Dokumentacja
System alarmowy

Grupa E13 – Czwartek (parzysty) 12:00

Wydział: FTIMS

Kierunek: Informatyka Stosowana

Rok akademicki: 2024/2025

Przedmiot: Systemy Wbudowane

Grupa: E13 - Czwartek (parzysty) 12:00

Członkowie zespołu:
Nikodem Nowak (lider) – 251598
Mateusz Urbaniak – 251654
Igor Kuna – 252498

Spis treści

Wykorzystywany sprzęt:

- LPCExpresso 1769,
- HC-SR04 – Ultradźwiękowy czujnik odległości
- RC522 – Moduł RFID
- CMD14 – Czujnik magnetyczny otwarcia drzwi/okien

Zakres projektu:

Lp.	Funkcjonalność	Osoby odpowiedzialne	Udział
1.	Wyświetlacz OLED	Nikodem Nowak	34%
2.	Interfejs SPI		
3.	Czytnik RFID		
4.	Interfejs I2C do PCA9532	Mateusz Urbaniak	33%
5.	PCA95322 diody LED		
6.	Czujnik natężenia światła		
7.	GPIO - joystick	Igor Kuna	33%
8.	Czujnik ultradźwiękowy odległości		
9.	Timer		
9.5.	Głośnik		

1. Instrukcja użytkownika

1.1 Użytkowanie

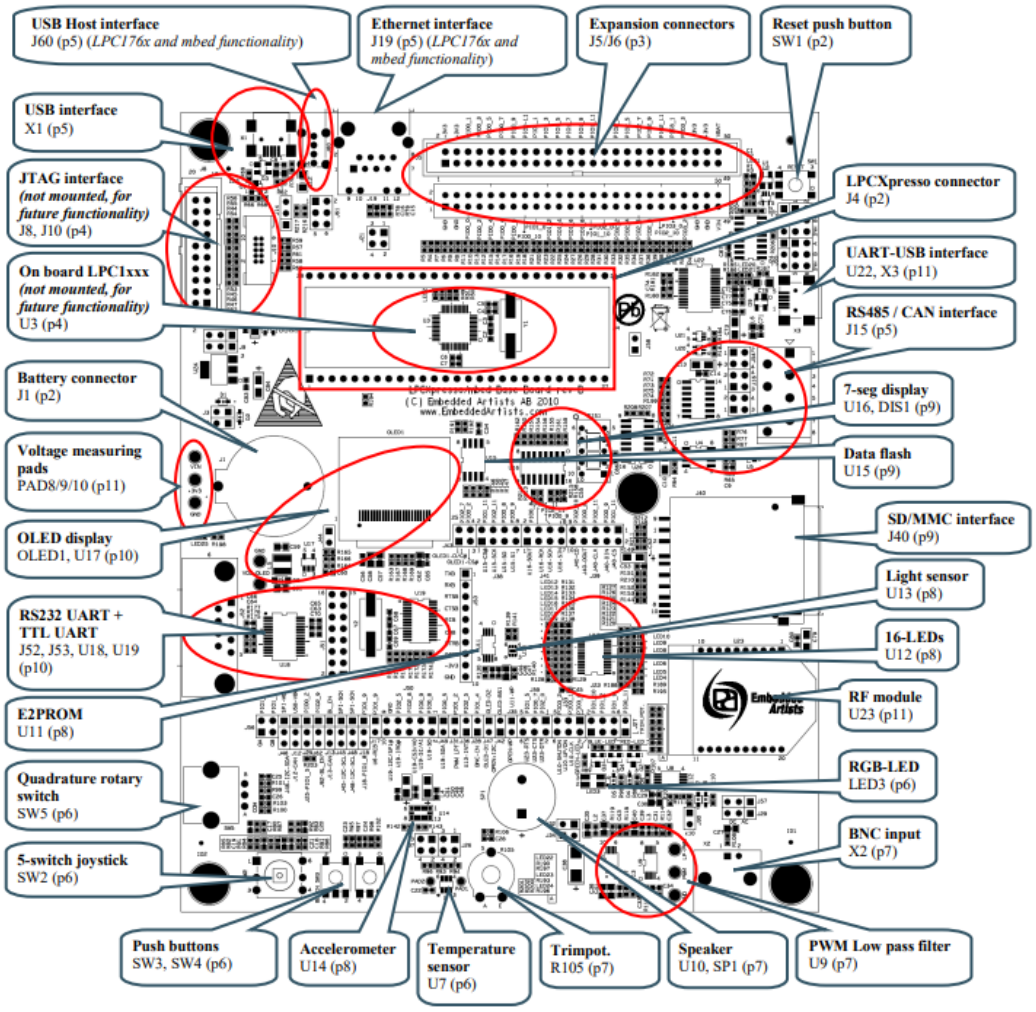
Urządzenie jest systemem alarmowym, którego stanem można sterować za pomocą joysticka i autoryzować za pomocą karty zbliżeniowej RFID.

- Stan początkowy:
  Po podłączeniu zasilania urządzenie uruchamia się, wyświetlając ekran powitalny. Po 2 sekundach przechodzi w domyślny stan pracy: system wyłączony (rozbrojony). W tym stanie urządzenie jest bezpieczne, a czujniki nie powodują aktywacji alarmu.
- Uzbrajanie systemu:
  Aby uzbroić system, należy jednokrotnie wcisnąć środek joysticka (przycisk SW2). Urządzenie wyda sygnał dźwiękowy uzbrojenia, a na ekranie pojawi się komunikat "ZABEZPIECZONY". Od tego momentu system aktywnie monitoruje czujniki.
- Rozbrajanie systemu:
  Aby rozbroić system (zarówno w stanie czuwania, jak i podczas aktywnego alarmu), należy jednokrotnie wcisnąć środek joysticka. Urządzenie wyda sygnał dźwiękowy rozbrojenia i powróci do stanu wyłączonego.
- Autoryzacja i wyciszanie alarmu:
  Gdy system jest uzbrojony i alarm zostanie wyzwolony (przez czujnik ruchu lub

otwarcie drzwi), zacznie emitować sygnał dźwiękowy i świetlny. Aby tymczasowo wyciszyć alarm i uzyskać dostęp do strefy chronionej, należy przyłożyć do czytnika autoryzowaną kartę RFID. Urządzenie wyda krótki, melodyjny sygnał dźwiękowy potwierdzający autoryzację, a następnie:

1) Natychmiastowe wyciszenie alarmu.
2) Przejście systemu w stan "AUTORYZOWANY" na okres 10 sekund.
3) Na ekranie wyświetlony zostanie licznik odliczający czas autoryzacji.

Po upływie 10 sekund system automatycznie powróci do stanu pełnego uzbrojenia i ponownie zacznie monitorować czujniki. Aby trwale rozbroić system, należy wcisnąć joystick.



Rys. 1 – Główne komponenty płyty bazowej LPCXpresso

1.2 Opis algorytmu

Program działa w oparciu o nieskończoną pętlę while(1), w której cyklicznie sprawdzane są stany wejść i czujników, a logika programu jest realizowana przez

maszynę stanów opartą na flagach globalnych. Główny cykl pętli jest synchronizowany opóźnieniem 100 ms.

1.2.1 Główna pętla i stany programu

Logika programu opiera się na trzech kluczowych zmiennych flagowych:

- isArmed (typu uint8_t): Określa, czy system jest uzbrojony (1) czy wyłączony (0).
- isAuthorized (typu uint8_t): Określa, czy użytkownik jest w 10-sekundowym oknie autoryzacji po przyłożeniu karty RFID (1) czy nie (0).
- alarmTriggered (typu uint8_t): Określa, czy alarm został wyzwolony (1) czy nie (0).

1.2.2 Obsługa zdarzeń i timerów

Program wykorzystuje nieblokujące timery programowe, zrealizowane jako liczniki inkrementowane w każdym cyklu głównej pętli (co 100 ms).

- Obsługa joysticka: W każdym cyklu pętli odczytywany jest stan joysticka. Wciśnięcie przycisku środkowego powoduje negację flagi isArmed oraz resetuje wszystkie pozostałe flagi i timery do stanu początkowego.
- Timer autoryzacji (authorizationTimer_ticks): Po poprawnej autoryzacji kartą RFID, licznik ten jest ustawiany na 100 (100 * 100 ms = 10 sekund). W każdym cyklu pętli, jeśli flaga isAuthorized jest ustawiona, licznik jest dekrementowany. Gdy osiągnie zero, flaga isAuthorized jest zerowana.
- Timer sprawdzania czujnika ruchu (motionCheckTimer_ticks): Inkrementowany co 100 ms. Gdy osiągnie wartość 2 (co 200 ms), uruchamiana jest procedura odczytu z czujnika ultradźwiękowego.
- Timer sprawdzania czujnika światła (lightCheckTimer_ticks): Inkrementowany co 100 ms. Gdy osiągnie wartość 30 (co 3 sekundy), uruchamiana jest procedura odczytu czujnika światła i ewentualnej inwersji ekranu.
- Timer odświeżania wyświetlacza (displayUpdateTimer_ticks): Inkrementowany co 100 ms. Gdy osiągnie wartość 5 (co 500 ms), lub gdy nastąpi zmiana stanu systemu, ekran jest odświeżany.

1.2.3 Logika detekcji i alarmu

Logika ta jest aktywna tylko, gdy isArmed == 1.

- Detekcja RFID: W każdym cyklu pętli sprawdzana jest obecność karty. Jeśli zostanie wykryta nowa, autoryzowana karta, ustawiana jest flaga

isAuthorized = 1, resetowana jest flaga alarmTriggered = 0, a timer autoryzacji jest ustawiany na 10 sekund.

- Detekcja naruszenia (gdy isAuthorized == 0):
  - Czujnik odległości: Co 200 ms mierzona jest odległość. Jeśli zmierzona wartość mieści się w zakresie od 5.0 cm do 200.0 cm, flaga alarmTriggered jest ustawiana na 1.
  - Kontaktron (czujnik drzwi): W każdym cyklu pętli sprawdzany jest stan pinu kontaktronu. Jeśli pin jest w stanie wysokim (drzwi otwarte), flaga alarmTriggered jest ustawiana na 1.
- Aktywacja alarmu (gdy alarmTriggered == 1 i isAuthorized == 0):
  - W głównej pętli uruchamiana jest procedura runAlarm_internal(), która generuje naprzemiennie wysoki (ok. 880 Hz) i niski (ok. 262 Hz) ton.
  - Uruchamiana jest procedura moveBar_internal(), która naprzemiennie włącza górny i dolny rząd 8 diod na wskaźniku.
  - Na wyświetlaczu pojawia się komunikat "ALARM!".

1.3 Opis wykonanego sprzętu

Na potrzeby projektu nie wykonywano żadnego własnego, dedykowanego sprzętu w postaci płytek PCB. Wszystkie komponenty zostały podłączone do gotowej płytki bazowej LPCXpresso Base Board rev. B za pomocą przewodów połączeniowych.

2. Wyświetlacz OLED

2.1 Cel

Głównym celem wykorzystania wyświetlacza OLED jest zapewnienie użytkownikowi czytelnej i bieżącej informacji o statusie alarmu. Na ekranie wyświetlane są komunikaty dotyczące:

- Aktualnego trybu pracy (zabezpieczony, wyłączony, alarm).
- Statusu autoryzacji po przyłożeniu karty RFID, wraz z czasem pozostałym na działania.
- Wykrycia naruszenia strefy przez czujnik ultradźwiękowy lub otwarcia drzwi.
- Instrukcji dla użytkownika, np. prośby o przyłożenie karty w celu dezaktywacji alarmu.

