Dokumentacja projektu: Inteligentna donica

Julia Dobroszek 251504
Aleksander Kaźmierczak 251544 (kierownik grupy)
Malwina Wodnicka 251663

11 czerwca 2025

|

Funkcjonalność	Członek zespołu
I2C Czujnik światła PWM ADC (czujnik wilgotności)	Aleksander Kaźmierczak (34%)
GPIO Obsługa głośnika Timer	Malwina Wodnicka (33%)
SPI OLED Obsługa termometru	Julia Dobroszek (33%)

Tabela 1: Zawartość zespołu

1 Instrukcja użytkowania

1.1 Uruchomienie urządzenia

Aby uruchomić inteligentną donicę, należy wykonać następujące kroki:

1. Podłączyć urządzenie do źródła zasilania za pomocą dołączonego przewodu.
2. Umieścić czujnik wilgotności (dwa metalowe druty) w glebie w doniczce. Czujnik powinien być umieszczony na głębokości około 3-5 cm, aby zapewnić prawidłowy pomiar wilgotności gleby.
3. Po podłączeniu zasilania, system automatycznie rozpocznie pracę i wyświetli informacje na ekranie OLED.

1.2 Interpretacja wskaźników

Inteligentna donica dostarcza użytkownikowi informacji o stanie rośliny poprzez:

1. **Wyświetlacz OLED** - Na ekranie prezentowane są aktualne statystyki dotyczące:
   - Poziomu wilgotności gleby:
     - **sucho** - oznacza niski poziom wilgotności gleby, wskazujący na konieczność podlania rośliny
     - **w normie** - oznacza optymalny poziom wilgotności gleby, idealny dla wzrostu większości roślin
     - **mokro** - oznacza wysoki poziom wilgotności gleby, wskazujący na nadmiar wody, który może być szkodliwy dla rośliny
   - Temperatury otoczenia (w stopniach Celsjusza)
   - Natężenia światła - system interpretuje odczyt z czujnika i wyświetla jeden z trzech komunikatów:
     - "jasno gdy wartość odczytu przekracza 2000 lux
     - "OK gdy wartość odczytu mieści się w przedziale od 350 do 2000 lux

  - "ciemno gdy wartość odczytu jest mniejsza niż 350 lux
- Dodatkowych informacji o stanie systemu

2. **Dioda RGB** - W zależności od poziomu wilgotności gleby, dioda RGB świeci się różnymi kolorami:
   - Kolor żółto-zielony (#c8ff1e) - oznacza stan **sucho**, sygnalizując potrzebę podlania rośliny
   - Kolor zielony (#1eff00) - oznacza stan **w normie**, sygnalizując optymalny poziom wilgotności
   - Kolor błękitny (#1effff) - oznacza stan **mokro**, sygnalizując nadmiar wody w glebie

1.3 Kalibracja i konserwacja

1. **Kalibracja** - System został skalibrowany fabrycznie na podstawie eksperymentów empirycznych. Urządzenie nie posiada możliwości zewnętrznej kalibracji przez użytkownika.
2. **Konserwacja** - Aby zapewnić prawidłowe działanie urządzenia:
   - Utrzymywać wyświetlacz OLED w czystości, unikając bezpośredniego kontaktu z wodą
   - **Kategorycznie unikać kontaktu elektroniki z wodą** - Jedynym elementem, który może mieć bezpośredni kontakt z wilgocią, jest czujnik wilgotności (metalowe druty). Pozostałe elementy elektroniczne należy chronić przed zalaniem, zachlapaniem oraz nadmierną wilgotnością powietrza, gdyż może to prowadzić do trwałego uszkodzenia urządzenia i utraty gwarancji

2 Opis głównej pętli programu

2.1 Inicjalizacja

- Konfiguracja portów GPIO
- Inicjalizacja interfejsów komunikacyjnych (I2C, SPI)
- Inicjalizacja przetwornika ADC
- Inicjalizacja peryferiów: wyświetlacz OLED, czujnik światła, moduł PWM, czujnik temperatury
- Konfiguracja timera SysTick dla odliczania milisekund

2.2 Główna pętla

W każdej iteracji pętli program wykonuje:

1. **Odczyt temperatury** - pobiera wartość z czujnika i konwertuje na stopnie Celsjusza
2. **Wyświetlanie temperatury** - aktualizuje wyświetlacz OLED z aktualną temperaturą
3. **Odczyt poziomu oświetlenia** - pobiera wartość z czujnika światła
4. **Pomiar wilgotności gleby** - wykorzystuje przetwornik ADC do odczytu z czujnika wilgotności
5. **Analiza wilgotności gleby**:
   - Gleba sucha (>2801): wyświetla "sucho", dioda świeci żółto-zielono, odtwarza smutną melodię
   - Gleba wilgotna (1601-2800): wyświetla "w normie", dioda świeci zielono, odtwarza wesołą melodię
   - Gleba mokra (<1601): wyświetla "mokro", dioda świeci cyjanowo
6. **Analiza oświetlenia**:
   - Jasno (>2000 lux): wyświetla "jasno"
   - Normalnie (350-2000 lux): wyświetla "OK"
   - Ciemno (<350 lux): wyświetla "ciemno"
7. **Opóźnienie** - 200ms między iteracjami

3 Funkcjonalności

3.1 Magistrala I2C

1. Interfejs I2C jest używany do komunikacji z czujnikiem światła oraz innymi komponentami wymagającymi komunikacji szeregowej. Implementacja obejmuje inicjalizację magistrali I2C, konfigurację adresów urządzeń oraz obsługę błędów komunikacji.

2. Płytka LPC1769 posiada trzy magistrale I2C, w projekcie została użyta magistrala I2C2.

3. Czujnik światła jest podłączony do magistrali I2C2, a jego adres jest ustawiony na 0x44. Szczegóły konfiguracji znajdują się w sekcji 3.3.

3.1.1 Omówienie rejestrów

Rejestry mikroprocesora zostały opisane w tabeli 384. I2C register map w dokumentacji płytki. Poniżej znajdują się opisane rejestry:

- **I2CONSET** - Rejestr ustawień kontrolnych I2C. Używany do ustawiania bitów w rejestrze kontrolnym I2CON. Pozwala ustawiać pojedyncze bity bez konieczności odczytywania i modyfikowania całego rejestru. Zapisanie '1' do bitu w tym rejestrze ustawia odpowiadający bit w rejestrze kontrolnym I2C. Poniżej zostały opiasne flagi:

  1. Bity 0, 1 oraz 7:31 są zarezerowane
  2. **AA** - Bit 2
     Gdy:
     AA = 1 urządzenie wyśle ACK (czyli ustawi linię SDA w stan niski w czasie impulsu zegarowego na linii SCL).
     AA = 0 urządzenie wyśle NACK (czyli pozostawi linię SDA w stanie wysokim).
  3. **SI** - Bit 3
     Flaga ta informuje o zmianie stanu magistrali np. zakończenie nadawania danych. Gdy:
     SI = 1, niski poziom sygnału SCL jest wydłużony. Transmisja zostaje wstrzymana.
     SI resetuje się poprzez zapisanie 1 do bitu SIC (SI Clear) w rejestrze I2CONCLR
  4. **STO** - Bit 4
     Flaga zakończenia transmisji danych. Jej działanie zależy od trybu, w jakim pracuje układ: master czy slave.
     W trybie master, ustawienie STO = 1 powoduje wysłanie warunku STOP na magistrali I2C, czyli SDA przechodzi z LOW na HIGH, gdy SCL jest HIGH. To kończy bieżącą transmisję i linia jest zwalniana.
     Gdy magistrala wykryje warunek STOP, flaga STO zostaje automatycznie wyczyszczona przez sprzęt.
  5. **STA** - Bit 5
     Służy do rozpoczęcia transmisji przez wysłanie warunku START
     - Jeżeli interfejs I2C jest w trybie slave, to:
       * Interfejs przechodzi do trybu master
       * Jeśli magistrala jest wolna – natychmiast wysyła warunek START (SDA z HIGH do LOW przy SCL = HIGH).
       * Jeśli magistrala jest zajęta – czeka na warunek STOP, który „uwolni" magistralę, a potem wysyła START po pół okresie zegara wewnętrznego.
     - Jeśli interfejs już jest w trybie master, to:
       * Ustawienie STA = 1 powoduje wysłanie repeated START – kolejnego warunku START bez wcześniejszego STOP.
     - W przypadku, gdy STA oraz STO ustawimy jednocześnie na 1, to:
       * w tybie master zostanie syłany sygnal STOP, następnie wysyłany jest START (restart transmisji)
       * w trybie slave nastąpi zatrzymanie transmisji bez wysłania STOP na magistralę, a następnie, jeśli magistrala jest wolna, zostanie wysłany START
  6. **I2EN** - Bit 6
     Flaga odpowiedzialna za włączenie lub wyłączenie interfejsu I2C. Gdy:
     I2EN = 0, interfejs I2C jest wyłączony
     I2EN = 1, interfejs I2C jest włączony

- **I2CONCLR** - Rejestr kasowania ustawień kontrolnych. Używany do kasowania bitów w rejestrze kontrolnym I2C. Zapisanie '1' do bitu w tym rejestrze kasuje odpowiadający bit w rejestrze kontrolnym I2C.

  1. Bity 0, 1, 4 oraz 7:31 są zarezerowane
  2. **AAC** - Bit 2 - służy do wyzerowania bitu AA (Assert Acknowledge) w rejestrze I2CON. Bit AA odpowiada za to, czy urządzenie I2C nada ACK (potwierdzenie) podczas komunikacji.
  3. **SIC** - Bit 3 - służy do czyszczenia flagi przerwania I2C (Interrupt Clear).
  4. **STAC** - Bit 5 - służy do czyszczenia flagi START (STA)
  5. **I2CENC** - Bit 6 - wyłącza interfejs I2C

- **I2STAT** - Rejestr statusu I2C. Dostarcza szczegółowe kody statusu pokazujące aktualny stan interfejsu. Z tego rejestru można tylko odczytywać dane.

  1. Bity 0:2 oraz 8:31 są zarezerowane
  2. Bity 7:3 - zawierają kod aktualnego stanu interfejsu I2C. Kod statusu możemy wyodrębnić dzięki masce 0xF8:

     ```
     uint8_t status = I2STAT & 0xF8;
     ```

     W rejestrze może wystąpić 26 kodów statusu. Kody, które mogą wystąpić znajdują się w dokumentacji LPC1769, w tabelach 399 oraz 402.
     Jeśli jakikolwiek z powyższych kodów wystąpi, to oznacza, że na pewno bit SI zostanie ustawiony na 1.

- **I2DAT** - Rejestr danych. Podczas trybu nadawania lub odbioru, dane (8 bitów) są zapisywane lub odczytywane z tego rejestru.
- **I2SCLH** - Rejestr cyklu zegara SCL (wysoka połowa).Zawiera wysoką połowę cyklu zegara I2C.
- **I2SCLL** - Rejestr cyklu zegara SCL (niska połowa). Zawiera niską połowę cyklu zegara I2C.
- **MMCTRL** - Rejestr kontrolny trybu monitorowania. Używany do kontrolowania trybu monitorowania I2C.
- **I2DATA_BUFFER** - Rejestr bufora danych I2C (Adres: 0x2C). Zawiera zawartość 8 najstarszych bitów rejestru przesuwnego I2DAT.
- **I2ADR0, I2ADR1, I2ADR2, I2ADR3** - Rejestry adresowe I2C. Zawierają 7-bitowy adres slave do działania interfejsu I2C w trybie slave. W na bicie 0 można ustawić czy urządzenie ma odpowiadać na GC (General Call) lub tylko na określony adres.
- **I2MASK0, I2MASK1, I2MASK2, I2MASK3** - Rejestry maski I2C. Służą do określenia dopasowania adresu w trybie slave.

