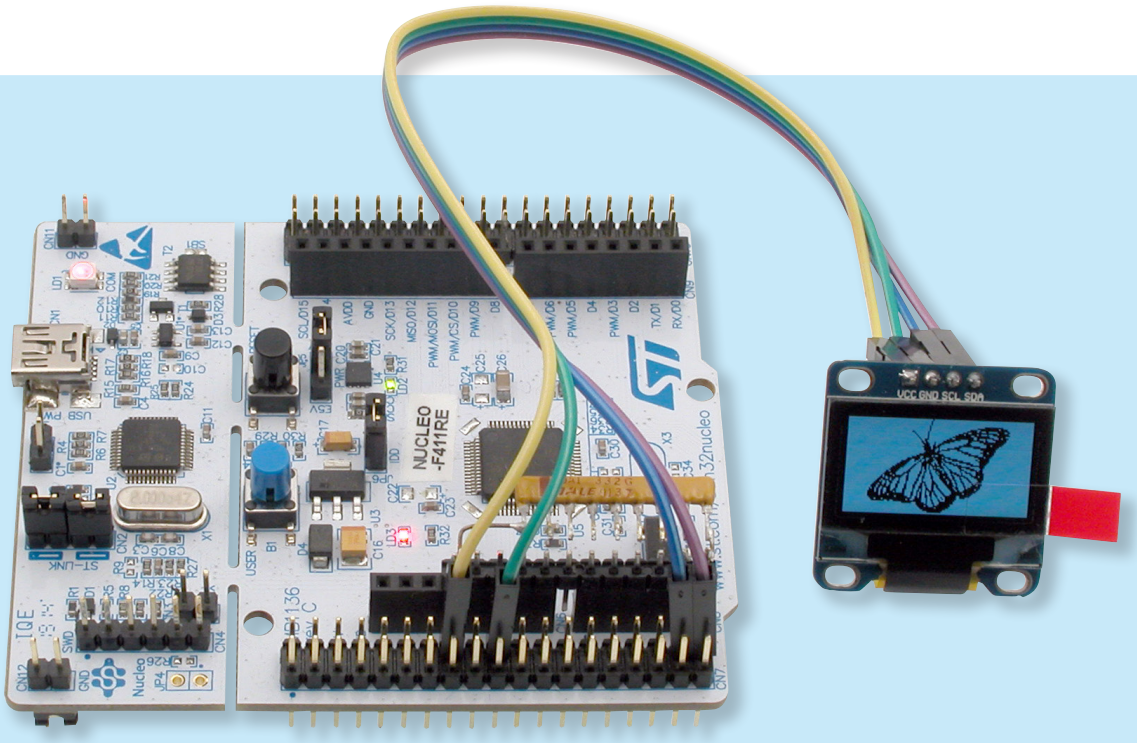


Nucleo i wyświetlacz OLED

Sterowanie wyświetlaczem OLED z kontrolerem SSD1306 za pomocą STM32

Doskonałym rozszerzeniem płytki Nucleo może być wyświetlacz OLED. Przyda się on do realizacji interfejsu użytkownika w budowanych urządzaniach lub po prostu do prezentowania wyników pracy programu. W artykule opisano sposób dołączenia i sterowania takiego wyświetlacza.

Użyty w tym przykładzie graficzny wyświetlacz OLED ze sterownikiem SSD1306 jest niewielki, ponieważ ma wymiary jedynie 27 mm×27 mm×3,5 mm. Ma rozdzielczość 128×64 piksele i wyświetla obrazy jednokolorowe, składające się z zaświeconych i zgaszonych punktów. Może być używany do prezentowania fotografii, wykresów, symboli oraz informacji tekstowych. Dzięki zasilaniu napięciem 3,3 V i interfejsowi komunikacyjnemu I²C doskonale nadaje się do bezpośredniego dołączenia, i rozszerzenia możliwości płytki startowej. Z kolei, płytki Nucleo z mikrokontrolerami STM32F z wbudowanym programatorem i zestawem złącz, w tym zgodnych ze standardem Arduino, stanowią wygodną bazę do budowania urządzeń prototypowych i amatorskich.