Dodatkowo, wyświetlacz automatycznie dostosowuje swój wygląd do warunków oświetleniowych, co poprawia komfort użytkowania.

2.2 Implementacja

Komunikacja z mikrokontrolerem odbywa się poprzez interfejs SSP1 (Synchronous Serial Port), skonfigurowany do pracy w trybie SPI (Serial Peripheral Interface).

Ze względu na specyfikę kontrolera SSD1305, który w trybie komunikacji szeregowej nie pozwala na odczyt danych z pamięci wideo, zaimplementowano programowy bufor ramki (ang. shadow framebuffer). Jest to tablica static uint8_t shadowFB (96 kolumn * 8 stron = 768 bajtów) w pamięci RAM mikrokontrolera, która przechowuje aktualny stan każdego piksela na ekranie. Każda operacja rysowania najpierw modyfikuje dane w buforze, a następnie odpowiedni fragment bufora jest wysyłany do wyświetlacza.

2.3 Konfiguracja

Inicjalizacja wyświetlacza obejmuje konfigurację pinów, włączenie zasilania i zegara dla peryferia SSP1 oraz ustawienie jego rejestrów sterujących.

2.3.1 Konfiguracja pinów

Funkcje pinów mikrokontrolera są ustawiane poprzez zapis do rejestrów PINSELx.

- Pin P0.9 (MOSI1): W rejestrze PINSEL0 bity 19:18 są ustawiane na wartość 0b10, co przypisuje pinowi funkcję 2, czyli MOSI1 (Master Out Slave In).
- Pin P0.7 (SCK1): W rejestrze PINSEL0 bity 15:14 są ustawiane na wartość 0b10, co przypisuje pinowi funkcję 2, czyli SCK1 (Serial Clock).

- Pin P0.6 (CS): W rejestrze PINSEL0 bity 13:12 są ustawiane na wartość 0b00, co przypisuje pinowi funkcję 0, czyli GPIO. Kierunek pinu jest ustawiany na wyjście poprzez zapis 1 do bitu 6 rejestru LPC_GPIO0->FIODIR.
- Pin P2.7 (D/C): W rejestrze PINSEL4 bity 15:14 są ustawiane na wartość 0b00, co przypisuje pinowi funkcję 0, czyli GPIO. Kierunek pinu jest ustawiany na wyjście poprzez zapis 1 do bitu 7 rejestru LPC_GPIO2->FIODIR.
- Pin P2.1 (Power): W rejestrze PINSEL4 bity 3:2 są ustawiane na wartość 0b00, co przypisuje pinowi funkcję 0, czyli GPIO. Kierunek pinu jest ustawiany na wyjście poprzez zapis 1 do bitu 1 rejestru LPC_GPIO2->FIODIR.

2.3.2 Włączenie zasilania i zegara dla SSP1

- Zasilanie modułu SSP1: W rejestrze PCONP (Power Control for Peripherals, adres 0x400FC0C4) ustawiany jest bit 10 (PCSSP1) na 1. Operacja LPC_SC->PCONP |= (1 << 10); włącza zasilanie dla interfejsu SSP1.
- Zegar modułu SSP1: W rejestrze PCLKSEL0 (Peripheral Clock Selection 0, adres 0x400FC1A8) bity 21:20 (PCLK_SSP1) są ustawiane na 0b01. Oznacza to, że zegar dostarczany do modułu SSP1 (PCLK) jest równy zegarowi procesora (CCLK), czyli 100 MHz.

2.3.3 Konfiguracja rejestrów SSP1

- Rejestr SSPCR0 (Control Register 0, adres 0x40088000):
  - Bity 3:0 (DSS): Ustawiane na 0b0111 (wartość 7), co definiuje długość ramki danych na 8 bitów.
  - Bity 5:4 (FRF): Ustawiane na 0b00, co wybiera format ramki SPI.
  - Bit 6 (CPOL): Ustawiany na 0, co oznacza, że zegar SCK1 w stanie spoczynku ma poziom niski.
  - Bit 7 (CPHA): Ustawiany na 0, co oznacza, że dane są próbkowane na pierwszej (narastającej) krawędzi zegara
- Rejestr SSPCPSR (Clock Prescaler Register, adres 0x4008 8010):
  - Bity 7:0 (CPSDVSR): Ustawiane na wartość 8. Prędkość zegara SCK1 jest obliczana ze wzoru: SCK1_Freq = PCLK / CPSDVSR. Dla PCLK = 100 MHz, prędkość transmisji wynosi 12.5 Mbit/s.
- Rejestr SSPCR1 (Control Register 1, adres 0x40088004):
  - Bit 1 (SSE): Ustawiany na 1, aby aktywować interfejs SSP1 po zakończeniu konfiguracji.
  - Bit 2 (MS): Ustawiany na 0, aby skonfigurować interfejs jako Master.

2.3.4 Sekwencja inicjalizująca kontroler SSD1305

Do kontrolera wysyłana jest seria 1-bajtowych komend. Kluczowe z nich to:

- 0xAE: Wyłączenie wyświetlacza (Display OFF) na czas konfiguracji.
- 0xA8, 0x3F: Ustawienie współczynnika multipleksowania na 64 (wartość 0x3F), co odpowiada liczbie wierszy wyświetlacza.
- 0x8D, 0x14: Włączenie wewnętrznej przetwornicy DC-DC (Charge Pump).
- 0xA1: Ustawienie mapowania segmentów (odwrócenie horyzontalne).
- 0xC8: Ustawienie kierunku skanowania wyjść COM (odwrócenie wertykalne).
- 0xA6: Ustawienie trybu normalnego (niezinwertowanego).
- 0xAF: Włączenie wyświetlacza (Display ON) po zakończeniu konfiguracji.

2.4 Zasada Działania

Operacje zapisu na ekranie są realizowane poprzez sekwencję operacji na rejestrach GPIO i SSP.

2.4.1 Wysyłanie pojedynczego bajtu (komendy lub danych):

- Aktywacja układu (Chip Select): Pin P0.6 jest ustawiany w stan niski poprzez zapis 1 do bitu 6 rejestru LPC_GPIO0->FIOCLR.
- Wybór trybu (Data/Command):
  - Dla komendy: Pin P2.7 jest ustawiany w stan niski poprzez zapis 1 do bitu 7 rejestru LPC_GPIO2->FIOCLR.
  - Dla danych: Pin P2.7 jest ustawiany w stan wysoki poprzez zapis 1 do bitu 7 rejestru LPC_GPIO2->FIOSET.
- Transmisja bajtu:
  - Wartość do wysłania jest wpisywana do rejestru danych LPC_SSP1->DR.
  - Program odczytuje w pętli rejestr statusu LPC_SSP1->SR i czeka, aż bit 4 (BSY - Busy) zostanie wyzerowany. Oznacza to, że interfejs SSP zakończył transmisję.
- Dezaktywacja układu: Pin P0.6 jest ustawiany w stan wysoki poprzez zapis 1 do bitu 6 rejestru LPC_GPIO0->FIOSET.

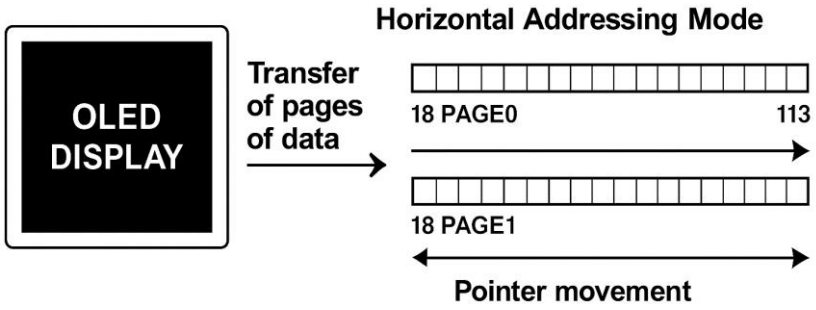
2.4.2 Tryby Adresowania Pamięci i Aktualizacja Ekranu

Kontroler SSD1305 posiada kilka trybów adresowania pamięci. Domyślnym trybem jest tryb adresowania stron (Page Addressing Mode). W tym trybie, aby zapisać pojedynczy bajt danych, należy za każdym razem wysłać komendę ustawiającą adres strony (wiersze 0-7,

8-15, itd.) oraz adres kolumny. Jest to nieefektywne przy odświeżaniu całego ekranu.
Dlatego podczas sekwencji inicjalizacyjnej, do kontrolera wysyłana jest komenda 0x20 (Set Memory Addressing Mode), a zaraz po niej bajt danych 0x00, co przełącza kontroler w tryb adresowania horyzontalnego (Horizontal Addressing Mode).
W tym trybie zachowanie wskaźnika zapisu do pamięci zmienia się fundamentalnie:

- Przed rozpoczęciem zapisu danych do bufora ramki, wskaźnik jest ustawiany na początkową pozycję za pomocą komend:
    - 0x22 (Set Page Address), z argumentami 0 (start page) i 7 (end page).
    - 0x21 (Set Column Address), z argumentami 18 (start column) i 113 (end column). Przesunięcie to wynika z faktu, że fizyczny panel ma 96 kolumn, a kontroler obsługuje do 132.
- Po zapisaniu jednego bajtu danych do pamięci wyświetlacza, wskaźnik kolumny jest automatycznie inkrementowany.
- Gdy wskaźnik osiągnie końcowy adres kolumny (113), jest on resetowany do adresu początkowego (18), a wskaźnik strony jest inkrementowany.

Proces ten jest zilustrowany na poniższym rysunku. Wskaźnik przesuwa się od lewej do prawej w obrębie jednej strony (PAGE0), a następnie "przeskakuje" na początek kolejnej strony (PAGE1) i kontynuuje zapis.



Dzięki temu, aby odświeżyć cały ekran, wystarczy jednorazowo ustawić wskaźnik na pozycję startową (strona 0, kolumna 18), a następnie

wysłać w jednej, ciągłej transmisji wszystkie 768 bajtów z programowego bufora ramki (shadowFB).

2.4.3 Automatyczna inwersja kolorów:

Funkcjonalność jest realizowana przez wysłanie pojedynczej komendy do kontrolera:

- Tryb inwersji (jasne na ciemnym): Wysyłana jest komenda 0xA7.
- Tryb normalny (ciemne na jasnym): Wysyłana jest komenda 0xA6.

Decyzja o zmianie trybu podejmowana jest na podstawie odczytu z czujnika światła i porównania z progiem DARK_THRESHOLD (30 lux).

3. Interfejs SSP (tryb SPI)

3.1 Cel

Celem implementacji magistrali komunikacji szeregowej jest zapewnienie szybkiej i niezawodnej transmisji danych pomiędzy mikrokontrolerem a dwoma kluczowymi modułami zewnętrznymi o dużej przepustowości:

- Wyświetlaczem OLED (sterownik SSD1305), który służy do wizualizacji danych i statusu systemu alarmowego.
- Czytnikiem RFID (układ MFRC522), odpowiedzialnym za proces autoryzacji użytkownika na podstawie odczytanej karty.