3.1.2 Inicjalizacja magistrali I2C2

Inicjalizacja magistrali I2C2 odbywa się poprzez wykonanie następujących kroków:

1. Ustawienie odpowiednich pinów
   Użyte zostały piny P0.10 i P0.11, które odpowiadają pinom SCL i SDA na magistrali I2C2.
   Dla powyższych pinów wybrane zostały funkcje nr. 2, ponieważ z dokumentacji LPC1769 wynika, że funkcja nr. 2 odpowiada pinom SCL i SDA.

2. Włączenie zasilania magistrali I2C2
   Z tabeli 46. w dokumentacji LPC1769 można odczytać, że bit 6 w rejestrze PCONP odpowiada za włączenie zasilania magistrali I2C2.

   $$PCONP| = (1 << 26)$$

3. Ustawienie taktowania zegara
   Taktowanie zegara dla magistrali I2C2 ustawiamy poprzez wpisanie do rejestru I2SCLH i I2SCLL odpowiednich wartości. Wpierw pobieramy zawartość rejestru PCLK; robimy to poprzez odczytanie danych z bitów 21:20 z adresu 0x400F C1AC przy użyciu maski. Stąd wiemy, że dzielnikiem zegara jest 2, więc

$$PCLK_{I2C2} = \frac{CCLK}{2}$$
$$I2SCLH = \frac{PCLK_{I2C2}}{2 \cdot target\_clock}$$
$$I2SCLL = \frac{PCLK_{I2C2}}{target\_clock} - I2SCLH$$

   CCLK wynosi 100MHz. Mając powyższe wartości możemy wyliczyć taktowanie z poniższego wzoru:

$$I2C_{bitfrequency} = \frac{PCLK}{I2SCLH + I2SCLL}$$

   Z Tabeli 395 w dokumentacji LPC1769 można odczytać, że standardowe taktowanie wynosi 100kHz, więc do rejestrów I2SCLH oraz I2SCLL wpisujemy 250

4. Wyzerowanie flag rejestru I2CON
   W celu resetowania flag AC, STA, I2EN ustawiamy w rejestrze I2CONCLR bity 2, 5,6, które zerują te flagi w rejestrze I2CON.

5. Włączenie magistrali I2C
   W celu włączenia magistrali I2C ustawiamy bit 6 w rejestrze I2CON.

3.1.3 Wykorzystanie magistrali I2C2

1. Początkowe ustawienia
   Rozpoczęcie transmisji odbywa się poprzez ustawienie flagi START w rejestraze I2CON (bit 5) na 1 oraz flagi SI (przerwanie) w rejestrze I2CON (5 bit) na 0. Później master czeka na zmianę flagi SI na 1, która oznacza, że transmisja została rozpoczęta. Następnie sprawdzany jest rejest stanu I2STAT, aby sprawdzić, czy warunek START został wysłany (0x08). Ustawiamy SI na 0.

2. Wysyłanie danych

   (a) Wysyłanie adresu urządzenia. Adres urządzenia jest 7-bitowy. W rejestrze I2DAT należy wpisać adres urządzenia, a następnie ustawić pierwszy bit na 0, aby wskazać, że wysyłamy adres. Następnie ustawiamy flagę SI w rejestrze I2CON na 0.

   (b) Master oczekuje na potwierdzenie (ACK) od slave - (z Tabeli 400) w rejestrze I2STAT powinna znajdowaćsię wartość 0x18. Gdy znajduje się tam inna wartość, to adres jest ponownie wysyłany.

   (c) Po otrzymaniu ACK, master ustawia bit 5 w rejestrze I2CON (STA) na 0.

   (d) Następnie master wysyła dane. W rejestrze I2DAT należy wpisać dane, które chcemy wysłać. Następnie ustawiamy flagę SI w rejestrze I2CON na 0.

   (e) Oczekiwanie na ustawienie SI rejestru I2CON na 1

   (f) Master oczekuje na potwierdzenie (ACK) od slave w rejestrze I2STAT powinna znajdowaćsię wartość 0x28. Gdy otrzymano ACK, to przechodzi do wysłania kolejnych danych.

3. Odbieranie danych

   (a) Żeby wskazać adres urządzenia z któego będziemy odbierać dane należy wpisać 7-bitowy adres slave do rejestru I2DAT. Ponadto należy ustawić pierwszy bit na 1, aby wskazać, że odbieramy dane. Następnie ustawiamy flagę SI w rejestrze I2CON na 0.

   (b) Master oczekuje na potwierdzenei (ACK) od slave - (z Tabeli 400) w rejestrze I2STAT powinna znajdowaćsię wartość 0x40. Gdy znajduje się tam inna wartość, to adres jest ponownie wysyłany.

   (c) Po otrzymaniu ACK, master ustawia bit 5 w rejestrze I2CON (STA) na 0.

   (d) Master ustawia bit 2 w rejestrze I2CON (AA) na 1, aby ustawić ACK. I odbiera bajty danych:

      i. W rejestrze I2CON ustawiamy bit 3 (SI) na 0.

ii. Master czeka na zmianę flagi SI w rejestrze I2STAT
iii. Odczyt danych z I2DAT (8 bitów)
iv. przed ostatnim bajtem danych master ustawia bit 2 w rejestrze I2CON (AA) na 0, aby wyłączyć ACK.

4. Zmiana i odczyt rejestrów urządzeń slave

   (a) Wysyłamy adres rejestru, który chcemy odczytać lub zmodyfikować.
   (b) Jeśli chcemy modyfikować rejestr, należy wysłać nową wartość rejewstru.
   (c) Jeśli chcemy odczytać, należy zacząć odbierać dane

5. Koniec transmisji

   (a) W celu zakończenia transmisji należy ustawić bit 4 (STO) w rejestrze I2CON na 1.
   (b) Zmiana wartości flagi SI w rejestrze I2CON na 0.

3.2 Sterowanie GPIO

3.2.1 Wstęp

System wykorzystuje piny GPIO mikrokontrolera do sterowania różnymi elementami wykonawczymi donicy. Funkcjonalność ta obejmuje konfigurację kierunku pinów (wejście/wyjście), ustawianie stanów logicznych oraz obsługę przerwań sprzętowych. GPIO jest wykorzystywane między innymi do sterowania diodami sygnalizacyjnymi.

3.2.2 Konfiguracja portów GPIO

W systemie opartym o mikrokontroler LPC1768/9, obsługa pinów GPIO odbywa się dwuetapowo: najpierw należy przypisać pinowi odpowiednią funkcję, a następnie ustawić jego kierunek oraz zarządzać stanem logicznym.

Funkcja pinu Piny mikrokontrolera mogą pełnić różne role – od ogólnych wejść/wyjść (GPIO), przez interfejsy komunikacyjne (SPI, UART), aż po funkcje specjalne. Wybór funkcji dokonywany jest za pomocą rejestrów `PINSEL0` do `PINSEL9`, z których każdy kontroluje dwa bity odpowiadające konkretnemu pinowi:

- `00` – GPIO (funkcja domyślna),
- `01`, `10`, `11` – funkcje alternatywne (specyficzne dla każdego pinu).

Pełna lista funkcji alternatywnych znajduje się w dokumentacji mikrokontrolera (sekcja 8.5.1 dokumentu UM10360).

Rejestry GPIO Po przypisaniu pinu funkcji GPIO, konfiguracja odbywa się za pomocą zestawu rejestrów `FIOx`, gdzie `x` oznacza numer portu (od 0 do 4). Poniżej opis poszczególnych rejestrów:

- `FIOxDIR` – *Direction Register* (Rejestr kierunku):

  Ten rejestr pozwala ustalić kierunek działania każdego pinu portu. `1` ustawia dany pin jako wyjście (output), natomiast `0` oznacza wejście (input).

- `FIOxMASK` – *Mask Register* (Rejestr maski):

  Maskowanie pozwala ignorować niektóre bity przy operacjach odczytu/zapisu. Bity ustawione na `1` są pomijane przez funkcje korzystające z `FIOxSET`, `FIOxCLR` i `FIOxPIN`. Ustawienie bitu na `0` oznacza, że operacje na danym pinie są aktywne.

- `FIOxPIN` – *Pin Value Register* (Rejestr stanu pinów):

  Ten rejestr służy zarówno do odczytu, jak i zapisu. Odczyt zwraca aktualny stan logiczny wszystkich pinów danego portu, natomiast zapis nadpisuje piny wyjściowe wartościami logicznymi (jeśli są skonfigurowane jako output).

- `FIOxSET` – *Set Register* (Rejestr ustawień):

  Rejestr służy do ustawiania wybranych pinów wyjściowych na stan wysoki (`logic 1`). Każdy bit ustawiony na `1` wymusza stan wysoki na odpowiadającym mu pinie. Ustawienie bitu na `0` nie zmienia niczego. Nie działa na pinach wejściowych.

- `FIOxCLR` – *Clear Register* (Rejestr zerowania):

  Rejestr ten jest odwrotnością `FIOxSET`. Pozwala ustawić konkretne piny wyjściowe na stan niski (`logic 0`). Podobnie jak wcześniej, ustawienie `1` na wybranym bicie spowoduje wyczyszczenie (ustawienie stanu niskiego) danego pinu. Nie wpływa na piny wejściowe.

3.2.3 Do czego używamy GPIO?

Tabela 2: Przypisanie pinów mikrokontrolera LPC1768/9 do obsługi układu audio LM4811

Pin	Funkcja	Opis
P0.26	LM4811 SIGNAL	Generowanie sygnału dźwiękowego
P0.27	LM4811 CLK	Sterowanie zegarem układu audio LM4811
P0.28	LM4811 UP/DN	Regulacja głośności LM4811
P2.13	LM4811 SHUTDOWN	Włączenie/wyłączenie układu LM4811

Piny są skonfigurowane jako wyjścia za pomocą funkcji GPIO i sterowane przez rejestry FIOxDIR oraz FIOxSET/FIOxCLR.

3.2.4 Przerwania sprzętowe – opis i zastosowanie

- **Czym są przerwania sprzętowe?**
  Przerwania sprzętowe (ang. *hardware interrupts*) pozwalają mikrokontrolerowi natychmiast reagować na określone zdarzenia zewnętrzne lub wewnętrzne, bez konieczności ich ciągłego sprawdzania w pętli głównej programu (polling).

- **Mechanizm w LPC1768/9:**
  Obsługa przerwań w mikrokontrolerze **LPC1768/9** opiera się na wbudowanym w rdzeń **Cortex-M3** kontrolerze **NVIC** (*Nested Vectored Interrupt Controller*), który zarządza priorytetami i wywoływaniem odpowiednich procedur obsługi przerwań.

