFLG & PIT_TFLG_TIF_MASK) {
        PIT->CHANNEL[0].TFLG = PIT_TFLG_TIF_MASK;
        timer_ms++;
    }
}
```

Procedura obsługi przerwania licznika PIT, wywoływana cyklicznie co 1 ms. Funkcja zatwierdza wystąpienie przerwania poprzez wyzerowanie odpowiedniej flagi sprzętowej oraz inkrementuje globalną zmienną timer_ms, która służy do bieżącego odmierzania czasu w systemie.

5.3 Funkcja void ADC0_IRQHandler()

```
void ADC0_IRQHandler()
{
    temp = ADC0->R[0];
    if (!wynik_ok) {
        wynik = temp;
        wynik_ok = 1;
    }

}
```

Procedura obsługi przerwania przetwornika analogowo-cyfrowego, uruchamiana automatycznie po zakończeniu każdego pomiaru. Funkcja pobiera surowy wynik konwersji z rejestru sprzętowego i przekazuje go do zmiennych programu, sygnalizując pętli głównej gotowość nowych danych o aktualnym natężeniu oświetlenia.

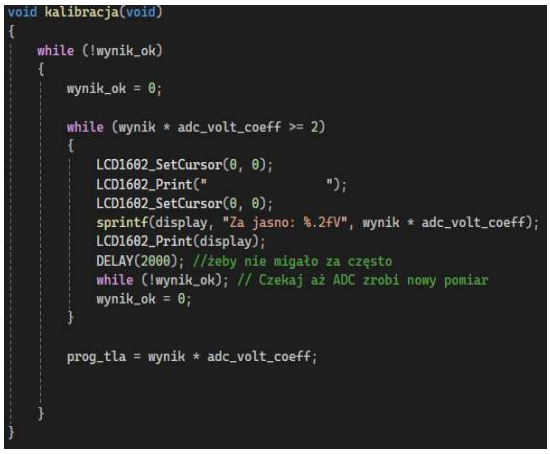
5.4 Funkcja void dekoduj_znak(char* kod)

```
char dekoduj_znak(char* kod) {
    if (strcmp(kod, ".-") == 0) return 'A';
    if (strcmp(kod, "-...") == 0) return 'B';
    if (strcmp(kod, "-.-.") == 0) return 'C';
    if (strcmp(kod, "-..") == 0) return 'D';
    if (strcmp(kod, ".") == 0) return 'E';
    if (strcmp(kod, "..-.") == 0) return 'F';
    if (strcmp(kod, "--.") == 0) return 'G';
    if (strcmp(kod, "....") == 0) return 'H';
    if (strcmp(kod, "..") == 0) return 'I';
    if (strcmp(kod, ".---") == 0) return 'J';
    if (strcmp(kod, "-.-") == 0) return 'K';
    if (strcmp(kod, ".-..") == 0) return 'L';
    if (strcmp(kod, "--") == 0) return 'M';
    if (strcmp(kod, "-.") == 0) return 'N';
    if (strcmp(kod, "---") == 0) return 'O';
    if (strcmp(kod, ".--.") == 0) return 'P';
    if (strcmp(kod, "--.-") == 0) return 'Q';
    if (strcmp(kod, ".-.") == 0) return 'R';
    if (strcmp(kod, "...") == 0) return 'S';
    if (strcmp(kod, "-") == 0) return 'T';
```

```
    if (strcmp(kod, "..-") == 0) return 'U';
    if (strcmp(kod, "...-") == 0) return 'V';
    if (strcmp(kod, ".--") == 0) return 'W';
    if (strcmp(kod, "-..-") == 0) return 'X';
    if (strcmp(kod, "-.--") == 0) return 'Y';
    if (strcmp(kod, "--..") == 0) return 'Z';


    if (strcmp(kod, ".----") == 0) return '1';
    if (strcmp(kod, "..---") == 0) return '2';
    if (strcmp(kod, "...--") == 0) return '3';
    if (strcmp(kod, "....-") == 0) return '4';
    if (strcmp(kod, ".....") == 0) return '5';
    if (strcmp(kod, "-....") == 0) return '6';
    if (strcmp(kod, "--...") == 0) return '7';
    if (strcmp(kod, "---..") == 0) return '8';
    if (strcmp(kod, "----.") == 0) return '9';
    if (strcmp(kod, "-----") == 0) return '0';

    return '?'; // Nieznany znak
}
```

Funkcja realizuje translację odebranej sekwencji kropek i kresek na znaki alfanumeryczne. Algorytm porównuje zawartość bufora wejściowego ze zdefiniowanym słownikiem kodu Morse'a i zwraca odpowiadający mu symbol ASCII lub znak zapytania, jeśli sekwencja nie została rozpoznana.

5.5 Funkcja void kalibracja(void)