W projekcie wykorzystano sprzętowy moduł SSP (Synchronous Serial Port) mikrokontrolera, skonfigurowany do pracy w standardzie SPI (Serial Peripheral Interface).

3.2 Implementacja

Do obsługi komunikacji z urządzeniami peryferyjnymi wykorzystano interfejs SSP1, jeden z dwóch modułów Synchronous Serial Port dostępnych w mikrokontrolerze LPC1769.

W celu zapewnienia komunikacji z dwoma niezależnymi urządzeniami na jednej magistrali, linie wyboru układu (Chip Select, CS) oraz linia resetu (RST) są sterowane manualnie z poziomu oprogramowania za pomocą standardowych pinów GPIO.

Wykorzystywane piny w module SSP:

- SCK1 (Serial Clock) [Wejście/Wyjście] - Sygnał zegarowy modułu SSP1, synchronizujący transfer danych.
- MOSI1 (Master Out, Slave In) - Linia sygnałowa, którą moduł SSP1 (pracując jako master) wysyła dane szeregowe do urządzeń peryferyjnych.
- MISO1 (Master In, Slave Out) - Linia sygnałowa, którą moduł SSP1 (pracując jako master) odbiera dane szeregowe od urządzeń peryferyjnych.

Implementacja sterowania liniami CS/RST poprzez GPIO:

- CS_OLED (P0.6) - Pin GPIO skonfigurowany jako wyjście. Ustawienie go w stan niski aktywuje komunikację z wyświetlaczem OLED.
- CS_RFID (P2.2) - Pin GPIO skonfigurowany jako wyjście. Ustawienie go w stan niski aktywuje komunikację z czytnikiem RFID.
- RST_RFID (P2.3) - Pin GPIO skonfigurowany jako wyjście, używany do sprzętowego resetowania modułu RFID RC522.

3.3 Rejestry

Moduł SSP1 posiada 10 podstawowych rejestrów. Poniżej opisano te, które są kluczowe dla realizacji komunikacji w projekcie. Adres bazowy dla peryferium SSP1 to 0x40030000.

- SSP1CR0 - Rejestr Kontrolny 0 (adres: 0x4003 0000)
  Główny rejestr konfiguracyjny, określający podstawowe parametry transmisji.
  - Bity 3:0 – DSS (Data Size Select) - Określa liczbę bitów w ramce w następujący sposób:

0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
4	5	6	7	8	9	10	11	12	13	14	15	16

  - Bity 5:4 – FRF (Frame Format) - Wybiera format ramki (0b00 dla SPI).
  - Bit 6 – CPOL (Clock Polarity) - Polaryzacja zegara w stanie spoczynkowym.
  - Bit 7 – CPHA (Clock Phase) - Faza zegara, określająca zbocze próbkowania danych.
  - Bity 15:8 – SCR (Serial Clock Rate) - 8-bitowy dzielnik zegara (PCLK / (SCR+1)).
  - Bity 31:16 – Zarezerwowane.

- SSP1CR1 - Rejestr Kontrolny 1 (adres: 0x4003 0004)
  Rejestr kontrolujący tryb pracy i aktywację modułu.
  - Bit 0 – LBM (Loopback Mode) - Włącza (1) tryb pętli zwrotnej.
  - Bit 1 – SSE (SSP Enable) - Włącza (1) lub wyłącza (0) moduł SSP.
  - Bit 2 – MS (Master/Slave Mode) - Wybiera tryb pracy (0 dla Master, 1 dla Slave).
  - Bit 3 – SOD (Slave Output Disable) - W trybie Slave wyłącza linię MISO.
  - Bity 31:4 – Zarezerwowane.
- SSP1DR - Rejestr Danych (adres: 0x4003 0008)
  Rejestr ten służy do zapisu i odczytu danych. Zapis powoduje umieszczenie danych w kolejce nadawczej (TX FIFO), a odczyt pobiera dane z kolejki odbiorczej (RX FIFO).
  - Bity 15:0 – DATA - Pole danych do zapisu lub odczytu.
  - Bity 31:16 – Zarezerwowane.
- SSP1SR - Rejestr Statusu (adres: 0x4003 000C)
  Rejestr tylko do odczytu, zawierający flagi stanu modułu.
  - Bit 0 – TFE (Transit FIFO Empty) - 1 = Kolejka nadawcza jest pusta.
  - Bit 1 – TNF (Transmit FIFO Not Full) - 1 = Kolejka nadawcza nie jest pełna.

    - Bit 2 – RNE (Receive FIFO Not Empty) - 1 = Kolejka odbiorcza nie jest pusta.
    - Bit 3 – RFF (Receive FIFO Full) - 1 = Kolejka odbiorcza jest pełna.
    - Bit 4 – BSY (Busy) - 1 = Moduł jest zajęty transmisją.
    - Bity 31:5 – Zarezerwowane.
- SSP1CPSR - Rejestr Dzielnika Zegara (adres: 0x4003 0010) Rejestr przechowujący główny dzielnik częstotliwości zegara.
    - Bity 7:0 – CPSDVSR – Wartość preskalera (od 2 do 254, musi być parzysta).
    - Bity 31:8 – Zarezerwowane.
- SSP1IMSC – Rejestr Maski Przerwań (adres: 0x4003 0014) Rejestr do włączania i wyłączania poszczególnych źródeł przerwań.
    - Bit 0 – RORIM - Włącza (1) przerwanie od przepełnienia bufora odbiorczego (Receive Overrun).
    - Bit 1 – RTIM - Włącza (1) przerwanie od przekroczenia czasu odbioru (Receive Timeout).
    - Bit 2 – RXIM - Włącza (1) przerwanie, gdy bufor odbiorczy jest co najmniej w połowie pełny.
    - Bit 3 – TXIM - Włącza (1) przerwanie, gdy bufor nadawczy jest co najmniej w połowie pusty.
    - Bity 31:4 – Zarezerwowane.
- SSP1RIS - Rejestr Surowego Stanu Przerwań (adres: 0x4003 0018) Rejestr odzwierciedla stan przerwań przed ich zamaskowaniem przez rejestr IMSC.
    - Bit 0 – RORRIS - Surowy stan przerwania od przepełnienia bufora odbiorczego.
    - Bit 1 – RTRIS - Surowy stan przerwania od przekroczenia czasu odbioru.
    - Bit 2 – RXRIS - Surowy stan przerwania od zapełnienia bufora odbiorczego.
    - Bit 3 – TXRIS - Surowy stan przerwania od opróżnienia bufora nadawczego.
    - Bity 31:4 – Zarezerwowane.
- SSP1MIS - Rejestr Zamaskowanego Stanu Przerwań (adres: 0x4003 001C) Rejestr zawiera stan przerwań, które są jednocześnie aktywne (w RIS) i włączone (w IMSC).
    - Bit 0 – RORMIS - Zamaskowany stan przerwania od przepełnienia bufora odbiorczego.
    - Bit 1 – RTMIS - Zamaskowany stan przerwania od przekroczenia czasu odbioru.
    - Bit 2 – RXMIS - Zamaskowany stan przerwania od zapełnienia bufora odbiorczego.

    - Bit 3 – TXMIS - Zamaskowany stan przerwania od opróżnienia bufora nadawczego.
    - Bity 31:4 – Zarezerwowane.
- SSP1ICR - Rejestr Kasowania Przerwań (adres: 0x4003 0020) Rejestr tylko do zapisu, używany do kasowania flag przerwań.
    - Bit 0 – RORIC - Zapis 1 kasuje przerwanie od przepełnienia bufora odbiorczego.
    - Bit 1 – RTIC - Zapis 1 kasuje przerwanie od przekroczenia czasu odbioru.
    - Bity 31:2 – Zarezerwowane.
- SSP1DMACR - Rejestr Kontroli DMA (adres: 0x4003 0024) Rejestr włączający obsługę transferu danych przez kontroler DMA.
    - Bit 0 – RXDMAE - Włącza (1) tryb DMA dla odbiornika.
    - Bit 1 – TXDMAE - Włącza (1) tryb DMA dla nadajnika.
    - Bity 31:2 – Zarezerwowane.

3.4 Konfiguracja

Konfiguracja interfejsu SSP1 w celu realizacji komunikacji SPI jest procesem wieloetapowym, obejmującym ustawienia systemowe mikrokontrolera oraz rejestry samego modułu.

Krok 1: Włączenie zasilania i zegara peryferyjnego

Zasilanie modułu: W rejestrze PCONP (Power Control for Peripherals, adres 0x400FC0C4) ustawiany jest bit 10 (PCSSP1) na 1, co zapewnia zasilanie dla interfejsu SSP1.

Zegar modułu: W rejestrze PCLKSEL0 (Peripheral Clock Selection 0, adres 0x400FC1A8) bity 21:20 (PCLK_SSP1) określają dzielnik zegara systemowego (CCLK). Po resecie, bity te mają wartość 0b00, co ustawia zegar peryferyjny na CCLK / 4. Przy zegarze systemowym 100 MHz, zegar dostarczany do modułu SSP1 (PCLK) wynosi 25 MHz. Ta wartość jest zachowana w projekcie.

Krok 2: Konfiguracja pinów (Pin Muxing)

Pin P0.7 (SCK1) jest przypisywany do funkcji SSP1 poprzez ustawienie bitów 15:14 w rejestrze PINSEL0 (adres 0x4002C000) na wartość 0b10.

Pin P0.8 (MISO1) jest przypisywany do funkcji SSP1 poprzez ustawienie bitów 17:16 w rejestrze PINSEL0 na wartość 0b10.

Pin P0.9 (MOSI1) jest przypisywany do funkcji SSP1 poprzez ustawienie bitów 19:18 w rejestrze PINSEL0 na wartość 0b10.

Pin P0.6 (CS_OLED) pozostaje w domyślnej funkcji GPIO poprzez ustawienie bitów 13:12 w rejestrze PINSEL0 na wartość 0b00.

Pin P2.2 (CS_RFID) pozostaje w domyślnej funkcji GPIO poprzez ustawienie bitów 5:4 w rejestrze PINSEL4 (adres 0x4002C010) na wartość 0b00.

Pin P2.3 (RST_RFID) pozostaje w domyślnej funkcji GPIO poprzez ustawienie bitów 7:6 w rejestrze PINSEL4 na wartość 0b00.

Krok 3: Ustawienie parametrów pracy modułu SSP1

Konfiguracja wewnętrznych parametrów modułu odbywa się poprzez bezpośredni zapis do rejestrów kontrolnych.

W rejestrze SSP1CR0 (adres 0x4003 0000) ustawiane są: format ramki SPI (FRF = 0b00), rozmiar danych 8 bitów (DSS = 0b0111) oraz tryb zegara 0 (CPOL = 0, CPHA = 0).

W rejestrze SSP1CR1 (adres 0x4003 0004) ustawiany jest tryb Master (MS = 0).

Do rejestru SSP1CPSR (adres 0x4003 0010) wpisywana jest wartość 2. Przy zegarze PCLK o częstotliwości 25 MHz, prędkość transmisji SCK1 wynosi: 25 MHz / 2 = 12.5 Mbit/s.

Krok 4: Aktywacja modułu

Ostatnim krokiem jest finalne uruchomienie modułu SSP1. Ustawiany jest bit SSE (SSP Enable - bit 1) w rejestrze SSP1CR1, co w pełni aktywuje interfejs i przygotowuje go do transmisji danych.