- **Typowe zastosowania przerwań:**
  - zmiana stanu pinu GPIO (np. naciśnięcie przycisku),
  - zakończenie konwersji ADC,
  - odebranie danych przez UART, SPI czy I^2C,
  - przepełnienie licznika lub zegara.

- **Podejście w aktualnym projekcie:**
  - W obecnej implementacji programu **nie są wykorzystywane przerwania**.
  - Wszystkie operacje (odczyty czujników, sterowanie wyjściami, reakcje na zmiany warunków) są realizowane sekwencyjnie w **pętli głównej** z użyciem metody *polling*.

- **Zalety i ograniczenia tego podejścia:**
  - **Zalety:** prostsza implementacja – brak konieczności definiowania procedur ISR (Interrupt Service Routine).
  - **Ograniczenia:**
    * mniejsze wykorzystanie potencjału mikrokontrolera,
    * możliwe opóźnienia w reakcji na zdarzenia,
    * ograniczona skalowalność w bardziej złożonych aplikacjach.

3.3 Czujnik światła

Inteligentna donica wykorzystuje czujnik światła oparty na protokole komunikacyjnym I2C. Implementacja tego elementu obejmuje trzy kluczowe funkcjonalności: ISL29003 to zintegrowany czujnik światła z 16-bitowym przetwornikiem ADC. Wewnętrzny przetwornik ADC zapewnia 16-bitową rozdzielczość. Ogólne zasady działania ADC zostały przedstawione w rozdziale 3.11.

1. **Inicjalizacja i konfiguracja czujnika ISL29003** - Proces inicjalizacji obejmuje ustawienie adresu urządzenia na magistrali I2C, konfigurację rejestrów czujnika oraz określenie parametrów pomiaru, takich jak czułość i częstotliwość próbkowania.
2. **Odczyt danych z czujnika** - Funkcjonalność ta odpowiada za komunikację z czujnikiem poprzez protokół I2C, wysyłanie komend odczytu do odpowiednich rejestrów oraz interpretację otrzymanych danych. Implementacja uwzględnia obsługę błędów komunikacji oraz weryfikację poprawności odczytanych wartości.
3. **Przetwarzanie i analiza danych** - Po odczytaniu surowych danych z czujnika, system przetwarza je na użyteczne informacje o poziomie natężenia światła. Funkcjonalność ta obejmuje kalibrację, filtrowanie zakłóceń oraz konwersję wartości na jednostki zrozumiałe dla użytkownika (np. luxy).

3.3.1 Inicjalizacja czujnika ISL29003

Aby zainicjować czujnik ISL29003, należy ustawić wartość 1 na 7. bicie rejestru Command (0x00). Sposób wysyłania zawartości rejestrów do urządzeń podrzędnych zostałopisany w poprzednim podrozdziale o I2C.

Wysyłanie adresu urządzenia ISL29003 na magistrali I2C - adres urządzenia to 0x44.
Informacje na temat rejestrów i ich wartości znajdują się w dokumentacji ISL29003 (Tabela 1).

Kolejnym etapem przygotowania czujnika do pracy jest zdefiniowanie zakresu. Stałe te będą używane do przeliczania danych wyjściowych czujnika. W naszym programie ustawiamy zakres na 3892, co oznacza skalę od 0 do 3892 lux. Aby ustawić wybrane przez nas wartości zakresu (range) należy sutawić bity 3:2 (tabela 9.) w rejestrze Control (0x01) na 0:1

Bits 3:2	k	RANGE (k)
0:0	1	973
0:1	2	3892
1:0	3	15,568
1:1	4	62,272

Tabela 3: Zakresy skali lux

Ustawiamy także liczbę cykli zegara na konwersację analogowo cyfrową poprzez umieszczenie wartości 0:0 na bitach 1:0 w rejestrze Command (0x00) (tabela 7.)

BITS 1:0	NUMBER OF CLOCK CYCLES
0:0	$2^{16} = 65,536$
0:1	$2^{12} = 4,096$
1:0	$2^{8} = 256$
1:1	$2^{4} = 16$

Tabela 4: Liczby cykli zegara

3.3.2 Odczyt danych z czujnika

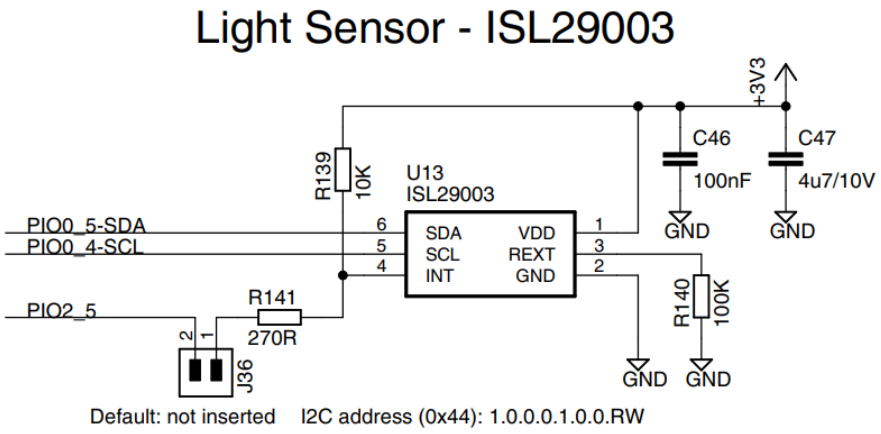
Dane z ISL29003 odczytujemy przy pomocy protokołu I2C. Dane wyjściowe przechowywane są w dchów rejestrach, ponieważ odczyt jest liczbą 16-bitową. W rejestrze o adresie 0x04 znajduje się najniższy bajt danych, a w rejestrze 0x05 najwyższy bajt. Sposób odczytywania danych został opisany w podrozdziale 3.

3.3.3 Przetwarzanie i analiza danych

Po odczytaniu danych z czujnika, system przetwarza je na użyteczne informacje o poziomie natężenia światła. Aby uzyskać wartość w luxach, należy zastosować poniższy wzór (ten wzór obowiązuje tylko gdy mamy zewnętrzny rezystor 100kΩ):

$$E = \frac{FSR \cdot DATA}{2^n} \tag{1}$$

Gdzie:
E - wartość w luxach
FSR - zakres skali (FSR = 3892 lux)
DATA - odczytana wartość z rejestru
n - liczba bitów w rejestrze (16 bitów)



Rysunek 1: Schemat podłączenia czujnika ISL29003

3.4 Czujnik temperatury

Czujnik MAX6576 zapewnia pomiar temperatury otoczenia rośliny.

3.4.1 Konfiguracja mikrokontrolera

Do obsługi czujnika MAX6576 wykorzystano pin `PIO1_5` odpowiadający pinowi `P0.2`. Konfiguracja tego pinu przebiega w następujący sposób:

- `PINSEL0` – bity 5:4 ustawione na 00, konfiguracja P0.2 jako funkcję GPIO.
- `FIODIR0` – bit 2 ustawiony na 0, co ustawia kierunek pinu P0.2 jako wejściowy.
- `FIOPIN0` – odczyt wartości logicznej pinu w celu wykrywania zboczy sygnału z czujnika.

Do zliczania czasu pomiędzy zboczami wykorzystano Timer0:

- `T0TCR` – ustawiony na 0x02 (reset), a następnie na 0x01 (start).
- `T0PR` – preskaler ustawiony tak, aby timer inkrementował wartość co 1 ms, tj. dla 100 MHz: `T0PR = 100 000 - 1`.
- `T0TC` – licznik główny, odczytywany przy wystąpieniu pierwszego i ostatniego zbocza (w milisekundach).

Zbocza sygnału wykrywane są w programie poprzez porównanie aktualnego i poprzedniego stanu pinu `P0.2`.

3.4.2 Inicjalizacja i odczyt

1. Konfiguracja pinu `PIO1_5` (P0.2) jako wejściowego.
2. System oczekuje na zbocze sygnału wejściowego, co inicjuje pomiar czasu.
3. Zliczanie `NUM_HALF_PERIODS` (tutaj 340) kolejnych zboczy sygnału z czujnika.
   (Licznik sprzętowy generuje przerwanie co 1 ms.)
4. Pomiar czasu Δt_{ms} – liczba milisekund, które upłynęły między pierwszym a ostatnim zboczem.

5. Przeliczenie czasu na temperaturę:

$$10 \cdot T(^\circ\mathrm{C}) = \frac{2 \cdot 1000 \cdot \Delta t_{\mathrm{ms}}}{\texttt{NUM\_HALF\_PERIODS} \cdot \texttt{TEMP\_SCALAR\_DIV10}} - 2731$$

   - Δt_{ms} – zmierzony czas (w ms) dla zadanej liczby półokresów.
   - `TEMP_SCALAR_DIV10` – współczynnik zależny od konfiguracji pinów `TS0`/`TS1` (tutaj: 1).
   - `NUM_HALF_PERIODS` – liczba półokresów poddanych pomiarowi (tutaj: 340).
   - 2731 – stała przeliczająca wynik z kelwinów na dziesiąte części stopnia Celsjusza.

6. Wykorzystanie odczytanych danych – temperatura wyświetlana jest na ekranie OLED.

3.5 Magistrala SPI

System wykorzystuje magistralę SPI z interfejsem SSP1 do komunikacji z wyświetlaczem OLED.

3.5.1 Piny SPI:

- **P0.7** - SCK1 (Serial Clock for SSP1); generowany przez mastera sygnał zegarowy służący do synchronizacji przesyłu danych.
- **P0.8** - MISO1 (Main In, Sub Out); służy do przesyłu danych od slave'a do mastera; SPI używamy tylko dla OLED więc MISO nie jest używany;
- **P0.9** - MOSI1 (Main Out, Sub In); służy do przesyłu danych od mastera do slave'a;
- **P2.2 - SSEL** (Slave Select); umożliwia wybór slave'a do transmisji, jest aktywowany stanem niskim; sterowany przez GPIO;

3.5.2 Rejestry magistrali SPI:

- **S0SPCR** - SPI Control Register; rejestr sterujący pracą interfejsu SPI, pozwala m.in. na ustawienie trybu pracy, włączenie SPI i przerwań.

Tabela 5: Opis bitów rejestru S0SPCR.

Bity	Symbol	Opis
0–1	—	Zarezerwowane.
2	BitEnable	0 – transmisja zawsze 8-bitowa, 1 – długość słowa określona przez bity 11:8 (`BITS`).
3	CPHA	Faza zegara: 0 – dane próbkowane na pierwszym zboczu, 1 – na drugim zboczu.
4	CPOL	Polaryzacja zegara: 0 – SCK aktywny wysoki, 1 – SCK aktywny niski.
5	MSTR	Tryb pracy SPI: 0 – tryb Slave, 1 – tryb Master.
6	LSBF	Kolejność bitów: 0 – MSB pierwszy, 1 – LSB pierwszy.
7	SPIE	Przerwania SPI: 0 – wyłączone, 1 – włączone (na zdarzenia SPIF lub MODF).
8–11	BITS	Liczba bitów przesyłanych w jednym transferze (aktywny tylko gdy BitEnable = 1). Patrz tabela poniżej.
12–31	—	Zarezerwowane.

1000	1001	1010	1011	1100	1101	1110	1111	0000
8	9	10	11	12	13	14	15	16

- **S0SPSR** - SPI Status Register; rejestr stanu, zawiera flagi informujące o gotowości transmisji, zakończeniu przesyłu i błędach.