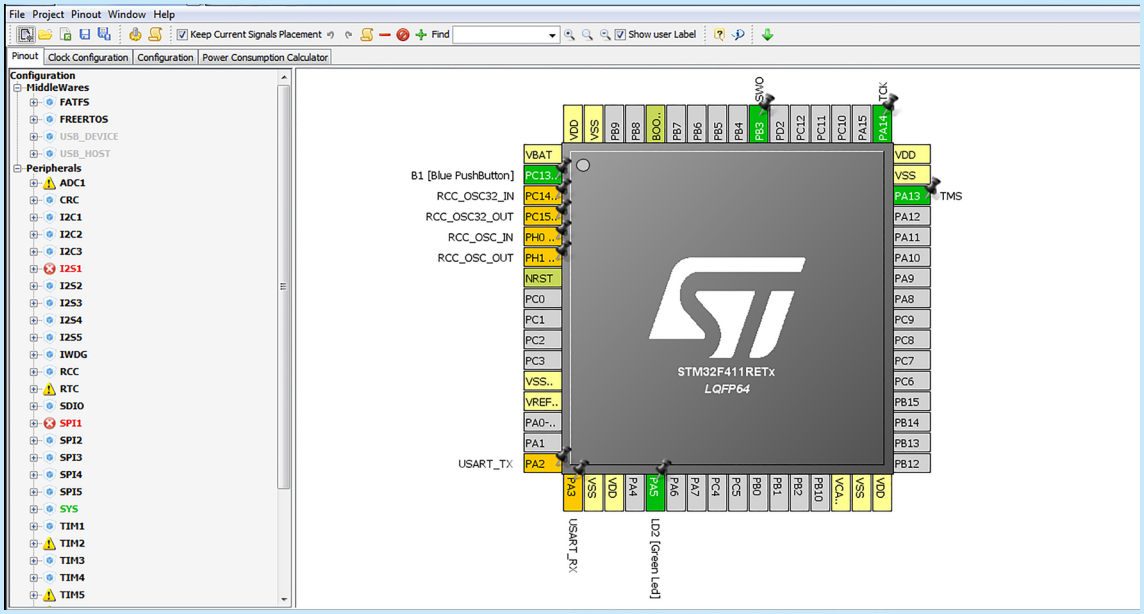
Dodatkową zaletą płytek Nucleo jest szybkość i łatwość tworzenia dla nich oprogramowania. Jest to możliwe dzięki oprogramowaniu narzędziowemu **STM32CubeMX**, które pozwala na błyskawiczne wygenerowanie szkieletu aplikacji. Należy tylko wskazać wybrany typ mikrokontrolera STM32F lub płytki ewaluacyjnej. Wygenerowany projekt jest oparty o aktualną wersję interfejsu HAL. Do stworzonego w taki sposób zalążka aplikacji użytkownik może dodawać własne funkcje.

Kolejnym elementem układanki jest środowisko programistyczne. Od pewnego czasu firma ST udostępnia darmowe środowisko o kodowej nazwie *AC6 System Workbench* przeznaczone dla produkowanych przez nią mikrokontrolerów STM32F. Jest to kompletny pakiet programistyczny (IDE, kompilator, debuger) wykonany na bazie *Eclipse*. W najnowszej wersji konfigurator *STM32CubeMX* jest w stanie tworzyć szkielety projektów, które w dalszej kolejności mogą być otwierane, edytowane i uruchamiane za pomocą *AC6 Workbench*.

Przykładowy projekt

Do wykonania przykładowego projektu wybrano następujący sprzęt i oprogramowanie:

- Wyświetlacz OLED o rozdzielczości 128×64 piksele, biały, ze sterownikiem SSD1306 i interfejsem I²C. Dodatkowe pliki z informacjami o wyświetlaczu można pobrać ze strony *https://goo.gl/9MSp01*.
- Pytka startowa Nucleo typu *STM32F411RET6 64 PINS*. Dodatkowe informacje o płytkach Nucleo są dostępne na stronie internetowej *http://goo.gl/xnNBNt*.