```
void kalibracja(void)
{
    while (!wynik_ok)
    {
        wynik_ok = 0;

        while (wynik * adc_volt_coeff >= 2)
        {
            LCD1602_SetCursor(0, 0);
            LCD1602_Print("               ");
            LCD1602_SetCursor(0, 0);
            sprintf(display, "Za jasno: %.2fV", wynik * adc_volt_coeff);
            LCD1602_Print(display);
            DELAY(2000); //żeby nie migało za często
            while (!wynik_ok); // Czekaj aż ADC zrobi nowy pomiar
            wynik_ok = 0;
        }

        prog_tla = wynik * adc_volt_coeff;


    }
}
```

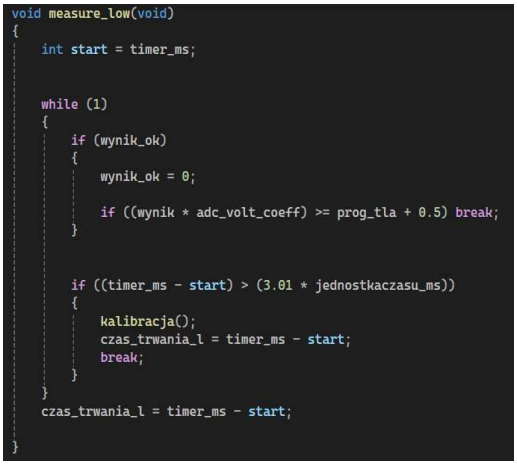
Funkcja adaptuje układ do oświetlenia otoczenia. W pierwszej kolejności weryfikuje stan nasycenia czujnika (napięcie > 2V). W przypadku przekroczenia tego progu blokuje dalsze działanie programu i wyświetla komunikat błędu, który nie znika, dopóki warunki nie ulegną poprawie. Jeżeli natomiast pomiar mieści się w normie, funkcja rejestruje aktualnie zmierzone napięcie jako zmienną prog_tla, ustanawiając tym samym nowy punkt odniesienia dla detekcji sygnałów.

5.6 Funkcja void measure_high(void)

```
void measure_high(void) {
    int start = timer_ms;


    while (1) {
        if (wynik_ok)
        {
            wynik_ok = 0;

            if ((wynik * adc_volt_coeff) < prog_tla + 0.25)
            {
                break;

            }

            if ((timer_ms - start) > 5 * jednostkaczasu_ms) {
                kalibracja();
                break;
            }

        }
    }
    czas_trwania_h = timer_ms - start;
}
```

Funkcja realizuje pomiar czasu trwania aktywnego impulsu świetlnego. Po zarejestrowaniu momentu początkowego pętla monitoruje sygnał wejściowy aż do spadku napięcia poniżej progu detekcji (z uwzględnieniem histerezy). Procedura wyposażona jest w zabezpieczenie typu timeout – jeżeli stan wysoki utrzymuje się zbyt długo (powyżej 5 jednostek czasu), system przerywa pomiar i wymusza ponowną kalibrację tła.

5.7 Funkcja void measure_low(void)



```
void measure_low(void)
{
    int start = timer_ms;


    while (1)
    {
        if (wynik_ok)
        {
            wynik_ok = 0;

            if ((wynik * adc_volt_coeff) >= prog_tla + 0.5) break;
        }


        if ((timer_ms - start) > (3.01 * jednostkaczasu_ms))
        {
            kalibracja();
            czas_trwania_l = timer_ms - start;
            break;
        }
    }
    czas_trwania_l = timer_ms - start;

}
```

Funkcja realizuje pomiar czasu trwania przerwy między impulsami (stanu niskiego). Po zarejestrowaniu momentu początkowego pętla monitoruje sygnał wejściowy aż do wzrostu napięcia powyżej progu detekcji z uwzględnieniem histerezy (sygnalizującego nadejście kolejnego błysku). Jeżeli stan niski utrzymuje się dłużej niż założony limit (ok. 3 jednostki czasu), system przerywa oczekiwanie i wywołuje kalibrację tła, adaptując się do warunków spoczynkowych.