Tabela 6: Opis bitów rejestru S0SPSR.

Bity	Symbol	Opis
0–2	—	Zarezerwowane.
3	ABRT	Przerwanie transmisji w trybie Slave: 1 – wystąpiło przerwanie; bit czyszczony przez odczytanie tego rejestru.
4	MODF	Błąd trybu: 1 – wystąpił błąd trybu. Bit czyszczony przez odczytanie tego rejestru i zapis do rejestru kontrolnego SPI.
5	ROVR	Nadpisanie przy odczycie: 1 – wystąpił błąd przepełnienia przy odbiorze. Bit czyszczony przez odczytanie tego rejestru.
6	WCOL	Kolizja zapisu: 1 – wystąpiła kolizja przy zapisie. Bit czyszczony przez odczytanie tego rejestru i dostęp do rejestru danych SPI.
7	SPIF	Flaga zakończenia transmisji SPI: 1 – zakończono transmisję. Dla Mastera: ustawiana na końcu ostatniego cyklu; dla Slave'a: przy ostatnim zboczu próbkującym SCK. Bit czyszczony przez odczytanie tego rejestru i dostęp do rejestru danych SPI.
8–31	—	Zarezerwowane.

- **S0SPDR** - SPI Data Register; dwukierunkowy rejestr danych, zapis do niego inicjuje transmisję, odczyt zwraca dane odebrane przez SPI.

Tabela 7: Opis bitów rejestru S0SPDR.

Bity	Symbol	Opis
0-7	DataLow	Dwukierunkowy port danych SPI. Służy do odczytu danych odebranych lub zapisu danych do wysłania. Wartość domyślna: `0x00`.
8–15	DataHigh	Dodatkowe bity transmisji/odbioru, gdy bit 2 rejestru `SPCR` (BitEnable) = 1 i bity 11:8 `SPCR` są różne od `1000` (czyli liczba bitów > 8). Gdy liczba bitów < 16, bardziej znaczące bity przy odczycie mają wartość zero. Wartość domyślna: `0x00`.
16–31	—	Zarezerwowane.

- **S0SPCCR** - SPI Clock Counter Register; ustala częstotliwość zegara SPI (SCK) poprzez wartość dzielnika.

Tabela 8: Opis bitów rejestru S0SPCCR.

Bity	Symbol	Opis
0–7	Counter	Ustawienie licznika zegara SPI0. Wartość musi być parzysta i większa lub równa 8. Określa częstotliwość zegara SPI: $$\mathrm{SCK}_{\mathrm{freq}} = \frac{PCLK}{\mathrm{Counter}}$$ Wartość domyślna: `0x00`.
8–31	—	Zarezerwowane.

- **S0SPINT** - SPI Interrupt Flag; flaga informująca o wystąpieniu przerwania SPI; może być programowo wyczyszczona.

Tabela 9: Opis bitów rejestru S0SPINT

Bity	Symbol	Opis
0	SPIF	Flaga przerwania SPI. Ustawiana przez interfejs SPI w celu wygenerowania przerwania. Zerowana przez zapisanie jedynki do tego bitu.
1–7	—	Zarezerwowane.
8–31	—	Zarezerwowane.

3.5.3 Rejestry SSP:

- **CR0** – Control Register 0; ustala rozmiar słowa danych, tryb protokołu SPI oraz współczynnik dzielnika zegara.

Tabela 10: Opis bitów rejestru CR0.

Bity	Symbol	Opis
0-3	DSS	Wybór długości słowa danych przesyłanych w jednej ramce (ang. *Data Size Select*). Wartości 0000–0010 są zarezerwowane i nieobsługiwane. Patrz tabela poniżej.
4-5	FRF	Format ramki: `00` – SPI, `01` – TI, `10` – Microwire, `11` – zarezerwowane (nie używać).
6	CPOL	Polaryzacja zegara: 0 – linia SCK niska między transmisjami, 1 – linia SCK wysoka między transmisjami.
7	CPHA	Faza zegara: 0 – dane próbkowane przy pierwszym zboczu, 1 – przy drugim zboczu sygnału zegarowego.
8-15	SCR	Ustawienie szybkości zegara (ang. *Serial Clock Rate*) – liczba cykli zegara preskalera na jeden bit - 1. Częstotliwość transmisji: $f = \frac{\text{PCLK}}{\text{CPSDVSR} \times (\text{SCR}+1)}$.
16-31	—	Zarezerwowane.

Tabela 11: Kodowanie pola `DSS` – długość słowa danych.

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

- **CR1** – Control Register 1; konfiguruje tryb master/slave i inne ustawienia pracy kontrolera SSP.

Tabela 12: Opis bitów rejestru CR1.

Bit	Symbol	Opis
0	LBM	Loop Back Mode: 0 – tryb normalny, 1 – sprzężenie zwrotne (dane wejściowe pochodzą z wyjścia).
1	SSE	SSP Enable: 0 – kontroler SSP wyłączony, 1 – kontroler SSP włączony (działa na magistrali SPI).
2	MS	Master/Slave Mode: 0 – tryb Master (SSP steruje SCLK, MOSI, SSEL), 1 – tryb Slave (SSP steruje tylko MISO). Może być ustawiony tylko gdy `SSE` = 0.
3	SOD	Slave Output Disable: 0 – wyjście MISO aktywne w trybie Slave, 1 – wyjście MISO zablokowane. Dotyczy tylko trybu Slave.
4-31	—	Zarezerwowane.

- **DR** – Data Register; rejestr danych – zapis wypełnia FIFO nadawcze, odczyt opróżnia FIFO odbiorcze.

Tabela 13: Opis bitów rejestru DR.

Bity	Symbol	Opis
0-15	DATA	Dane do wysłania lub odebrane dane.
		Zapis: Program może zapisać dane do rejestru, gdy bit `TNF` (Tx FIFO Not Full) w rejestrze statusu jest ustawiony na 1 (FIFO nadajnika nie jest pełne). Jeśli FIFO było puste i kontroler SSP nie jest zajęty, transmisja zaczyna się natychmiast. W przeciwnym wypadku dane zostaną wysłane po zakończeniu poprzednich. Jeśli długość danych jest mniejsza niż 16 bitów, dane powinny być wyjustowane do prawej strony (right-justified).
		Odczyt: Program może odczytać dane, gdy bit `RNE` (Rx FIFO Not Empty) w rejestrze statusu jest ustawiony na 1 (FIFO odbiornika nie jest puste). Odczyt zwraca najstarsze dane w FIFO, wyjustowane do prawej strony jeśli mniej niż 16 bitów.
16-31	—	Zarezerwowane.

- **SR** – Status Register; rejestr stanu – zawiera flagi informujące o stanie transmisji i buforów FIFO.

Tabela 14: Opis bitów rejestru SR.

Bit	Symbol	Opis
0	TFE	Transmit FIFO Empty — bit ustawiony na 1, gdy FIFO nadajnika jest puste, 0 gdy nie jest puste.
1	TNF	Transmit FIFO Not Full — bit ustawiony na 1, gdy FIFO nadajnika nie jest pełne, 0 gdy jest pełne.
2	RNE	Receive FIFO Not Empty — bit ustawiony na 1, gdy FIFO odbiornika nie jest puste, 0 gdy jest puste.
3	RFF	Receive FIFO Full — bit ustawiony na 1, gdy FIFO odbiornika jest pełne, 0 gdy nie jest pełne.
4	BSY	Busy — bit ustawiony na 1, gdy kontroler SSP jest zajęty (wysyła/odbiera ramkę lub FIFO nadajnika nie jest puste), 0 gdy jest bezczynny.
5-31	—	Zarezerwowane.

- **CPSR** – Clock Prescale Register; dzielnik preskalera zegara – ustala podstawową częstotliwość SSP.

Tabela 15: Opis bitów rejestru CPSR.

Bity	Symbol	Opis
0-7	CPSDVSR	Parzysta wartość z zakresu 2–254, przez którą dzielony jest sygnał zegara SSP_PCLK, aby uzyskać zegar preskalera. Bit 0 zawsze odczytywany jest jako 0 (rezerwuje parzystość wartości).
8-31	—	Zarezerwowane.

- **IMSC** – Interrupt Mask Set and Clear Register; umożliwia maskowanie (włączanie/wyłączanie) poszczególnych źródeł przerwań.

Tabela 16: Opis bitów rejestru IMSC.

Bit	Symbol	Opis
0	RORIM	Ustawienie tego bitu powoduje włączenie przerwania w przypadku przepełnienia odbiorczego FIFO (Receive Overrun), czyli gdy FIFO jest pełne, a kolejna ramka została całkowicie odebrana. W takim przypadku poprzednie dane w FIFO są nadpisywane.
1	RTIM	Włącza przerwanie w przypadku wystąpienia Receive Timeout, czyli gdy FIFO nie jest puste, ale dane nie były odczytane przez pewien czas określony jako okres timeout.
2	RXIM	Włącza przerwanie, gdy odbiorcze FIFO jest co najmniej w połowie zapełnione.
3	TXIM	Włącza przerwanie, gdy nadające FIFO jest co najmniej w połowie puste.
4-31	—	Zarezerwowane.

- **RIS** – Raw Interrupt Status Register; pokazuje status aktywnych przerwań niezależnie od ustawienia masek.

Tabela 17: Opis bitów rejestru RIS.

Bit	Symbol	Opis
0	RORRIS	Bit ustawiany na 1, gdy nowa ramka została całkowicie odebrana, podczas gdy Rx FIFO było pełne. Wtedy poprzednia ramka jest nadpisywana przez nową.
1	RTRIS	Bit ustawiany na 1, gdy Rx FIFO nie jest puste i nie zostało odczytane przez okres timeout (określony na podstawie częstotliwości SSP).
2	RXRIS	Bit ustawiany na 1, gdy Rx FIFO jest co najmniej w połowie zapełnione.
3	TXRIS	Bit ustawiany na 1, gdy Tx FIFO jest co najmniej w połowie puste.
4-31	—	Zarezerwowane.

- **MIS** – Masked Interrupt Status Register; pokazuje status przerwań uwzględniający ustawienia masek.

Tabela 18: Opis bitów rejestru MIS.

Bit	Symbol	Opis
0	RORMIS	Bit ustawiany na 1, jeśli nowa ramka została całkowicie odebrana podczas gdy Rx FIFO było pełne, i jeśli odpowiednia przerwanie jest włączone.
1	RTMIS	Bit ustawiany na 1, jeśli Rx FIFO nie jest puste, nie zostało odczytane przez okres timeout oraz przerwanie jest włączone.
2	RXMIS	Bit ustawiany na 1, jeśli Rx FIFO jest co najmniej w połowie zapełnione i przerwanie jest włączone.
3	TXMIS	Bit ustawiany na 1, jeśli Tx FIFO jest co najmniej w połowie puste i przerwanie jest włączone.
4-31	—	Zarezerwowane.

- **ICR** – Interrupt Clear Register (SSPICR); służy do ręcznego kasowania wybranych flag przerwań.

Tabela 19: Opis bitów rejestru ICR.