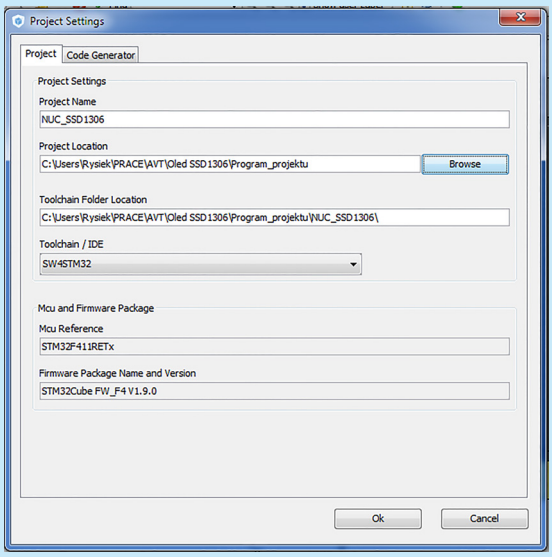
Rysunek 1. Okno programu STM32CubeMX po wybraniu mikrokontrolera

- Oprogramowanie **STM32CubeMX** w wersji 4.11.0 lub nowszej, do pobrania spod adresu *http://goo.gl/8TAepY*.
- Pakiet kompilatora **AC6 System Workbench** (**SW4STM32**) do pobrania ze strony internetowej *http://goo.gl/BjDpaj*. Pakiet jest darmowy, ale przed pobraniem należy się zarejestrować.
- Dokumentacja biblioteki HAL (*UM1884: Description of STM32L4 HAL and Low-layer drivers*) do pobrania spod adresu *http://goo.gl/GnsFYo*.

Mając pobrane i zainstalowane programy możemy przystąpić do wykonania aplikacji przykładowej.

Pierwszy krok: wygenerowanie szkieletu oprogramowania

Aby doprowadzić do wygenerowania szkieletu oprogramowania trzeba wykonać kilka nieskomplikowanych kroków. Po wybraniu opcji *New Project* należy wskazać typ mikrokontrolera lub płytki, dla której projekt ma być utworzony: *Board Selector* → *Type of Board: Ncleo64* → *NUCLEO-F411RE*. Po chwili powinien być wyświetlony pulpit podobny do pokazanego na **rysunku 1**. Na rysunku obudowy kontrolera już są zaznaczone wyprowadzenia używane przez zamontowane na płytce elementy. Są to między innymi nóżki służące do przyłączenia rezonatorów, przycisku i diody LED.



Rysunek 2. Ustawienia środowiska AC6 dla przykładowego projektu

Następnie, po wybraniu *Project* → *Settings* należy podać główne ustawienia projektu:

- *Project Name* (nazwę projektu).
- *Project Location* (ścieżkę dostępu do katalogu, w którym zostanie zapisany wygenerowany projekt; w tym przykładzie użyto katalogu *Program_projektu*).
- *Tolchain/IDE* (wybrany pakiet kompilatora, dla którego ma być wygenerowany projekt; w tym przykładzie SW4STM32.

Na **rysunku 2** pokazano ustawienia środowiska AC6 dla przykładowego projektu NUC_ SSD1306. Szkielet programu zostanie wygenerowany i zapamiętany we wskazanym katalogu po wybraniu *Project* → *Generate Code*.

Krok drugi: otwarcie projektu w środowisku AC6

Sposób importowania projektu utworzonego przez *STM-32CubeMX* do środowiska programistycznego AC6 System Workbench opisuje dokument na stronie *http://goo.gl/6kQOtI*, do którego prowadzi link *Importing a STCubeMX generated project* (dostęp do linku i dokumentu dopiero po zalogowaniu). Na podstawie tego dokumentu procedura importowania może wyglądać następująco:

- Otworzyć *AC6 System Workbench for STM32* podając w *Workspace Launcher* ścieżkę dostępu do katalogu, w którym znajduje się utworzony szkielet oprogramowania (w przykładzie będzie to katalog *Program_projektu*).
- Następnie należy wybrać opcję *File* → *Import* → *General* → *Existing Projects into Workspace* → *Next*.
- Podać ścieżkę dostępu do projektu ((dla tego przykładu katalog *Program_projektu*), zaznaczyć na wyświetlonej liście nazwę projektu (w przykładzie *NUC_SSD1306 Configuration(…)*).