4. Czytnik RFID

4.1 Cel

Głównym celem wykorzystania czytnika RFID w projekcie jest realizacja funkcji autoryzacji. System w stanie zabezpieczonym, po wykryciu naruszenia strefy, oczekuje na przyłożenie uprawnionej karty. Odczytanie poprawnego, zapisanego w programie unikalnego identyfikatora (UID) karty powoduje czasowe wyciszenie alarmu i przełączenie systemu w stan autoryzacji, dając użytkownikowi czas na interakcję z systemem.

4.2 Implementacja

Czytnik MFRC522 jest podłączony do mikrokontrolera LPC1769 poprzez interfejs SSP1. Dzieli on tę samą magistralę z wyświetlaczem OLED, a wybór aktywnego urządzenia odbywa się za pomocą dedykowanych linii Chip Select (CS).

4.3 Konfiguracja

Inicjalizacja czytnika RFID obejmuje konfigurację pinów GPIO, konfigurację współdzielonego interfejsu SSP1 oraz zapis sekwencji startowej do wewnętrznych rejestrów układu MFRC522.

4.3.1 Konfiguracja pinów

Funkcje pinów mikrokontrolera są ustawiane poprzez zapis do rejestrów PINSELx.

- Pin P2.2 (CS): W kodzie zdefiniowany jako RC522_CS_PIN. W rejestrze PINSEL4 bity 5:4 są ustawiane na wartość 0b00, co przypisuje pinowi funkcję 0, czyli GPIO. Kierunek pinu jest ustawiany na wyjście poprzez zapis 1 do bitu 2 rejestru LPC_GPIO2->FIODIR.
- Pin P2.3 (RST): W kodzie zdefiniowany jako RC522_RST_PIN. W rejestrze PINSEL4 bity 7:6 są ustawiane na wartość 0b00, co przypisuje pinowi funkcję 0, czyli GPIO. Kierunek pinu jest ustawiany na wyjście poprzez zapis 1 do bitu 3 rejestru LPC_GPIO2->FIODIR.

4.3.2 Konfiguracja interfejsu SSP1

Konfiguracja interfejsu SSP1 jest współdzielona z wyświetlaczem OLED i została szczegółowo opisana w punkcie 2.3. Kluczowe parametry to:

- Tryb pracy: SPI Master.
- Prędkość transmisji: 12.5 Mbit/s.
- Format ramki: 8-bitowa, CPOL=0, CPHA=0.

4.3.3 Konfiguracja wewnętrznych rejestrów MFRC522

Po wykonaniu resetu sprzętowego (stan niski, a następnie wysoki na pinie RST), do rejestrów konfiguracyjnych MFRC522 wpisywane są następujące wartości:

- Rejestr CommandReg (adres 0x01): Wpisywana jest wartość 0x0F (PCD_RESETPHASE). Powoduje to wykonanie programowego resetu układu, który zeruje wszystkie rejestry do wartości domyślnych.
- Rejestr TModeReg (adres 0x2A): Wpisywana jest wartość 0x8D.
    - Bit 7 (TAuto)=1: Timer wewnętrzny uruchamia się automatycznie po zakończeniu transmisji danych. Jest to kluczowe dla wykrywania braku odpowiedzi (timeout) od karty.
    - Bity 3:0 (TPrescaler_Hi)=1101b (0xD): Ustawienie 4 starszych bitów preskalera timera.
- Rejestr TPrescalerReg (adres 0x2B): Wpisywana jest wartość 0x3E. Razem z poprzednim rejestrem tworzy to 12-bitową wartość preskalera 0xD3E (3390).
- Rejestry TReloadRegH i TReloadRegL (adresy 0x2D, 0x2C): Wpisywane są wartości 0x00 i 30 (decymalnie). Ustawia to 16-bitową wartość przeładowania timera na 30. Czas timeoutu obliczany jest ze wzoru: (Preskaler * (2 * WartośćPrzeładowania + 1)) / 13.56 MHz. Daje to (3390 * (2 * 30 + 1)) / 13.56e6 Hz ≈ 15.2 ms.
- Rejestr TxAutoReg (adres 0x15): Wpisywana jest wartość 0x40. Bit 6 (Force100ASK) ustawiony na 1 włącza 100% modulację ASK (Amplitude-Shift Keying), która jest wymagana do komunikacji z kartami MIFARE.
- Rejestr ModeReg (adres 0x11): Wpisywana jest wartość 0x3D. Bity 7:6 (CRCPreset) ustawione na 0b01 wybierają wartość początkową dla kalkulatora CRC jako 0x6363, co jest standardem dla protokołu ISO/IEC 14443A.
- Rejestr TxControlReg (adres 0x14): Bity 1:0 (Tx1RFEn, Tx2RFEn) są ustawiane na 1. Włącza to sterowniki obu pinów anteny, co aktywuje pole RF.

4.4 Zasada Działania

Komunikacja z czytnikiem oraz kartą RFID odbywa się poprzez precyzyjnie zdefiniowane sekwencje odczytów i zapisów do rejestrów MFRC522.

4.4.1 Zapis i Odczyt z rejestrów MFRC522

Transmisja SPI do układu MFRC522 składa się z dwóch bajtów.

- Zapis do rejestru:

    - Pin CS (P2.2) jest ustawiany w stan niski.
    - Przez SSP1 wysyłany jest pierwszy bajt – bajt adresu. Jest on tworzony przez operację (adres_rejestru << 1) & 0x7E. Bit 7 (MSB) jest ustawiony na 0, co oznacza operację zapisu. Bit 0 (LSB) jest zerowany.
    - Przez SSP1 wysyłany jest drugi bajt – dana do zapisu.
    - Pin CS jest ustawiany w stan wysoki.
- Odczyt z rejestru:
    - Pin CS (P2.2) jest ustawiany w stan niski.
    - Przez SSP1 wysyłany jest pierwszy bajt – bajt adresu. Jest on tworzony przez operację ((adres_rejestru << 1) & 0x7E) | 0x80. Bit 7 (MSB) jest ustawiony na 1, co oznacza operację odczytu.
    - Przez SSP1 wysyłany jest dowolny drugi bajt (np. 0x00), aby wygenerować 8 cykli zegara potrzebnych do odczytu. Odebrana w tym czasie wartość jest wartością z danego rejestru.
    - Pin CS jest ustawiany w stan wysoki.

4.4.2 Proces komunikacji z kartą (detekcja i odczyt UID)

- Wysłanie zapytania (Request):
    - Do rejestru BitFramingReg (adres 0x0D) wpisywana jest wartość 0x07, aby ostatnie 7 bitów nie było traktowane jako pełny bajt.
    - Do 64-bajtowego bufora FIFO, poprzez zapis do rejestru FIFODataReg (adres 0x09), wpisywana jest komenda REQA (0x26).
    - Do rejestru CommandReg (adres 0x01) wpisywana jest komenda Transceive (0x0C). Układ MFRC522 wysyła zawartość FIFO (komendę REQA) w pole RF i przełącza się w tryb odbioru.
- Oczekiwanie na odpowiedź:
    - Program w pętli do...while odpytuje rejestr ComIrqReg (adres 0x04), czekając na zdarzenie kończące komunikację. Pętla jest przerywana, gdy wystąpi jedno z poniższych zdarzeń:
        - Ustawienie bitu 0 (TimerIRq): Minął czas oczekiwania na odpowiedź (timeout), zdefiniowany w rejestrach TReloadReg.
        - Ustawienie bitu 4 (IdleIRq): Transmisja została zakończona i układ przeszedł w stan bezczynności.
        - Ustawienie bitu 5 (RxIRq): Zakończono odbiór danych od karty.
    - Po wyjściu z pętli sprawdzany jest rejestr ErrorReg (adres 0x06) w celu weryfikacji, czy podczas transmisji nie wystąpiły błędy (np. kolizja, błąd parzystości, błąd bufora).

    - Jeśli nie wystąpiły błędy, a karta odpowiedziała, układ odbiera jej odpowiedź (np. ATQA) i umieszcza ją w buforze FIFO.
- Procedura antykolizyjna:
    - Do bufora FIFO wpisywana jest komenda ANTICOLLISION (0x93) oraz SEL_CASCADE_LEVEL_1 (0x20).
    - Ponownie uruchamiana jest komenda Transceive.
    - Karta odpowiada swoim 4-bajtowym numerem seryjnym (UID) oraz 1-bajtową sumą kontrolną (BCC). Dane te są umieszczane w buforze FIFO.
- Odczyt UID i weryfikacja:
    - Program odczytuje 5 bajtów z rejestru FIFODataReg.
    - Obliczana jest suma kontrolna XOR z pierwszych 4 odczytanych bajtów i porównywana z piątym odczytanym bajtem (BCC).
    - Jeśli suma się zgadza, odczytane 4 bajty UID są porównywane z autoryzowanym numerem zapisanym w pamięci programu.
- Zakończenie komunikacji:
    - Do bufora FIFO wpisywana jest komenda HALT (0x50) wraz z obliczonym CRC.
    - Uruchamiana jest komenda Transceive, która usypia kartę, aby nie odpowiadała na kolejne zapytania.

5. Interfejs I2C

5.1 Cel

Magistrala I2C (Inter-Integrated Circuit) jest kluczowym interfejsem komunikacyjnym w projekcie, służącym do obsługi dwóch zintegrowanych na płytce peryferiów:

- Cyfrowego czujnika natężenia oświetlenia ISL29003, który dostarcza danych do funkcji automatycznej inwersji kolorów wyświetlacza.
- 16-bitowego ekspandera LED PCA9532, który odpowiada za wizualną sygnalizację stanu systemu alarmowego.

W obu przypadkach mikrokontroler LPC1769 pełni rolę urządzenia nadrzędnego (Master), które inicjuję cała transmisję, wysyłając komendy i żądania odczytu/zapisu do urządzeń podrzędnych (Slave).

5.2 Implementacja

W projekcie wykorzystano sprzętowy moduł I2C2, jeden z trzech dostępnych w mikrokontrolerze LPC1769. Komunikacja odbywa się z wykorzystaniem dwóch lini:

- P0.10 jako linia danych SDA (Serial Data)
- P0.11 jako linia zegarowa SCL (Serial Clock)

Wszystkie transakcje na magistrali są realizowane w trybie blokującym (polling), co oznacza, że procesor aktywnie oczekuje na zakończenie każdej operacji (np. wysłanie bajtu), sprawdzając flagi statusu w pętli.

5.3 Rejestry

Obsługa magistrali I2C na poziomie sprzętowym wymaga konfiguracji kilku kluczowych rejestrów peryferium:

- I2CON – główny rejestr kontrolny magistrali. Operacji na nim odbywają się poprzez dwa pomocnicze rejestry:
  - I2CONSET (adres dla I2C2: 0x400A0000): Wpisanie 1 na dany bit w tym rejestrze ustawia odpowiadający mu bit w I2CON.
  - I2CONCLR (adres dla I2C2: 0x400A0018): Wpisanie 1 na dany bit w tym rejestrze zeruje odpowiadający mu bit w I2CON.
  - Najważniejsze bity I2CON używane w projekcie:
    - Bit 7: Zarezerwowany. Powinien być zapisany jako 0.
    - Bit 6 (I2EN – I2C Interface Enable): Włącza (1) lub wyłącza (0) moduł I2C.
    - Bit 5 (STA – Start Flag): Ustawienie tego bitu powoduje wygenerowanie warunku START (lub REPEATED START) na magistrali. Bit jest zerowy sprzętowo po wysłaniu warunku.
    - Bit 4 (STO – Stop Flag): Ustawienie tego bitu generuje warunek STOP.