Bit	Symbol	Opis
0	RORIC	Zapisanie 1 do tego bitu czyści przerwanie „ramka została odebrana, gdy Rx FIFO było pełne”.
1	RTIC	Zapisanie 1 do tego bitu czyści przerwanie „Rx FIFO nie jest puste i nie zostało odczytane przez okres timeout”. Okres timeout jest taki sam w trybie master i slave i zależy od szybkości SSP: 32 bity przy PCLK / (CPSDVSR × [SCR+1]).
2-31	—	Zarezerwowane.

- **DMACR** – DMA Control Register; umożliwia włączenie transmisji danych za pomocą kontrolera DMA.

Tabela 20: Opis bitów rejestru DMACR.

Bit	Symbol	Opis
0	RXDMAE	Włącza DMA dla odbiorczego FIFO, gdy ustawiony na 1. Wyłącza DMA dla odbiorczego FIFO, gdy ustawiony na 0.
1	TXDMAE	Włącza DMA dla nadawczego FIFO, gdy ustawiony na 1. Wyłącza DMA dla nadawczego FIFO, gdy ustawiony na 0.
2-31	—	Zarezerwowane.

3.5.4 Inicjalizacja magistrali

1. Ustawienie pinów
   (a) P0.7 — SCK
   (b) P0.8 — MISO
   (c) P0.9 — MOSI
   (d) P2.2 — SSEL
2. Włączenie zasilania SSP1 poprzez ustawienie bitu 10 w rejestrze PCONP.
3. Konfiguracja ustawień magistrali (tryb pracy: master, format ramki: SPI, liczba bitów danych: 8, polaryzacja i faza zegara: CPOL - wysoki poziom, CPHA - pierwsza krawędź, prędkość zegara: 1 MHz).
4. Inicjalizacja rejestrów kontrolnych SSP (CR0 i CR1).
5. Ustawienie prędkości taktowania SSP.
6. Włączenie magistrali SSP ustawiając bit SSE w rejestrze CR1.

3.6 Wykorzystanie

1. Konfiguracja pinów SPI (PINSEL), ustawienie parametrów w strukturze SSP_CFG_Type, wywołanie SSP_Init() i SSP_Cmd() do uruchomienia magistrali.
2. Wysyłanie danych; ustawienie linii SSEL nisko (aktywacja slave), wpisanie danych do SSP_DR, oczekiwanie na zakończenie transmisji (BSY=0), odbiór danych równolegle.
3. Odbieranie danych; Wysłanie tzw. dummy bajtów, oczekiwanie na odebranie danych (RNE=1), odczyt z SSP_DR.
4. Zmiana i odczyt rejestrów slave; wysłanie adresu rejestru, wysłanie lub odebranie danych zgodnie z potrzebą.
5. Koniec transmisji; ustawienie linii SSEL na wysoki poziom.

3.7 Wyświetlacz OLED

Do wyświetlania informacji (interfejsu użytkownika) wykorzystano wyświetlacz OLED UG-9664HSWAG01 (oznaczenie na schemacie OLED1). Jest to monochromatyczny wyświetlacz o rozdzielczości 96x64 pikseli, sterowany za pomocą magistrali SPI. Wyświetlacz ten oparty jest o sterownik SSD1305 (zdolny do obsługiwania rozdzielczości 132x63 pikseli).

3.7.1 Konfiguracja

Piny sterujące wyświetlaczem OLED:

- **P0.6** – CS (Chip Select); aktywny w stanie niskim;
- **P2.7** – D/C (Data/Command); w stanie wysokim dane są traktowane jako dane, w stanie niskim przesyłane do rejestru komend.
- **P2.1** - RES (reset); w stanie niskim wywołuje reset i inicjalizację, w trakcie normalnej pracy należy utrzymywać go w stanie wysokim;

3.7.2 Inicjalizacja:

1. Ustawiamy piny 2.1, 2,7 oraz 0.6 w tryb wyjścia za pomocą rejestrów GPIO.
2. Ustawiamy pin 2.1 w tryb niskiego stanu aby upewnić się, że wyświetlacz jest wyłączony.
3. Wysyłamy sekwencję instrukcji inicjalizujących do wyświetlacza OLED.
4. Odczekujemy krótki czas przed włączeniem wyświetlacza.
5. Ustawiamy pin 2.1 w tryb wysokiego stanu aby włączyć wyświetlacz.

3.7.3 Przesyłanie danych:

1. Ustawienie pinu CS w stan niski.
2. Ustawienie pinu D/C w stan wysoki w trybie danych lub w stan niski w trybie komendy.
3. Przygotowanie struktury transferu.
4. Inicjacja transmisji SPI.
5. Ustawienie pinu CS w stan wysoki.

3.7.4 Rysowanie pojedynczego piksela

Pamięć wyświetlacza podzielona jest na 8 stron, z których każda zawiera 132 bajty (kolumny). Każdy bajt reprezentuje 8 pionowych pikseli. Aktywny obszar matrycy to 96x64 piksele, jednak z powodu nadmiarowych kolumn konieczne jest przesunięcie (offset) adresowania o 18 kolumn w poziomie.

1. Na podstawie współrzędnej y określany jest numer strony (ang. page address) — każda strona odpowiada 8 poziomym liniom pikseli w pamięci graficznej SSD1305.
2. Obliczany jest adres kolumny x, przy czym należy uwzględnić fakt, że kontroler SSD1305 obsługuje rozdzielczość 132x64 piksele, natomiast fizyczny wyświetlacz posiada tylko 96x64 piksele, co oznacza konieczność stosowania przesunięcia poziomego podczas adresowania kolumn.
3. Na podstawie współrzędnej y wyliczany jest numer bitu (0–7), który odpowiada konkretnej pozycji w bajcie strony, a następnie tworzona jest odpowiednia maska bitowa.
4. W buforze cieniowania wykonywana jest modyfikacja odpowiedniego bajtu: bit jest ustawiany lub czyszczony w zależności od wartości koloru (czarny/biały).
5. Ustawiany jest adres w pamięci wyświetlacza (komendy: wybór strony, kolumny niskiej i wysokiej).
6. Bajt danych z bufora cieniowania (zaktualizowany) jest przesyłany przez interfejs SPI do pamięci graficznej wyświetlacza tak jak opisano w 3.7.3.

3.7.5 Wyświetlanie pojedynczego znaku

1. Z tablicy font5x7 pobierany jest odpowiedni wzorzec bitowy znaku (bitmapa o wysokości 8 linii).
2. Dla każdej z 8 linii (wierszy) znaku:
   - Wczytywany jest bajt danych określający wzorzec piksela w danym wierszu.
   - Iteracja po 6 kolumnach piksela w danej linii:
     – Jeśli odpowiedni bit w bajcie danych jest ustawiony — wybierany jest kolor pierwszoplanowy, W przeciwnym razie — kolor tła).
     – Ustawienie danego piksela zgodnie z kolorem i współrzędnymi 3.7.4.
3. Po przerysowaniu całej linii znakowej kursor pionowy jest przesuwany w dół o jeden piksel, a poziomy resetowany do początkowej pozycji.

3.7.6 Wyświetlanie ciągu znaków, rysowanie prostokąta

Wyświetlenie ciągu znaków i rysowanie prostokąta realizowane są w oparciu o te same mechanizmy niskopoziomowe co rysowanie pojedynczego piksela, znaku, przy czym operacje te są zoptymalizowane pod kątem wydajności poprzez grupowanie zapisów do pamięci wyświetlacza.

W przypadku ciągów znaków każdy znak traktowany jest jako zestaw pikseli, a wypełnienie prostokąta sprowadza się do sekwencyjnego ustawiania bloków pikseli w określonym obszarze, co minimalizuje liczbę koniecznych operacji adresowania.

3.7.7 Czyszczenie ekranu

Czyszczenie ekranu polega na ustawieniu wszystkich strony (od 0xB0 do 0xB7) i wysyłaniu do każdej po 132 bajty zer (lub wartości 0xFF przy inwersji), w celu wyczyszczenia zawartości.

3.8 System audio

3.8.1 Wstęp

System audio inteligentnej donicy składa się z głośnika (oznaczenie na schemacie SP1) oraz układu sterowania LM4811MM (oznaczenie na schemacie U10). Układ ten umożliwia generowanie sygnałów dźwiękowych informujących użytkownika o stanie rośliny. Implementacja obejmuje inicjalizację układu sterowania, konfigurację parametrów dźwięku oraz bibliotekę funkcji do odtwarzania różnych melodii w zależności od poziomu wilgotności gleby.

3.8.2 Podstawy Teoretyczne

Generacja Dźwięku w Mikrokontrolerze Dźwięk powstaje poprzez przełączanie pinu GPIO w odpowiedniej częstotliwości, tworząc przebieg prostokątny (square wave).

gdzie T to czas stanu wysokiego/niskiego w mikrosekundach.

Wzmacniacz LM4811

- Sterowany cyfrowo (3 piny: CLK, UP/DN, SHDN).
- Zakres głośności: 64 kroki (sterowane impulsami CLK).
- Tryb shutdown (SHDN = 0 wyłącza wzmacniacz).

3.8.3 Implementacja

Konfiguracja Sprzętowa

Element	Pin LPC1768/9	Funkcja
LM4811 SIGNAL	P0.26	Generowanie sygnału dźwiękowego
LM4811 CLK	P0.27	Zegar zmiany głośności
LM4811 UP/DN	P0.28	Kierunek zmiany głośności
LM4811 SHDN	P2.13	Włączanie/wyłączanie wzmacniacza

W programie piny te są konfigurowane jako wyjścia przy pomocy funkcji `GPIO_SetDir()`:

```
GPIO_SetDir(0, 1<<27, 1); // CLK
GPIO_SetDir(0, 1<<28, 1); // UP/DN
GPIO_SetDir(2, 1<<13, 1); // SHDN
GPIO_SetDir(0, 1<<26, 1); // Square wave
```

Następnie są ustawiane w stan niski w celu inicjalizacji układu LM4811:

```
GPIO_ClearValue(0, 1<<27);
GPIO_ClearValue(0, 1<<28);
GPIO_ClearValue(2, 1<<13);
```

Odtwarzanie melodii Melodie odtwarzane są w zależności od poziomu wilgotności gleby. Odpowiednia funkcja 'playSong()' generuje na pinie P0.26 przebieg prostokątny o odpowiedniej częstotliwości i czasie trwania, co przekłada się na dźwięki słyszalne z głośnika. Dwie melodie są używane: „smutna" i „wesoła".

3.9 Timer

3.9.1 Wstęp

System wykorzystuje timery sprzętowe mikrokontrolera do precyzyjnego odmierzania czasu i wykonywania cyklicznych zadań. W przeciwieństwie do PWM, który służy do generowania sygnałów o zmiennym wypełnieniu, timery są wykorzystywane do wywoływania przerwań w określonych odstępach czasu (dokładniej o tym w sekcji PWM). Funkcjonalność ta obejmuje konfigurację preskalerów, rejestrów porównania oraz obsługę przerwań. Timery są używane do planowania pomiarów, kontroli cykli nawadniania oraz implementacji funkcji oszczędzania energii.