        - Bit 3 (SI – Serial Interrupt Flag): Flaga ustawiona sprzętowo po zakończeniu każdej operacji (np. wysłanie START, adresu, danych). Dalsza transmisja jest wstrzymana do momentu programowego skasowania tej flagi.
        - Bit 2 (AA – Assert Acknowledge Flag): Decyduje o sygnale potwierdzenia. Ustawienie na 1 powoduje wysłanie ACK po odebraniu danych, 0 powoduje wysłanie NACK.
        - Bit 1: Zarezerwowany. Powinien być zapisany jako 0.
        - Bit 0: Zarezerwowany. Powinien być zapisany jako 0.
    - I2STAT (adres dla I2C2: 0x400A0004) – Rejestr statusu. Zawiera 5-bitowy kod, który precyzyjnie określa aktualny stan magistrali I2C i jej wynik ostatniej operacji (np. 0x08 – START wysłany)
    - I2DAT (adres dla I2C2: 0x400A0008) – 8-bitowy rejestr danych. Służy zarówno do zapisywania danych do wysłania, jak i odczytywania danych odebranych z magistrali.
    - I2SCLH i I2SCLL (adresy dla I2C2: 0x400A0010, 0x400A0012) – rejestry dzielnika zegara. Określają czas trwania stanu wysokiego (I2SCLH) i niskiego (I2SCLL) na linii SCL, co bezpośrednio definuje prędkość transmisji.

5.4 Konfiguracja

Konfiguracja magistrali I2C odbywa się w funkcji init_i2c() i jest realizowana przez wywołanie funkcji z biblioteki lpc17xx_i2c.c. Proces ten sprowadza się do następujących kroków na poziomie rejestrów:

1. Konfiguracja pinów
    - W rejestrze PINSEL0 dla pinów P0.10 i P0.11 ustawiana jest funkcja alternatywna nr 2 (0b10), która przypisuje je do obsługi przez moduł I2C2.
2. Włączenie zasilania i zegara dla modułu I2C2
    - W rejestrze PCONP (Power Control for Peripherals) ustawiany jest bit 26 (PCI2C2), co włącza zasilanie dla modułu I2C2.
    - W rejestrze PCLKSEL1 konfigurowany jest dzielnik zegara dla I2C2. Domyślnie jest to CCLK/2.
3. Ustawienie prędkości magistrali
    - Funkcja I2C_Init() wywołuje wewnętrzną funkcję I2C_SetClock(LPC_I2C2, 100000), która oblicza odpowiednie wartości dla rejestrów I2SCLH i I2SCLL, aby uzyskać docelową prędkość 100 kHz.
      Wzór na częstotliwość: BitFrequency = PCLK / (I2SCLH + I2SCLL)
4. Włączenie interfejsu I2C
    - Funkcja I2C_Cmd(LPC_I2C2, ENABLE) ustawia bit I2EN (bit 6) w rejestrze I2CON poprzez zapis do I2CONSET. Od tego momentu moduł I2C jest aktywny i gotowy do pracy.

5.5 Komunikacja

W projekcie komunikacja ze wszystkimi urządzeniami na magistrali I2C2 odbywa się poprzez funkcje I2CWrite i I2CRead, które są opakowaniem dla I2C_MasterTransferData.

Każda transakcja na magistrali I2C jest inicjowana przez urządzenie nadrzędne (Master), którym w naszym projekcie jest mikrokontroller LPC1769.

5.5.1 Ogólny Schemat Transmisji

1. Rozpczęcie transmisji: Aby przejąć kontrolę nad magistralą. Master generuje warunek START poprzez ustawienie bitu STA w rejestrze I2CON. Po tej operacji Master oczekuje na ustawienie flagi SI, co sygnalizuje gotowść do następnego kroku, a w rejestrze I2STAT pojawia się odpowiedni kod statusu.
2. Wysyłanie danych: Master wysyła dane (adres urządzenia lub bajt danych), wpisując je do rejestru danych I2DAT. Następnie programowo zeruje flagę SI i czeka na jej ponowne ustawienie przez sprzęt, co oznacza, że bajt został wysłany, a urządzenie podrzędne (Slave) odpowiedziało stygnałem potwierdzenia ACK. Każdy krok jest weryfikowany przez odczytanie kodu z rejestru I2STAT.
3. Odbieranie danych: Master inicjuje odbiór, zerując flagę SI. Po odebraniu bajtu przez sprzęt i umieszczeniu go w I2DAT, flaga SI jest ponownie ustawiana. Master odczytuje dane z I2DAT. Przed odebraniem ostatniego bajtu, Master musi wyzerować bit AA w I2CON, aby po jego odebraniu wysłac sygnał NACK i poinformować Slave'a o zakończeniu odczytu.
4. Zakończenie transmisji: Master kończy komunikację, generując warunek STOP poprzez ustawienie bitu STO w rejestrze I2CON

5.5.2 Przykłady komunikacji w projekcie

Poniżej przedstawiono, jak ten schemat realizujemy dla konkretnch urządzeń w naszym projekcie.

Komunikacja z czujnikiem światła polegająca na odczycie jego rejestrów wewnętrznych. Jest to transakcja typu Zapis-Odczyt (Write-Read):

1. START: Mikrokontroler generuje warunek START.
2. ZAPIS:
   - Mikrokontroler ponownie wysyła adres czujnika (0x44), z bitem R/W ustawionym nam 0 (zapis)
   - Po otrzymaniu ACK, wysyła 1-bajtowy adres wewnętrznego rejestru czujnika, z którego chce odczytać dane.
3. REPEATED START: Mikrokontroler generuje powtórzony warunek START, aby zmienić kierunek transmisji bez zwalniania magistrali.
4. ODCZYT:

- Mikrokontroler ponownie wysyuła adres czujnika (0x44), tym razem z bitem R/W ustawionym na 1 (odczyt).
- Po otrzymaniu ACK, odczytuje 1 bajt z danych czujnika i wysła potwierdzenie NACK, sygnalizując koniec odczytu.

5. STOP: Mikrokontroler kończy transmisję, generując warunek STOP.

Komunikacja z ekspanderem LED polega na zapisie danych do jego rejestrów sterujących. Jest to transakcja typu Zapis (Write):

1. START: Mikrokontroler generuje warunek START.

2. ZAPIS:

- Mikrokontroler wysyła 7-bitowy adres ekspendera (0x60) z bitem R/W ustawionym nam 0 (zapis).
- Po otrzymaniu ACK wysyła ciąg bajtów:
  - Pierwszy bajt to adres początkowego rejestru do zapisu
  - Kolejne bajty to dane do wpisywania do kolejnych rejestrów (LS0, LS1, LS2, LS3). Dzięki trybowi Auto-Increment, nie ma potrzeby wysyłania adresu każdego rejestru osobno.

3. STOP: Po wysłaniu wszystkich danych, mikrokontroler generuje warunek STOP, kończąc transakcję.

6. PCA95322 diody LED

6.1 Cel

W projekcie wykorzystano 16-bitowy ekspander I/O z funkcją ściemniacza LED, PCA9532, który na płytce deweloperskiej jest zrealizowany w postaci 16-diodowej linijki. Jego jedynym, lecz kluczowym zadaniem w systemie alarmowym jest dostarczanie intensywnej, wizualnej sygnalizacji w momencie wyzwolenia alarmu. Migające światła mają na celu zwrócenie uwagi otoczenia oraz potencjalne odstraszenie intruza.

6.2 Implementacja

Ekspander PCA9532 jest podłączony do tej samej magistrali I2C2 co czujnik światła i komunikuję się z mikrokontrolerem jako urządzenie podrzędne (Slave) pod stałym adresem *0x60*. Całym sterowaniem diodami zarządza dedykowany sterownik pca9532.c.

W pliku main.c do sterowania diodami używana jest funkcja *moveBar(uint16_t value)*, która jest prostym opakowaniem dla funkcji *pca9532_setLeds()* ze sterownika.

```
//main.c
static void moveBar_internal(uint16_t value)
{
    pca9532_setLeds(value, 0xFFFFU);
}
```

Funkcja *moveBar* przyjmuje 16-bitową maskę, która bezpośrednio określa, które diody maja być włączone. Drugi parametr przekazywany do *pca9532_setLeds* (*0xffff*) to maska diod do wyłączenia, co w praktyce oznacza, że wszystkie diody nieobjęte maską ledOn zostaną zgaszone.

6.3 Rejestry

Sterowanie ekspanderem PCA9532 odbywa się poprzez zapis do jego wewnętrznych rejestrów. Kluczowe z nich to:

- *PSC0*, *PWM0* (adresy *0x02*, *0x03*) oraz *PSC1*, *PWM1* (adresy *0x04*, *0x05*): Rejestry te kontrolują dwa niezależne generatory PWM. PSCx (Prescaler) ustawia częstotliwość migania, a PWMx (Pulse Width Modulation) ustawia współczynnik wypełnienia (jasność). W naszym projekcie funkcjonalność migania nie jest wykorzystywana, więc rejestry te pozostają w wartościach domyślnych.

- *LS0*, *LS1*, *LS2*, *LS3* (adresy od *0x06* do *0x09*): Są to cztery kluczowe rejestry wyboru stanu diod (LED Select). Każdy z nich kontroluje 4 diody (np. *LS0* dla diod 0-3, *LS1* dla diod 4-7, itd.). Każdej diodzie przypisane są 2 bity, które definiują jej tryb pracy:
  - *00*: Dioda wyłączona
  - *01*: Dioda włączona (stan wysoki, pełna jasność).
  - *10*: Dioda miga zgodnie z ustawieniami generatora PWM0.
  - *11*: Dioda miga zgodnie z ustawieniami generatora PWM1.
- Flaga Auto-Increment (AI): PCA9532 wspiera tryb automatycznej inkrementacji adresu rejestru. Jeśli bit 4 w bajcie komendy jest ustawiony na 1, po zapisie do jednego rejestru (np. *LS0*), wewnętrzny wskaźnik adresu układu automatycznie przesuwa się na następny (*LS1*). Pozwala to na zapisanie stanu wszystkich 16 diod w jednej, ciągłej transakcji I2C.

6.4 Sposób działania i użycia w projekcie

W systemie alarmowym ekspander LED jest aktywowany wyłącznie w momencie wyzwolenia alarmu. W każdym innym stanie – gdy system jest wyłączony, uzbrojony, ale spokojny lub w trakcie autoryzacji – wszystkie diody pozostają wygaszone. Jest to realizowane poprzez wywołanie:

```
//main.c
moveBar_internal(0U);
```

Ta komenda wysyła poprzez I2C polecenie do ekspandera, aby ustawił wszystkie 16 diod w tryb *00* (dioda wyłączona).

Najważniejsza rola ekspandera pojawia się w momencie gdy flaga alarmTriggered przyjmuje wartość 1. W pętli głównej programu, w skecji odpowiedzialnej za sygnalizację alarmu, realizowane jest naprzemienne miganie dwóch grup diod.

Logika ta jest zaimplementowana w następujący sposób

```
// main.c, fragment pętli głównej w stanie alarmu
if ((isArmed != 0U) && (isAuthorized == 0U) && (alarmTriggered != 0U)) {
if (tick_counter == 0U) {
moveBar_internal(0xFFU);     // Włącza diody 0-7 (maska 0x00FF)
tick_counter = 1U;
} else {
moveBar_internal(0xFF00U);   // Włącza diody 8-15 (maska 0xFF00)
tick_counter = 0U;
}
runAlarm_internal();
}
```

- W jednej iteracji pętli (tick_counter = 0U), wywoływana jest funkcja moveBar_internal(0xFFU), co zapala pierwszą połowę diod (bity 0-7).
- W kolejnej iteracji (tick_counter = 1U), wywoływana jest funkcja moveBar_internal(0xFF00U), która gasi pierwszą połowę i zapala drugą połowę diod (bity 8-15).
- Zmienna tick_counter jest naprzemiennie ustawiana jest na 0 i 1, co sprawia, że cykl się powtarza.

Ponieważ główna pętla programu wykonuje się co 100 ms, efekt ten powoduje szybkie, parzemienne miganie diod z częstotliwością 5 Hz (każdy stan trwa 100 ms, pełny cykl 200ms), co jest dobrze widoczne i skutecznie sygnalizuje stan alarmowy.

Każde wywołanie funkcji movebar_internal() uruchamia sekwencję operacji wewnątrz sterownika pca9532.c, która kończy się transakcją I2C.

1. pca9532_setLeds(ledOnMask, ledOffMask):

   Funkcja ta operuje na trzech wewnętrznych, 16-bitowych zmiennych typu "shadow": ledStateShadow, blink0Shadow i blink1Shadow. Przechowują one w pamięci mikrokontrolera pożądany stan dla każdej z 16 diod, odpowiednio dla trybu: stałego włączenia, migania z generatorem PWM0 i migania z generatorem PWM1.

   Krok 1: Wyłączenie diod w trybie stałym. Używając maski ledOffMask, funkcja zeruje bity w ledStateShadow odpowiadające diodom, które mają być wyłączone.

   Krok 2: Włączenie diod w trybie stałym. Następnie, używając maski ledOnMask, ustawia bity dla diod, które mają być włączone. Ta operacja ma priorytet, ponieważ odbywa się po wyłączeniu.

   Krok 3: Wyłączenie diod w trybach migania. Funkcja używa tej samej maski ledOffMask, aby wyłączyć diody, które mogły być wcześniej ustawione w tryb migania. Gwarantuje to, że polecenie "wyłącz diodę" ma najwyższy priorytet i anuluje każdy inny jej stan.

Krok 4: Aktualizacja sprzętu. Na końcu wywoływana jest wewnętrzna funkcja setLeds(), aby zaktualizować stan fizycznych diod na podstawie zmodyfikowanych zmiennych "shadow".

```
void pca9532_setLeds (uint16_t ledOnMask, uint16_t ledOffMask)
{
    /* turn off leds */
    ledStateShadow &= (~(ledOffMask) & 0xffff);

    /* ledOnMask has priority over ledOffMask */
    ledStateShadow |= ledOnMask;

    /* turn off blinking */
    blink0Shadow &= (~(ledOffMask) & 0xffff);
    blink1Shadow &= (~(ledOffMask) & 0xffff);

    setLeds();
}
```

2. setLeds(viod):
    - Jest to funkcja, która przygotowuje i wysyła dane przez I2C
    - Przygotowanie danych: Tworzy 4-elementową tablicę ls, która będzie reprezentować wartości dla rejestrów LS0-LS3 ekspandera. Funkcja setLsStates jest wywoływana trzykrotnie, aby wypełnić tę tablicę na podstawie wszystkich trzech zmiennych „shadow” (ledStateShadow, blink0Shadow, blink1Shadow). Każdej diodzie przypisane są 2 bity, które są ustawiane na odpowiedni tryb (0b01 dla ON, 0b10 dla BLINK0, 0b11 dla BLINK1).
    - Transakcja I2C: Przygotowuje 5-bajtowy bufor do wysłania:
        - buf[0]: Adres pierwszego rejestru do zapisu (PCA9532_LS0) podłączony z flagą Auto-Increment (PCA9532_AUTO_INC).
        - buf[1] do buf[4]: Wartości dla rejestrów LS0, LS1, LS2 i LS3, pobrane z przygotowanej wcześniej tablicy ls.
    - Na końcu wywołuje I2CWrite(), która realizuje fizyczną transakcję zapisu tych 5 bajtów na magistrali I2C. Dzięki trybowi AutoIncrement, stan wszystkich 16 diod jest aktualizowany w jednej, wydajnej operacji.

7. Czujnik natężenia światła

7.1 Wprowadzenie i rola w projekcie.

W projekcie wykorzystano cyfrowy czujnik natężenia oświetlenia ISL29003 firmy Intersil, który jest bezpośrednio zintegrowany na używanej przez nas płytce deweloperskiej LPCXpresso. Układ ten, przetwarza natężenie światła padającego na jego powierzchnię na 16-bitową wartość cyfrową. Komunikacja z czujnikiem odbywa się za pośrednictwem magistrali I^2C.

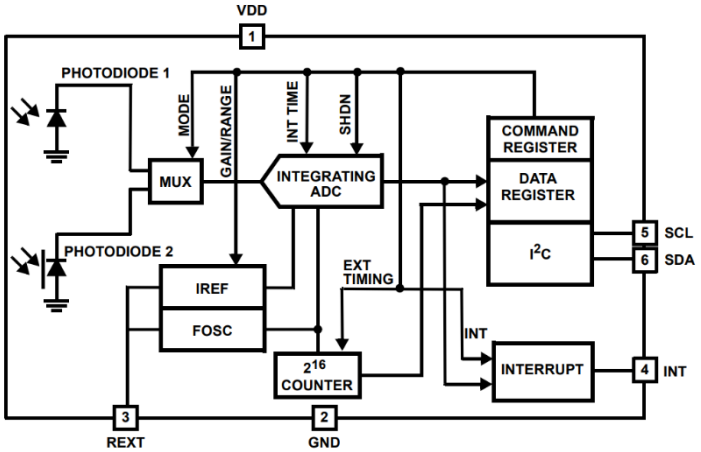
Głównym zadaniem czujnika w zrealizowanym projekcie systemu alarmowego jest pomiar natężenia światła otoczenia, dzięki czemu realizowana jest funkcja „Auto Inwersji" wyświetlacza OLED.

W zależności od wartości pomiaru wyświetlacz przyjmuje dwa stany:

- Tryb dzienny (dobre warunki oświetleniowe), wyświetlacz OLED działa w trybie normalnym (czarne znaki na białym tle).
- Tryb nocny (słabe warunki oświetleniowe), w przypadku kiedy wartość zmiennej DARK_THRESHOLD jest mniejsza niż 30 lux, kolory wyświetlacza są odwracane (białe znaki na czarnym tle).

7.2 Teoria działania czujnika ISL29003

Działanie układu ISL29003 opiera się na kilku kluczowych elementach, które zostały przedstawioen na poniższym schemacie blokowym z noty katalogowej.



7.2.1 Fotodiody

Czujnik posiada dwie wbudowane fotodiody o różnej czułości spektralnej:

- PHOTODIODE 1 – Czuła zarówno na światło widzialne, jak i na promieniowanie podczerwone (IR).
- PHOTODIODE 2 – Czuła głównie na promieniowanie podczerwone (IR)

Taka konstrukcja pozwala na skuteczne odrzucenie promieniowania podczerwonego, którego ludzkie oko nie rejestruje, a które jest emitowane przez wiele sztucznych źródeł światła (np. piloty do telewizorów, żarówki). Aby pomiar odpowiadał percepcji ludzkiego oka, czujnik może pracować w trybie, w którym odczyt z fotodiody 2 jest odpowiednio przeskalowany i odejmowany od odczytu z fotodiody 1.

W naszym projekcie wykorzystano uproszczony pomiar wyłącznie z fotodiody 1, co jest w pełni wystarczające do detekcji ogólnego poziomu jasności na potrzeby funkcji Auto Inwersji.

7.2.2 Przetwornik Analogow-Cyfrowy (ADC)

Układ wykorzystuje 16-bitowy przetwornik ADC Typu całkującego (charge-balacing integrating type). Prąd generowany przez wybraną fotodiodę (proporcjonalny do natężenia światła) jest całkowany (akumulowany) przez określony czas, zwany czasem integracji. Po tym czasie skumulowany ładunek jest zamieniany na wartość cyfrową.

7.2.3 Interfejs I2C i rejestry

Wszelka konfiguracja czujnika oraz odczyt wyników pomiaru odbywa się poprzez magistralę I2C. Układ ISL29003 posiada stały, 7-bitowy adres slave: 0b1000100 (0x44). Mikrokontroler (master) komunikuje się z czujnikiem, zapisując i odczytując wartości z jego 8-bitowych rejestrów wewnętrznych.

7.3 Konfiguracja i użycie w projekcie

7.3.1 Rejestr Command (adres 0x00)

Jest to główny, 8-bitowy rejestr sterujący pracą czujnika:

- Bit 7 (Enable): Ustawienie na 1 włącza rdzeń ADC i rozpoczyna cykl konwersji. Ustawienie na 0 resetuje i wyłącza ADC.
- Bit 6 (ADCPD): Ustawienie na 1 wprowadza układ w tryb niskiego poboru mocy. W projekcie bit ten jest ustawiony na 0 (normalna praca).
- Bit 5 (Timing_Mode): Ustawienie na 0 aktywuje wewnętrzny generator do taktowania czasu integracji. W projekcie używany jest ten tryb.
- Bit 4 Reserved: Bit zarezerwowany, należy go ustawić na 0.
- Bity 3:2 (Mode<1:0>): Umożliwiają wybranie trybu pracy fotodiod.
  - 00: Pomiar tylko z Diody 1 (światło widzialne + IR).
  - 01: Pomiar tylko z Diody 2 (głownie IR).
  - 10: Tryb różnicowy (Dioda 1 – Dioda 2) dla odrzucenia IR.
- Bity 1:0 (Width<1:0>): Określają rozdzielczości (i czas integracji).
  - 00: 16 bitów (czas integracji ~100 ms)

- 01: 12 bitów (~6.25 ms)
- 10: 8 bitów (~0.4 ms)
- 11: 4 bity (~25 µs)

W funkcji light_enable() do rejestru komend (ADDR_CMD) zapisywana jest wartość 0b10000000 (0x80). Oznacza to, że przetwornik ADC jest włączony, a pozostałe bity (6:0) przyjmują domyślne wartości 0, co skutkuje pracą z fotodiody 1 i 16-bitową rozdzielczością.

```
// light.c, fragment funkcji light_enable()
void light_enable (void)
{
    uint8_t buf[2];
    buf[0] = ADDR_CMD;       // Adres Rejestru Komend (0x00)
    buf[1] = CMD_ENABLE;     // Wartość do zapisu (0x80)
    I2CWrite(LIGHT_I2C_ADDR, buf, 2);
    // ...
}
```

7.3.2 Rejestr Control (adres 0x01)

Jest to drugi kluczowy rejestr konfiguracyjny. Ustawia on parametry takie jak zakres pomiarowy oraz działanie przerwań. W projekcie do rejestru Control zapisywana jest wartość 0x00, co odpowiada domyślnym ustawieniom fabrycznym, które są optymalne dla naszej aplikacji.