3.9.2 Rejestry Timer0

Działanie modułu Timer0 opiera się na zestawie rejestrów, które pozwalają na jego konfigurację oraz kontrolę. W poniższej tabeli przedstawiono podstawowe rejestry związane z Timerem0:

Tabela 21: Rejestry Timer0 mikrokontrolera LPC

Rejestr	Nazwa	Opis
T0IR	Interrupt Register	Rejestr przerwań — informuje o źródle przerwania (np. dopasowanie wartości MR0). Ustawienie bitu na 1 czyści przerwanie.
T0TCR	Timer Control Register	Steruje uruchamianiem (bit 0) i resetowaniem (bit 1) timera.
T0TC	Timer Counter	Licznik główny — zlicza taktowania zegara (po uwzględnieniu preskalera).
T0PR	Prescale Register	Wartość, do której zlicza preskaler. Po osiągnięciu tej wartości licznik T0TC jest zwiększany o 1.
T0PC	Prescale Counter	Licznik preskalera — inkrementowany z każdym taktem zegara, aż osiągnie wartość z T0PR.
T0MCR	Match Control Register	Określa zachowanie po dopasowaniu licznika T0TC do wartości rejestru porównania T0MRx (np. wygenerowanie przerwania, zatrzymanie, reset).
T0MR0-3	Match Registers 0–3	Rejestry porównania. Gdy wartość T0TC osiągnie wartość z T0MRx, wykonywane są akcje zdefiniowane w T0MCR.
T0EMR	External Match Register	Sterowanie pinami zewnętrznymi w reakcji na dopasowanie (jeśli aktywne).
T0CTCR	Count Control Register	Umożliwia przełączenie trybu zliczania: timer / licznik zewnętrzny / enkoder kwadraturowy.

3.9.3 Implementacja

W ramach implementacji zastosowano moduł TIMER0, należący do grupy czterech dostępnych timerów mikrokontrolera. W projekcie wykorzystano go do odmierzania opóźnień w dwóch wariantach — w milisekundach oraz mikrosekundach.

Funkcja `Timer0_Wait(uint32_t time)` Funkcja ta realizuje opóźnienie o określoną liczbę milisekund. Konfiguracja timera przebiega w kilku krokach:

- **Ustawienie preskalera**:
  - `PrescaleOption` ustawione na `TIM_PRESCALE_USVAL` — timer przelicza czas w mikrosekundach,
  - `PrescaleValue` ustawione na 1000 — rejestr TC zwiększa się co 1 ms.
- **Konfiguracja kanału porównania MR0** przy pomocy struktury `TIM_MATCHCFG_Type`:
  - `MatchChannel = 0` — użyto kanału 0,
  - `MatchValue = time` — timer porównuje wartość licznika z wartością parametru wejściowego,
  - `IntOnMatch = TRUE` — ustawienie flagi przerwania po dopasowaniu,
  - `ResetOnMatch = FALSE` — licznik nie resetuje się automatycznie,
  - `StopOnMatch = TRUE` — timer zatrzymuje się po dopasowaniu.
- **Uruchomienie timera**:

```
TIM_Init(LPC_TIM0, TIM_TIMER_MODE, &TIM_ConfigStruct);
TIM_ConfigMatch(LPC_TIM0, &TIM_MatchConfigStruct);
TIM_Cmd(LPC_TIM0, ENABLE);
```

- **Oczekiwanie na zakończenie odmierzania czasu**:

```
while (!(TIM_GetIntStatus(LPC_TIM0, 0)));
TIM_ClearIntPending(LPC_TIM0, 0);
```

  Funkcja aktywnie sprawdza flagę przerwania i kończy działanie po osiągnięciu ustawionego czasu.

Funkcja `Timer0_us_Wait(uint32_t time)` Funkcja ta działa analogicznie do `Timer0_Wait()`, lecz służy do tworzenia krótkich opóźnień rzędu mikrosekund.

Główne różnice w konfiguracji to:

- `PrescaleValue = 1` — co oznacza, że licznik TC zwiększa się co 1 µs.
- Pozostałe ustawienia są identyczne jak w funkcji `Timer0_Wait()` — timer zatrzymuje się po dopasowaniu, ustawiana jest flaga przerwania, a licznik nie jest resetowany automatycznie.

Dzięki zastosowaniu tych dwóch funkcji możliwe jest tworzenie zarówno długich, jak i bardzo krótkich opóźnień z wykorzystaniem jednego zasobu sprzętowego — timera TIMER0. Implementacja ta bazuje na bibliotece CMSIS, co upraszcza dostęp do rejestrów sprzętowych i zwiększa przenośność kodu.

3.9.4 Zastosowanie Timer0 w programie

Timer0 jest wykorzystywany w programie do realizacji opóźnień w dwóch głównych miejscach:

- **W funkcji głównej** `main()` – timer służy do opóźnienia kolejnych cykli pętli `while(1)`. Wykonanie funkcji `Timer0_Wait(200)` powoduje zatrzymanie programu na 200 ms, co stabilizuje częstotliwość odczytów z czujników i aktualizacji OLED.
- **W funkcji** `playNote()` – funkcja ta odpowiada za generowanie dźwięków przez odpowiednie ustawianie stanu pinu z częstotliwością zależną od wysokości dźwięku. Timer0 w trybie mikrosekundowym (`Timer0_us_Wait()`) pozwala dokładnie kontrolować czas trwania stanu wysokiego i niskiego, co przekłada się na częstotliwość generowanego sygnału. W przypadku nuty o wartości zero (pauza), stosowana jest funkcja `Timer0_Wait()` z parametrem w milisekundach.

Dzięki takiemu podejściu Timer0 pełni w systemie dwie ważne funkcje: zapewnia odpowiednie tempo działania głównej pętli programu oraz pozwala na precyzyjne generowanie dźwięków w oparciu o opóźnienia w mikrosekundach.

3.10 Modulacja szerokości impulsu (PWM)

Implementacja PWM umożliwia precyzyjne sterowanie elementami wykonawczymi wymagającymi sygnału analogowego, takimi jak silniki czy regulacja jasności diody LED. Funkcjonalność obejmuje konfigurację częstotliwości sygnału PWM, ustawianie wypełnienia impulsu (duty cycle) oraz dynamiczną zmianę parametrów w zależności od warunków środowiskowych. PWM jest wykorzystywane do regulacji intensywności świecenia diod LED w diodzie RGB LED3 [schemat]. W projekcie został wykorzystany PWM1.

3.10.1 Inicjalizacja PWM1

1. Konfigurację rozpoczynamy od włączenia zasilania. W tym celu ustawiamy 6 bit w rejestrze PCONP (tabela 46.)
2. Piny na których chcemy sterować impulsami to P2.0, P2.1, P2.2. Chcemy je ustawić jako PWM1.1, PWM1.2, PWM1.3, dlatego w rejestrze PINSEL4 ustawiamy bity 1:0, 3:2, 5:4 na 01 - funkcja PWM1.X
3. Opis rejestrów PWM1:

   (a) **IR** - rejestr ten służy do zarządzania przerwaniami. Przerwania
      i. **PWMMR0 Interrupt** - bit 0, flaga przerwania na kanale 0
      ii. **PWMMR1 Interrupt** - bit 1, flaga przerwania na kanale 1
      iii. **PWMMR2 Interrupt** - bit 2, flaga przerwania na kanale 2
      iv. **PWMMR3 Interrupt** - bit 3, flaga przerwania na kanale 3
      v. **PWMCAP0 Interrupt** - bit 4, flaga przerwania dla zerowego wejscia przechwytującego (capture input); capture oznacza przechwycenie aktualnej wartości licznika (timer'a), np. w momencie zbocza sygnału (narastającego lub opadającego) na pinie wejściowym. Umożliwia to np. pomiar czasu trwania impulsów, okresów czy częstotliwości.
      vi. **PWMCAP1 Interrupt** - bit 5, flaga przerwania dla pierwszego wejscia przechwytującego
      vii. Bity 7:6 - zarezerwowane
      viii. **PWMMR4 Interrupt** - bit 8, flaga przerwania na kanale 4
      ix. **PWMMR5 Interrupt** - bit 9, flaga przerwania na kanale 5
      x. **PWMMR6 Interrupt** - bit 10, flaga przerwania na kanale 6
      xi. Bity 31:11 - zarezerwowane

   (b) **PWM1TCR**
      i. **Counter Enable** - bit 0, włączanie i wyłączanie licznika timera oraz preskalera
      ii. **Counter Reset** - bit 1, resetowanie licznika timera, synchoronizacja z kolejnym zboczem narastającym PCLK, tak długo, aż wartość tego bitu będzie równa 1
      iii. Bit 2 - zarezerwowany
      iv. **PWM Enable** - gdy wartość bitu jest róna 1 - **tryb PWM**, licznik resetuje się do 1, aktywuje shadow registery.
      Gdy wartość bitu jest równa 0 - **tryb timer**, licznik resetuje się do 0, bez PWM - działą jak zwykły timer.
      v. Bit 4:31 - zarezerwowane

   (c) **PWM1CTCR** - rejestr służy do wyboru trybu licznika - timer lub jako licznik zliczający impulsy z zewnętrznego źródła.
      i. **Counter/Timer Mode** - bity 0:1, wartości i odpowiadające im działanie zostały opisane w poniższej tabeli.

Wartość	Opis
00	Timer mode: Timer Counter (TC) wykorzystuje Prescale Counter (PC) oraz Prescale Register (PR), aby móc działać wolniej. TC inkrementuje się gdy PC = PR
01	Counter Mode: TC zlicza impulsy przychodzące (inkrementacja przy każdym zboczu narastającym) z zewnętrznego pinu wybranego na bitach 3:2
10	Counter Mode: TC zlicza impulsy przychodzące (inkrementacja przy każdym zboczu opadającym) z zewnętrznego pinu wybranego na bitach 3:2
11	Counter Mode: TC zlicza impulsy przychodzące (inkrementacja przy na obu zboczach impulsu) z zewnętrznego pinu wybranego na bitach 3:2

Tabela 22: Rodzaje pracy w trybie Timer/Counter

ii. **Count Input Select** - bity 3:2 - ta częśćrejestru odpowiada za wybó pinu PCAP z którego będą zliczane impulsy.