Konstrukcja wartości 0x00 dla rejestru Control:

- Bit 7 (Ext_Mode): Bit zarezerwowany do użytku fabrycznego. Należy zawsze go ustawić na 0.
- Bit 6 (Test_mode): Bit ustawiony na 0 oznacza pracę normalną, ustawienie go na 1 aktywuje bity testowe.
- Bit 5 (Int_Flag): Flaga przerwania (Status). Nie jest konfigurowana, tylko odczytywana lub kasowana.
- Bit 4 Reserved: Bit zarezerwowany, należy go ustawić na 0.
- Bity 3-2 (Gain<1:0>): Zakres pomiarowy (Full Scale Range):
  - 00: Zakres 1(domyślnie 0 – 1000 lux)
  - 01: Zakres 2 (0 – 4000 lux)
  - 10: Zakres 3 (0 – 16000 lux)
  - 11: Zakres 4 (0 – 64000 lux)
- Bity 1-0 (Int_Persist<1:0>): Trwałość przerwania (Persistency). Określa ona po ilu kolejnych pomiarach musi wystąpić przerkoczenie progu, aby wygenerować przerwanie. Ponieważ nie używamy przerwań

ustawienie to nie ma wpływu, zaś można ustawić je w następujący sposób:

- 00: 1 cykl
- 01: 4 cykle
- 10: 8 cykli
- 11: 16 cykli

W naszym kodzie rejestr kontrolny (adres 0x01) pozostaje w swoim domyślnym stanie po resecie (0x00). Oznacza to, że nie jest on modyfikowany, ponieważ domyślne ustawienia (zwłaszcza Zakres 1 do 1000 luksów) są optymalne dla naszej aplikacji.

7.3.3 Odczyt i konwersja czujnika światła

Pomiar natężenia światła jest realizowany cyklicznie w pętli głównej programu przez funkcję checkLightAndInversion(), która wywołuje light_read().

```
// main.c, fragment funkcji checkLightAndInversion()
lightLevel = light_read();
```

Funkcja light_read() realizuje odczyt i konwersję danych w następujący sposób:

1. Odczyt surowej wartości: Odczytuje dwa bajty z czujnika:
   a. Mniej znaczący bajt (LSB) z Rejestru Danych LSB (adres 0x04).
   b. Bardziej znaczący bajt (MSB) z Rejestru Danych MSB (adres 0x05).

   Oba bajty są następnie łączone w jedną 16-bitową liczbnę data.

2. Konwersja na Lux: Sudowa wartość data jest przeliczna na fizyczną jednostkę (lux) zgodnie ze wzorem z noty katalogowej, zaimplementowanym w kodzie jako:

```
// light.c, fragment funkcji light_read()
return (range * data / width);
```

Dla konfiguracji użytej w naszym projekcie, zmienne te mają następujące wartości:

- Range = RANGE_K1 = 973 (stała kalibracyjna dla zakresu do 1000 lux.
- Width = WIDTH_16_VAL = 65536 (pełna skala dla 16-bitowego przetwornika 2^{16}).

8. GPIO – Ogólna charakterystyka i użycie (Joystick oraz kontaktron)

8.1 GPIO – Ogólna charakterystyka

GPIO (ang. General-Purpose Input/Output) to interfejs służący do komunikacji pomiędzy mikroprocesorem a urządzeniami peryferyjnymi. Piny GPIO mogą pełnić funkcję wejść (INPUT), służących do odbierania sygnałów z zewnątrz, jak i wyjść (OUTPUT), umożliwiających sterowanie podłączonymi komponentami. Ich działanie realizowane jest poprzez zestaw dedykowanych rejestrów konfiguracyjnych oraz dostępowych.

Rejestry GPIO (General Purpose Input/Output) w mikrokontrolerze LPC1769 mają rozmiar 32 bitów. Każdy bit reprezentuje jeden pin danego portu (np. bit n = P0.n, bit 26 = P0.26). Poniższa tabela zawiera zestawienie kluczowych rejestrów GPIO, ich funkcji oraz zastosowania w projekcie alarmowym:

Rejestr	Rozmiar	Opis	Przykład użycia w projekcie
FIODIR	32 bity	Ustawia kierunek pinów: 0 = wejście, 1 = wyjście	Joystick (P0.x) jako wejście, Głośnik (P0.26) jako wyjście
FIOPIN	32 bity	Odczytuje stan logiczny na pinach	Sprawdzenie stanu kontaktronu lub joysticka
FIOSET	32 bity	Ustawia wybrane piny w stan wysoki (1)	Aktywacja sygnału dźwiękowego na P0.26
FIOCLR	32 bity	Ustawia wybrane piny w stan niski (0)	Dezaktywacja sygnału dźwiękowego na P0.26
FIOMASK	32 bity	Rejestr maski dla portu. Rejestry FIOSET i FIOCLR zmieniają lub zwracają tylko bity włączone przez ustawienie zer w tym rejestrze.	Nie wykorzystywany w tym projekcie.

Dzięki GPIO w projekcie możliwa jest m.in. obsługa joysticka (przycisk CENTER), wykrywanie stanu drzwi (kontaktron) oraz sterowanie sygnalizacją dźwiękową.

8.2 Zastosowanie GPIO w joysticku

Joystick w tym projekcie pełni funkcję interfejsu użytkownika, umożliwiając uzbrojenie lub rozbrojenie systemu alarmowego. Mimo że joystick posiada pięć kierunków, w aplikacji wykorzystywany jest wyłącznie przycisk środkowy (CENTER).

W trybie oczekiwania pin joysticka znajduje się w stanie logicznym wysokim (pull-up). W momencie naciśnięcia przycisku dochodzi do zwarcia z masą, co powoduje przejście pinu w stan niski. Jest to interpretowane przez mikrokontroler jako aktywacja funkcji.

W programie stan pinu przypisanego do joysticka odczytywany jest cyklicznie. W momencie wykrycia zmiany stanu z wysokiego na niski, następuje zmiana trybu systemu (uzbrojenie/rozbrojenie) wraz z informacją dźwiękową i wizualną (ekran OLED).

Aby możliwa była obsługa, pin joysticka został skonfigurowany jako wejście cyfrowe – wpisując odpowiednią wartość do rejestru FIODIR. Następnie jego stan jest odczytywany poprzez FIOPIN.

8.3 Zastosowanie GPIO w kontaktronie

Kontaktron to czujnik magnetyczny zamknięcia drzwi. W stanie spoczynku (drzwi zamknięte) obwód jest zwarty, co odpowiada stanowi logicznemu niskiemu (0) na pinie. W momencie otwarcia drzwi obwód zostaje przerwany i pin przechodzi w stan wysoki (1).

Pin przypisany do kontaktronu skonfigurowano jako wejściowy poprzez ustawienie odpowiedniego bitu w rejestrze FIODIR. Jego stan odczytywany jest bezpośrednio z rejestru FIOPIN. Detekcja naruszenia drzwi następuje w pętli głównej programu i, w przypadku gdy system jest uzbrojony i użytkownik nie został wcześniej autoryzowany przez RFID, uruchamiany jest alarm.

Dzięki prostemu i niezawodnemu działaniu kontaktronu możliwa jest szybka i bezpieczna reakcja systemu na próby nieautoryzowanego otwarcia drzwi.

9. Czujnik ultradźwiękowy odległości

9.1 Wprowadzenie

Czujnik ultradźwiękowy HC-SR04 służy do pomiaru odległości obiektów w oparciu o czas powrotu fali ultradźwiękowej. Składa się z dwóch głównych elementów: nadajnika (TRIG) oraz odbiornika (ECHO). Po podaniu impulsu na pin TRIG czujnik wysyła ultradźwięki, które po odbiciu od przeszkody są odbierane przez ECHO. Na podstawie czasu trwania stanu wysokiego na pinie ECHO można obliczyć odległość od obiektu.

9.2 Wykorzystane piny GPIO

Moduł czujnika podłączony jest bezpośrednio do pinów mikrokontrolera LPC1769 poprzez interfejs GPIO:

- P0.23 – TRIG (wyjście)
- P0.24 – ECHO (wejście)

W kodzie TRIG definiowany jest jako bit 23 portu 0 (1<<23), natomiast ECHO jako bit 24 portu 0 (1<<24).

9.3 Rejestry GPIO – szczegółowy opis

Do obsługi czujnika wykorzystywane są standardowe rejestry GPIO mikrokontrolera LPC1769:

- FIODIR – Rejestr kierunku pinu. Określa, czy pin działa jako wejście (0) czy wyjście (1). Dla TRIG ustawiany jako OUTPUT, ECHO jako INPUT.
- FIOSET – Ustawia poziom wysoki (logiczne 1) na pinach skonfigurowanych jako wyjście. Używany do podania impulsu na TRIG.
- FIOCLR – Ustawia poziom niski (logiczne 0) na pinach wyjściowych. Również wykorzystywany do TRIG.
- FIOPIN – Odczyt aktualnego stanu logicznego pinów. Używany do sprawdzenia, kiedy ECHO zmienia stan.

Przy generowaniu impulsu TRIG najpierw ustawia się pin na wysoki poziom (FIOPIN |= TRIG), po czym szybko zeruje (FIOCLR |= TRIG), by wygenerować impuls ~15 µs. Następnie następuje pomiar czasu trwania stanu wysokiego na pinie ECHO poprzez odczyt rejestru FIOPIN i zliczanie iteracji.

9.4 Rola w projekcie

W systemie alarmowym czujnik odległości pełni funkcję wykrywania ruchu. W momencie uzbrojenia systemu, funkcja checkMotionDetection() co pewien czas wywołuje procedurę pomiarową. Jeśli wykryta odległość mieści się w zakresie 5–200 cm, a użytkownik nie jest autoryzowany, ustawiana jest flaga alarmu i

aktywowany jest alarm dźwiękowy oraz wizualny. Informacje są także prezentowane na ekranie OLED.

Aby zminimalizować błędy pomiarowe, funkcja obsługująca czujnik zawiera mechanizmy timeoutów – zarówno na oczekiwanie rozpoczęcia impulsu ECHO, jak i jego zakończenia. Wartość echo_counter przeliczana jest bezpośrednio na centymetry przy założeniu szybkości dźwięku i częstotliwości zegara.

10. Timer

10.1 Wprowadzenie

W projekcie systemu alarmowego Timer0 mikrokontrolera LPC1769 pełni kluczową rolę w odmierzaniu czasu oraz generowaniu sygnałów czasowych. Wykorzystano go do tworzenia opóźnień milisekundowych i mikrosekundowych, a także do generowania dźwięków przez przebiegi prostokątne. Dzięki jego zastosowaniu możliwe było zminimalizowanie użycia opóźnień programowych (np. pętli for), co zwiększyło niezawodność i czytelność kodu.

10.2 Konfiguracja zasilania i zegara Timer0

Zegar Timer0 wymaga aktywacji zasilania i ustawienia źródła taktowania.

- Zasilanie: funkcja CLKPWR_ConfigPPWR() ustawia bit PCTIM0 w rejestrze PCONP, włączając zasilanie peryferyjne timera.
- Źródło zegara: po inicjalizacji PLL procesor pracuje z częstotliwością CCLK = 100 MHz. Funkcją CLKPWR_SetPCLKDiv() ustawiono w rejestrze PCLKSEL0 bity 3:2 na 0b00, co wybiera dzielnik CCLK/4.
  PCLK_TIMER0 = 100 MHz / 4 = 25 MHz, co daje czas jednego cyklu timera T_cyklu = 1 / 25 MHz ≈ 40 ns.

10.3 Rejestry sterujące Timer0

W celu poprawnego działania Timer0 należy skonfigurować szereg rejestrów. Poniższa tabela przedstawia rejestry sprzętowe wykorzystywane przy konfiguracji i obsłudze Timer0 w mikrokontrolerze LPC 1769, wraz z opisem kluczowych bitów i funkcji.

Rejestr	Bit / Pole	Opis
PCONP	Bit 1 (PCTIM0)	1 = Timer0 zasilany, 0 = wyłączony
PCLKSEL0	Bity 3:2	00 = CCLK/4, 01 = CCLK/1, 10 = CCLK/2, 11 = CCLK/8
T0TCR	Bit 0 (CE)	1 = włącz timer, 0 = wyłącz timer
T0TCR	Bit 1 (CR)	1 = reset TC i PC, 0 = brak resetu

T0PR	Wartość całkowita	Preskaler – określa co ile cykli PCLK zwiększa się TC
T0PC	Licznik	Zlicza cykle PCLK do osiągnięcia PR
T0TC	Licznik	Zlicza czas – główny licznik timera
T0MR0	Wartość	Wartość porównania – po jej osiągnięciu aktywacja MCR
T0MCR	Bit 0	1 = generuj przerwanie przy T0MR0
T0MCR	Bit 1	1 = resetuj timer po T0MR0
T0MCR	Bit 2	1 = zatrzymaj timer po T0MR0
T0IR	Bit 0	Flaga przerwania MR0 (1 = aktywne)

Standardowo ustawia się PR = 999, co oznacza, że TC inkrementuje się co 1000 cykli, czyli co 1 ms.

10.4 Funkcja Timer0_Wait()

Funkcja Timer0_Wait(uint32_t time_ms) wprowadza precyzyjne opóźnienia w milisekundach.

Schemat działania:

- Konfiguracja Timer0 w trybie czasowym.
- Ustawienie PR = 24 999 (czyli 25 000 * 40 ns = 1 ms).
- Ustawienie MR0 = time_ms.
- Włączenie timera i oczekiwanie w pętli na ustawienie bitu 0 w T0IR.
- Zatrzymanie timera, wyzerowanie flagi przerwania i przywrócenie stanu początkowego.

Dzięki przeskalowaniu do 1 ms, wartości time_ms mogą sięgać tysięcy (np. 100 = 100 ms).

10.5 Funkcja Timer0_us_Wait()

Funkcja Timer0_us_Wait(uint32_t time_us) wprowadza opóźnienia w mikrosekundach.

Konfiguracja różni się tylko preskalerem:

- PR = 24 (25 * 40 ns = 1 µs).
- MR0 = time_us.

Timer zwiększa TC co 1 µs, więc time_us określa żądany czas z rozdzielczością mikrosekundy.

10.6 Generowanie dźwięku z Timer0

Timer0_us_Wait() jest używana w funkcji playNote() do generowania przebiegu prostokątnego. Dla częstotliwości f:

- okres T = 1/f,
- półokres T/2 wyznacza czas stanu wysokiego i niskiego pinu P0.26.

Przykład: f = 880 Hz → T/2 ≈ 568 µs → Timer0_us_Wait(568) pomiędzy zmianami stanu pinu.

10.7 Obliczenia częstotliwości i preskalera

Dane:

- PCLK = 25 MHz → T_cyklu = 40 ns.

Opóźnienia milisekundowe:

- PR_ms = 24 999
- T_ink(TC) = (PR_ms + 1) * T_cyklu
- T_ink(TC) = 25 000 * 40 ns = 1 ms

Opóźnienia mikrosekundowe:

- PR_us = 24
- T_ink(TC) = 25 * 40 ns = 1 µs

Tak dobrane preskalery pozwalają wygodnie operować na pełnych milisekundach i mikrosekundach bez dodatkowych podziałów lub mnożeń.

10.8 Zastosowanie Timer0 do sterowania pętlą główną i liczników _ticks

W programie głównym Timer0_Wait(100) jest wywoływana w każdej iteracji pętli while(1). Oznacza to, że pętla wykonuje się co dokładnie 100 ms. Na tej bazie realizowane są wszystkie „soft-timery” w systemie, np.:

- authorizationTimer_ticks – odlicza 10 s (100 iteracji),
- motionCheckTimer_ticks – co 200 ms inicjuje pomiar HC-SR04,
- displayUpdateTimer_ticks – co 500 ms odświeża ekran OLED,
- lightCheckTimer_ticks – co 3 s sprawdza jasność otoczenia.

Zmienna _ticks reprezentuje zatem licznik programowy oparty o główną pętlę 100 ms, a nie o przerwania sprzętowe.

11. Głośnik

11.1 Cel

W projekcie systemu alarmowego opartym na mikrokontrolerze LPC1769, moduł głośnika pełni funkcję sygnalizacji dźwiękowej stanów systemu – takich jak aktywacja, dezaktywacja alarmu oraz wykrycie zagrożenia. Głośnik jest elementem wbudowanym w płytkę i sterowany przy pomocy sygnału prostokątnego generowanego programowo.

11.2 Wykorzystane piny GPIO

W celu sterowania głośnikiem wykorzystywany jest pin P0.26 mikrokontrolera, który skonfigurowany jest jako wyjście (OUTPUT). Wartość wysoka (HIGH) i niska (LOW) na tym pinie odpowiadają odpowiednio za generowanie dźwięku poprzez szybkie przełączanie stanu logicznego. Piny P0.27, P0.28 oraz P2.13 służą do sterowania zewnętrznym wzmacniaczem LM4811, który umożliwia uzyskanie słyszalnego dźwięku na podłączonym głośniku.

11.3 Działanie modułu

Głośnik generuje dźwięki za pomocą funkcji wykorzystującej mechanizm opóźnień i cyklicznego przełączania stanu pinu P0.26. Program umożliwia odtwarzanie dźwięków o określonej częstotliwości (np. wysoki i niski ton alarmowy), a także sygnałów potwierdzających akcje użytkownika, takie jak uzbrojenie czy rozbrojenie systemu. Dzięki temu użytkownik otrzymuje natychmiastową informację zwrotną o stanie systemu.

11.4 Wzmacniacz LM4811

Aby uzyskać odpowiedni poziom głośności, wykorzystano zintegrowany na płytce wzmacniacz audio LM4811. Wzmacniacz ten jest kontrolowany przez trzy linie GPIO: P0.27 (CLK), P0.28 (UP/DN) oraz P2.13 (SHUTDOWN). Włączenie wzmacniacza odbywa się poprzez ustawienie stanu niskiego (LOW) na pinie SHUTDOWN. Sterowanie pozostałymi pinami umożliwia potencjalną kontrolę głośności, choć w projekcie zastosowano stałe wartości.

11.5 Podsumowanie

Moduł głośnika wykorzystuje proste techniki programowego PWM do generacji dźwięków oraz zintegrowany wzmacniacz LM4811 do zwiększenia sygnału audio. Sterowanie odbywa się za pomocą GPIO mikrokontrolera, bez użycia specjalistycznych interfejsów komunikacyjnych. Dzięki temu moduł jest łatwy w implementacji i skutecznie spełnia swoją funkcję ostrzegania dźwiękowego w systemie alarmowym.

12. Analiza skutków awarii

12.1 Koszt Awarii

Moduł	Szansa	Skutki Awarii	Wykrycie (Koszt)	Reakcja (Koszt)	Iloczyn
Magistrala I2C (Czujnik światła, LED bar)	0.2	Niegroźne(1)	5	2	2
Magistrala SSP (OLED, RFID)	0.2	Krytyczne(10)	8	8	128
Wyświetlacz OLED	0.4	Średnie(5)	7	3	42
Czytnik RFID	0.3	Krytyczne(10)	8	8	192
Czujnik światła	0.1	Niegroźne(1)	6	1	1
LED Expander (PCA9532)	0.1	Niegroźne(1)	5	2	1
Joystick (GPIO)	0.5	Krytyczne(10)	10	10	500
Kontaktron (GPIO)	0.3	Krytyczne(10)	9	9	243
Czujnik Ultradźwiękowy (GPIO)	0.4	Krytyczne(10)	5	4	80
Timer	0.2	Krytyczne(10)	3	4	24
Buzzer / Wzmacniacz Audio	0.3	Średnie(8)	8	2	38

12.2 Wykrycie i reakcja na awarie

W przypadku uszkodzenia mechanicznego przycisku „center” joysticka, który przestaje przewodzić prąd po wciśnięciu:

- Skutek: Użytkownik traci możliwość uzbrojenia i rozbrojenia alarmu. Jest to krytyczna utrata funkcjonalności..
- Sposób wykrycia: Brak. System nie jest w stanie odróżnić braku wciśnięcia od fizycznego uszkodzenia.
- Reakcja systemu: Brak.

Gdy czujnik kontaktronowy ulegnie zwarciu i na stałe będzie podawał stan „zamknięty”, mimo fizycznego otwarcia drzwi:

- Skutek: Krytyczny i najgroźniejszy. System nigdy nie wykryje włamania przez te drzwi, co prowadzi do całkowitej utraty ochrony przy jednoczesnym pozornym, poprawnym działaniu.
- Sposób wykrycia: Brak. System bezwarunkowo ufa sygnałowi z pinu.
- Reakcja systemu: Brak.

W sytuacji, gdy uszkodzeniu ulegnie antena czytnika, uniemożliwiając wykrycie jakiejkolwiek karty:

- Skutek: Użytkownik traci podstawową metodę autoryzacji i wyciszenia alarmu. Musi polegać wyłącznie na joysticku.
- Sposób wykrycia: Brak. System nie odróżnia sytuacji „brak karty w polu" od „uszkodzony czytnik".
- Reakcja systemu: Brak.

Jeśli nastąpi przerwa w obwodzie pinu *ECHO,* przez co sygnał nigdy nie wraca do mikrokontrolera:

- Skutek: Całkowita utrata funkcji detekcji ruchu.
- Sposób wykrycia: Tak, częściowo. Pętla oczekująca na sygnał *ECHO* w funkcji *getUltrasonicDistance_internal* zadziała jak timeout.
- Reakcja systemu: Funkcja zwróci wartość błędu (-1.0f). Pętla główna zinterpretuje to jako brak deteckji ruchu, więc system nie wywoła fałszywego alarmu.

W przypadku zwarcia na linii zegara (SCK) do masy:

- Skutek: Krytyczny paraliż komunikacji z wyświetlaczem OLED i czytnikiem RFID.
- Sposób wykrycia: Brak. Funkcje SSP_ReadWrite nie mają zaimplementowanej obsługi timeoutów, a ich kody powrotne nie są sprawdzane.
- Reakcja systemu: Brak.

Jeśli z powodu błędu programistycznego program zawiesi się w pętli oczekiwania na timer (np. w Timer0_Wait):

- Skutek: Krytyczne zatrzymanie całego systemu, który przestaje reagować na jakiekolwiek zdarzenia.
- Sposób wykrycia: Brak.
- Reakcja systemu: Brak. (Standardowym rozwiązaniem byłoby użycie Watchdog Timera do zresetowania systemu).