Wartość	Opis
00	Timer mode: Timer Counter (TC) wykorzystuje Prescale Counter (PC) oraz Prescale Register (PR), aby móc działać wolniej. TC inkrementuje się gdy PC = PR
01	Counter Mode: TC zlicza impulsy przychodzące (inkrementacja przy każdym zboczu narastającym) z zewnętrznego pinu wybranego na bitach 3:2
10	Counter Mode: TC zlicza impulsy przychodzące (inkrementacja przy każdym zboczu opadającym) z zewnętrznego pinu wybranego na bitach 3:2
11	Counter Mode: TC zlicza impulsy przychodzące (inkrementacja przy na obu zboczach impulsu) z zewnętrznego pinu wybranego na bitach 3:2

Tabela 23: Rodzaje pracy w trybie Timer/Counter

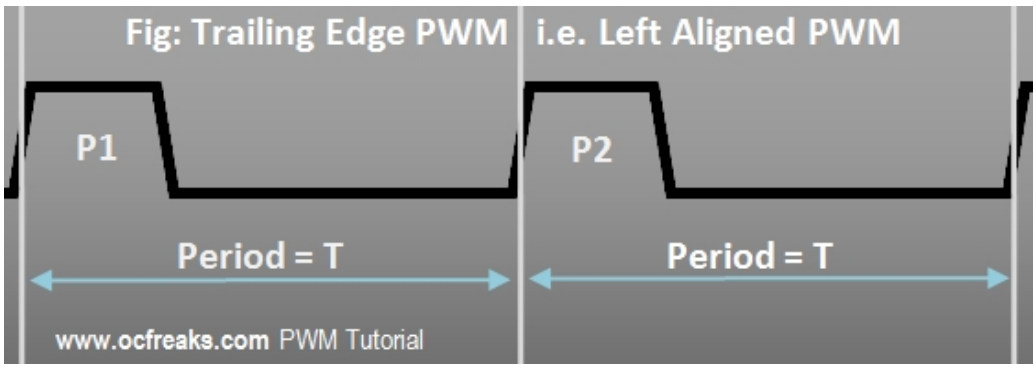
iii. **PWM1MCR** - steruje reakcjami układu PWM, gdy zawartość rejestru licznika czasowego PWM Timer Counter (TC) jest równa jednej z wartości w rejestrach dopasowania PWM Match Registers (MR0–MR6). Ustawienie jedynki na poniższych bitach powoduje włączenie danej funkcji.
   A. **PWMMR0I** - bit 0, gdy rejestr licznika PWM (PWM Timer Counter) osiągnie wartość ustawioną w Match Register 0 – PWMMR0, może wystąpić przerwanie - jeśli wartość ustawiona na tym bicie to 1
   B. **PWMMR0R** - bit 1, gdy zawartość licznika PWMTC jest równa wartości PWMMR0 nastąpi reset PWMTC; Pozwala to tworzyć cykliczny sygnał PWM – licznik odlicza od 0 do wartości w PWMMR0, potem reset.
   C. **PWMMR0S** - bit 3, zatrzymuje licznik, gdy wartość w rejestrze PWMTC jest równa wartości zapisanej w PWMMR0
   D. funkcjonalności kolejnych bitów tego rejestru są cyklicznie powtarzane dla PWMMR1f, PWMMR2f, PWMMR3f, PWMMR4f, PWMMR5f, PWMMR6f (gdzie 'f' oznacza numer rejestru PWMMR)
   Bity 21:31 są zarezerwowane.

iv. **PWM1CCR** - rejestr pozwala skonfigurować zdarzenia przechwytywania (capture events) na wejściach CAPn.x, czyli kiedy wartość licznika zostanie zapisana do rejestru przechwytywania (CRn), gdzie n jest numerem Timera - 0 lub 1. Warto zaznaczyć, że jeśli CAP input jest skonfigurowane jako źródło zliczania (Counter Mode w CTCR), to jego bity w PWM1CCR muszą mieć wartość 000. Poniższe funkcje są dostępne przy ustawieniu flagi na 1 we wskazanym bicie.
   A. **Capture on CAPn.0 rising edge** - bit 0, jeśli zostanie wykryte zbocze narastające, to do rejestru przechwytywania CR0 zostanie wpisana zawartość TC
   B. **Capture on CAPn.0 falling edge** - bit 1, jeśli zostanie wykryte zbocze opadające, to do rejestru przechwytywania CR0 zostanie wpisana zawartość TC
   C. **Interrupt on CAPn.0 event** - bit 3, jeśli zostanie wykryte zdarzenie przechwycenia, to zawartośćTC zostanie wpisana do CR0 oraz zostanie wygenerowane przewanie
   D. Takie same funkcje mają bity 5:3, ale dla CAPn.1 i zapis do CR1.
   E. Bity 31:6 są zarezerwowane.

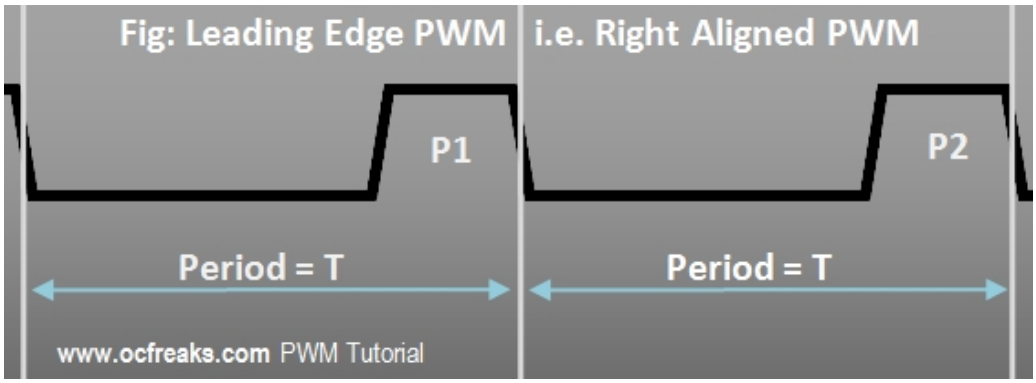
v. **PWM1PCR** - rejestr służy do włączania PWM oraz wybierania trybu działąnia kanału.
   A. **PWMSELx** - wybór trybu pracy dla kanału PWMx, gdzie x to numery 2-6.
   Poniższa tabela opisuje generowany impuls w zależności od ustawionego bitu.
   Bity 0:1 sąnieużywane, zawsze mają wartość 0.
   Bity 7:8 - są zarezerwowane.
   Zatem bity ustawiające tryby pracy to 6:2.

Wartość	Tryb	Opis
0	Single-edge	Generowany jest impuls z jedną krawędzią opadającą, ponieważ z dokumentacji (strona 520)., wynika iż każdy cykl PWM zaczyna się od stanu wysokiego.
1	Double-edge	Generowany jest impuls z dwiema krawędziami (umożliwia symetryczne impulsy)

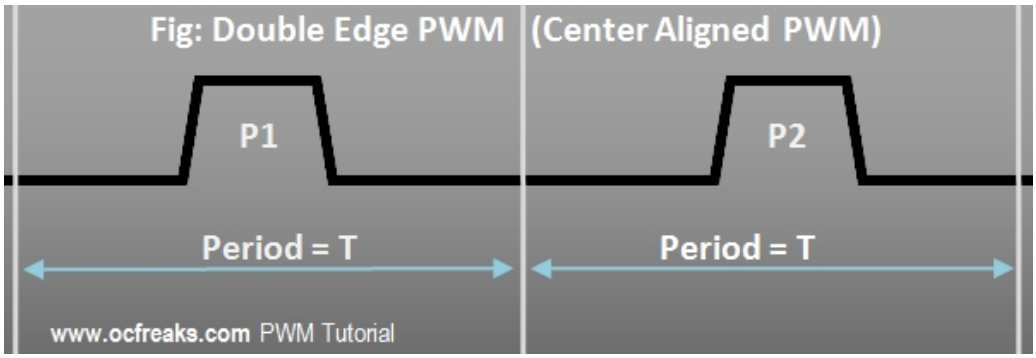
Tabela 24: Rodzaje pracy w trybie Timer/Counter



Rysunek 2: Przedstawienie trybu single-edge PWM, lewy impuls



Rysunek 3: Przedstawienie trybu single-edge PWM, prawy impuls

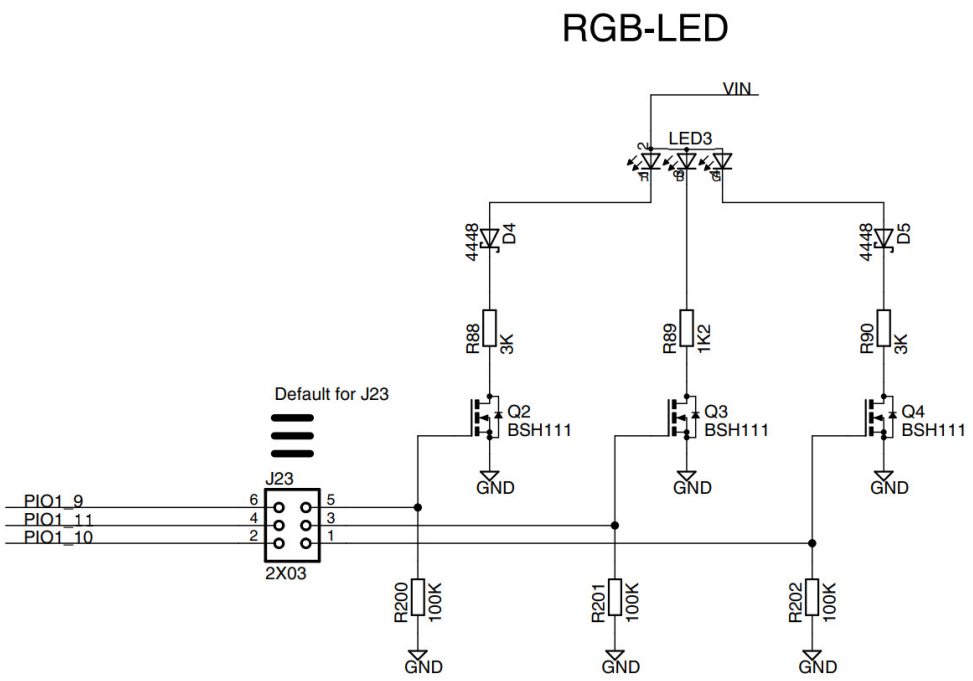


Rysunek 4: Przedstawienie trybu double-edge PWM

B. **PWMENAx** - służy do włączania lub wyłączania wyjścia PWM, gdzie x to numer kanału PWM (1-6). Ustawianie tych funkcji odbywa się na bitach 14:9. Wartość 1 włącza wyjście, a wartość 0 wyłącza. Bity 15:31 są nieużywane i mają wartość 0.

C. **PWM1LER** - używany jest do aktualizacji rejestró PWM Match, ponieważ w trybie PWM Mode, wartości wpisywane do rejestrów dopasowania nie są od razu używane, ale trafiają do rejestrów tymczasowych (shadow registers). Zmiany należy zatwierdzić poprzez ustawienie odpowiednich bitó w PWM1LER. Zmiany są widoczne dopiero po rozpoczęciu nowego cyklu PWM.

4. Omówienie zagadnienia PWM na podstawie sterowania diodą LED3.
LED3, to dioda RGB, składająca się z oddzielnych diod czerwonej, zielonej i niebieskiej (wspólny plus). Wykorzystanie PWM do pośredniego (sterujemy masą poprzez tranzystory (schemat)) sterowania jasnością diod RGB umożliwia dostosowanie jasności poszczególnych kolorów w zależności od potrzeb.



Rysunek 5: Schemat diody RGB

Jasność diody jest skorelowana z szerokością impulsu. Ze względu na zapis szestnastkowy kolorów RGB (każdy kolor ma zakres 0-255), zdecydowaliśmy się na okres 255.

```
LPC_PWM1->PCR = (1 << 9) | (1 << 10) | (1 << 11);  // ustawiamy tryb PWM na single-edge

LPC_PWM1->PR = 0;          // TC jest inkrementowany przy PR+1, wiec dzielnik zegara sie
    nie zmienia.
                           // Czestotliwosc wynosi 25 MHz / 1 = 25 MHz
LPC_PWM1->MR0 = 255;       // Okres PWM = 255 cykli
LPC_PWM1->MCR = (1 << 1); // Reset na MR0 (PWM cycle)

LPC_PWM1->MR1 = 255;       // Ustawiamy jasnosc na 100%, szerokosc impulsow = 255 / 255 =
    100%
LPC_PWM1->MR2 = 255;
LPC_PWM1->MR3 = 255;

LPC_PWM1->LER = (1 << 1) | (1 << 2) | (1 << 3) | (1 << 0); // Aktualizacja rejestrow MRx

LPC_PWM1->TCR = (1 << 0) | (1 << 3);  // Wlaczamy PWM
```

Kolor LED3 ustawiamy przy pomocy funkcji $PWM_SetColor(uint8_t red, uint8_t green, uint8_t blue)$, która zmienia zawartość rejestrów MR1, MR2 i MR3.

```
void PWM_SetColor(uint8_t red, uint8_t green, uint8_t blue) {
    if( red < 0 || red > 255 ) return; // ograniczenie zakresu
    if( green < 0 || green > 255 ) return;
    if( blue < 0 || red > 255 ) return;

    LPC_PWM1->MR1 = red; // ustawiamy szerokosc impulsu dla diody czerwonej
    LPC_PWM1->MR2 = blue; // ustawiamy szerokosc impulsu dla diody niebieskiej
    LPC_PWM1->MR3 = green; // ustawiamy szerokosc impulsu dla diody zielonej

    LPC_PWM1->LER = (1 << 1) | (1 << 2) | (1 << 3);  // zaktualizuj MR1-MR3
}
```

3.11 Przetwornik analogowo-cyfrowy (ADC)

System wykorzystuje przetwornik analogowo-cyfrowy do odczytu analogowego wejścia czujnika określanego jako "czujnik wilgotności". Należy zaznaczyć, że termin "wilgotność"jest w tym przypadku skrótem myślowym, ponieważ w rzeczywistości nie mierzymy bezpośrednio wilgotności gleby, a jej przewodność elektryczną, która jest skorelowana z zawartością wody w glebie. Implementacja obejmuje: Wbudowany przetwornik ADC jest przetwornikiem 12-bitowym, co oznacza, że może zwracać wartość z zakresu 0-4095. Przetwornik ten działa

przy użyciu metody SAR (Successive Approximation Register), która polega na sekwencyjnym przybliżaniu wartości.

1. **Inicjalizacja i konfiguracja przetwornika ADC** - Proces inicjalizacji obejmuje konfigurację rejestrów mikrokontrolera, ustawienie częstotliwości próbkowania, wybór kanału pomiarowego oraz określenie napięcia referencyjnego. ADC może wykonać do 200 000 konwersji na sekundę (200 kHz), przy pełnej 12-bitowej rozdzielczości. To znaczy, że maksymalnie co 5 mikrosekund może być gotowy nowy wynik pomiaru.
2. **Odczyt wartości analogowej** - Funkcjonalność ta odpowiada za uruchomienie konwersji analogowo-cyfrowej, odczyt wyniku konwersji oraz obsługę ewentualnych błędów pomiaru.
3. **Interpretacja danych pomiarowych** - System przetwarza odczytane wartości na względny poziom wilgotności gleby. Proces ten obejmuje kalibrację czujnika (określenie wartości dla suchej i mokrej gleby), filtrowanie zakłóceń oraz konwersję surowych danych na format zrozumiały dla użytkownika.

3.11.1 Inicjalizacja przetwornika ADC

1. Na początku ustawiono piny do pracy w trybie analogowym. Dla pinu 23 na porcie 0 ustawiono funkcję nr. 1, co jest odpowiednikiem funkcji analogowej (Tabela 81.)
   Aby ustawić wybraną funkcję, należy ustawić bity 15:14 w rejestrze PINSEL1 na 01.
2. Następnie ustawiono 12. bit w rejestrze PCONP o adresie 0x400F C0C4 na 1, aby włączyć zasilanie dla przetwornika ADC.
3. Kolejnym krokiem jest odczytanie dzielnika zapisanego w rejestrze PCLKSEL0 o adresie 0x400F C1A8 na bitach 25:24 (Tabela 40.). Domyślnie dzielnik ten wynosi 4 (w rejestrze pod podanymi bitami znajduje się 00). Wtedy obowiązuje wzór na PCLK:

$$PCLK = \frac{CCLK}{4}$$

Gdzie:

$$\text{CCLK} - \text{takt zegara układu;} \quad CCLK = 100\,\text{MHz}$$
$$\text{PCLK} - \text{takt zegara przetwornika ADC}$$

3.11.2 Obliczanie masek do odczytu wartości rejestrów

Aby móc zmieniać stan przetwarzania potrzebna jest maska, która pozwoli wybrać tylko interesujące nas bity z rejestru AD0CR.

1. W rejestrze AD0CR (0x4003 4000) na bicie 21 znajduje się informacja o stanie przetwornika:
   - 0 - przetwornik jest gotowy do pracy;
   - 1 - przetwornik jest wyłączony
2. Zatem potrzebujemy maski (1«21)

3.11.3 Ustawianie wartości AD0CR

W rejestrze AD0CR (0x4003 4000) na bitach 15:8 znajduje się wartość przez jaką dzielony jest PCLK, by uzyskać częstotliwość próbkowania. Zatem potrzebujemy maski

$$(0xCLKDIV << 8)OR(1 << 21)$$

Gdzie:

$$\text{CLKDIV} = \text{rate} \cdot 65$$
$$\text{CLKDIV} = \frac{2 \cdot PCLK + \text{CLKDIV}}{2 \cdot \text{CLKDIV}} - 1$$
$$\text{rate} = 0{,}2\,\text{MHz}$$

Aby ustawić wartość AD0CR, należy wykonać następujące kroki:

1. Wczytaj aktualną wartość rejestru AD0CR (0x4003 4000) do zmiennej.
2. Wykonaj operację OR na tej zmiennej oraz masce.
3. Zapisz zmienną z wynikiem do rejestru AD0CR

W opisywanym projekcie przetwornik ADC nie korzysta z przerwań.

3.11.4 Przeliczanie odczytu na dane liczbowe

Jak napisano na początku podrozdziału, przetwornik ADC korzysta z metody SAR (Successive Approximation Register). SAR jest realizowany sprzętowo. Sposób działania przetwornika jest następujący:
Dla uproszczenia przyjmujemy, że ADC ma 3 bity. Zakres wartości cyfrowych to 0-7. W rzeczywistości przetwornik ma 12 bitów.
Zakładamy, że:

$$V_{\text{ref}} = 3.3\,\text{V}$$
$$V_{\text{in}} = 2.0\,\text{V}$$

Każdy krok, to

$$\frac{3.3\,\text{V}}{8} = 0.4125\,\text{V}$$

1. Ustawiamy najbardziej znaczący bit na 1, więc liczba wynoch 100.
2. Następnie sprawdzamy, czy:

$$V_{\text{in}} \geq \frac{4}{8} \cdot V_{\text{ref}}$$
$$2.0\,\text{V} \geq 1.65\,\text{V}$$

3. Tak, więc zachowujemy najstarszy bit i ustawiamy kolejny bit na 1, czyli 110.
4. Następnie sprawdzamy, czy:

$$V_{\text{in}} \geq \frac{6}{8} \cdot V_{\text{ref}}$$
$$2.0\,\text{V} \geq 2.475\,\text{V}$$

5. Nie, więc zmieniamy bit na 0, czyli 101. Ustawiamy kolejny bit na 1, czyli 101.
6. Ponownie sprawdzamy, czy:

$$V_{\text{in}} \geq \frac{5}{8} \cdot V_{\text{ref}}$$
$$2.0\,\text{V} \geq 2.06\,\text{V}$$

7. Nie, więc zmieniamy bit na 0, czyli 100. Wartość cyfrowa wynosi 100, czyli 4 w systemie dziesiętnym, co odpowiada lekko ponad połowie zakresu odczytu ($\frac{4}{7} \approx 0.57$, a stosunek $\frac{V_{\text{in}}}{V_{\text{ref}}} \approx 0.60$).

Uzyskany pomiar służy do określenia wilgotności gleby i wypisanie na wyświtlacz odpowiedniego komunikatu, zostało to opisane w sekcji Interpretacja wskaźników.

4 Analiza awarii

Tabela 25: Analiza FMEA/FMECA

Ryzyko	Szansa	Znaczenie	Wykrywalność	Reakcja	Iloczyn
Błąd odczytu z czujnika wilgotności gleby (ADC)	0.3	7	Brak	Brak	10.5
Brak komunikacji z czujnikiem światła (I2C)	0.2	7	Brak	Brak	5.6
Uszkodzenie wyświetlacza OLED (SPI)	0.1	8	Brak	Brak	2.4
Uszkodzenie czujnika temperatury	0.1	7	Brak	Brak	4.2
Zawieszenie PWM (LED, audio)	0.2	3	Brak	Brak	2.4
Timer systemowy zatrzymany / błędny	0.5	9	Brak	Brak	22.5
Zanik zasilania	0.4	10	Brak	Brak	12.0
Zalanie układu	0.8	10	Brak	Brak	16.0

4.1 Analiza potencjalnych awarii

4.1.1 ADC (czujnik wilgotności)

Błąd odczytu z czujnika wilgotności gleby powoduje utratę informacji o potrzebie podlania rośliny. Awaria może wynikać z nieprawidłowego działania przetwornika analogowo-cyfrowego, zwarcia, przerwy lub degradacji czujnika.

4.1.2 I2C

Awaria zaburzy komunikację pomiędzy mikrokontrolerem a czujnikiem światła. Brak dostępu do danych oświetlenia uniemożliwi systemowi optymalną regulację doświetlenia rośliny, a także ograniczy możliwość reagowania na zmienne warunki zewnętrzne. W konsekwencji system może nie wykrywać braku światła, co może zaszkodzić roślinie w dłuższym czasie.

4.1.3 SPI / OLED

Uszkodzenie interfejsu SPI lub wyświetlacza OLED powoduje brak wyświetlania informacji dla użytkownika. Komunikaty o stanie systemu, wilgotności, temperaturze czy błędach nie będą prezentowane za pomocą wizualnego interfejsu, pozostaną jednak informacje przekazywane za pomocą diody i systemu audio.

4.1.4 Czujnik temperatury

Przegrzanie lub uszkodzenie czujnika temperatury prowadzi do odczytów poza zakresem lub całkowitego ich braku. Może to skutkować brakiem reakcji na niekorzystne warunki otoczenia.

4.1.5 PWM

Zawieszenie sterowania PWM skutkuje brakiem możliwości sterowania diodami LED. Użytkownik jest informowany jednak również za pomocą komunikatów na wyświetlaczu OLED więc to ryzyko jest niegroźne.

4.1.6 Timer systemowy

Zatrzymanie lub błędne działanie głównego timera zakłóca działanie logiki czasowej systemu.

4.1.7 Zanik zasilania

Zanik zasilania skutkuje pełnym zatrzymaniem działania donicy.

4.1.8 Zalanie wodą

Jedynie czujnik wilgotności gleby jest przeznaczony do pracy w kontakcie z wodą. Zalanie innych komponentów systemu może doprowadzić do ich trwałego uszkodzenia, zwarć lub nieprzewidywalnych stanów pracy.

- Wykrycie
  - System donicy nie posiada aktywnych czujników zalania.
- Reakcja
  - Zabezpieczenia powinny mieć charakter pasywny, użytkownik powinien zadbać aby tylko czujnik wilgotności gleby miał bezpośredni kontakt z wodą