Listing 1. Program przykładowy: testowanie stanu przycisku i zaświecenie diody LED

```
int main(void)
{
    /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
    HAL_Init();
    /* Configure the system clock */
    SystemClock_Config();
    /* Initialize all configured peripherals */
    MX_GPIO_Init();

    while (1)
    {
    /* USER CODE END WHILE */
    //kod testujący naciśnięcie niebieskiego przycisku na płytce
        if (HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13) ==GPIO_PIN_RESET)
        {
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);
        }
        else
        {
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);
        }
    }
}
```

Tabela 1. Połączenia pomiędzy wyświetlaczem a płytką Nucleo

Złącze wyświetlacza	Wyprowadzenie mikrokontrolera	Gniazdo „Morpho”	Gniazdo „Arduino”	Opis połączenia
SDA	PC0	CN7-38		Linia SDA I²C
SCL	PC1	CN7-36		Linia SCL I²C
GND		CN7-20		Masa
3,3V		CN7-16		Zasilanie 3,3 V
			CN8-6	Opornik podciągający linię SDA do +3,3 V
			CN8-5	Opornik podciągający linię SCL do +3,3 V

Pozostałe pola powinny pozostać niezaznaczone, jak na **rysunku 3**. Nacisnąć przycisk *Finish*.

Ponieważ wygenerowany został tylko szkielet oprogramowania, to po jego skompilowaniu i zaprogramowaniu kontrolera zamontowanego na płytce Nucleo nic się nie będzie działo, bo program krąży w nieskończonej, pustej pętli. Dla wstępnego testu płytki można w automatycznie wygenerowanym pliku *main.c* dopisać kod przedstawiony na **listingu 1**.

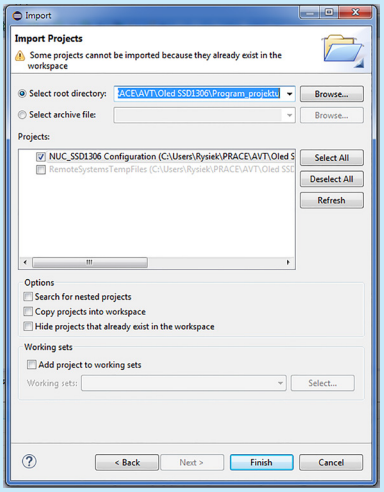
W efekcie tak zmodyfikowanego szkieletu oprogramowania, po naciśnięciu niebieskiego przycisku B1 na płytce NUCLEO, zapali się dioda LD2.

Krok trzeci: dołączenie wyświetlacza do płytki startowej

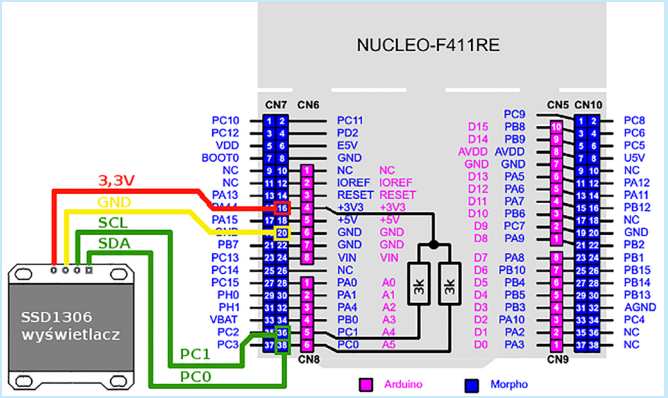
Zamontowany na płytce Nucleo-F411RE kontroler ma sprzętowy interfejs I²C. Jak się domyślamy, skorzystanie z tej opcji z jednej strony ułatwia napisanie oprogramowania, ale oznacza też konieczność dołączenia wyświetlacza do określonych wyprowadzeń mikrokontrolera. Dzięki użyciu programowej biblioteki do obsługi transmisji I²C, możliwe jest podłączenie wyświetlacza do dowolnych portów kontrolera. Na **rysunku 4** pokazano wszystkie konieczne do wykonania połączenia pomiędzy wyświetlaczem a płytką Nucleo. Wyświetlacz dołączono do styków płytki nazwanych „Morpho” za pomocą wtyków goldpin wsuniętych do odpowiednich gniazd „Arduino”. Dodano też niezbędne dla poprawnej pracy interfejsu I²C zewnętrzne oporniki podciągające o rezystancji 3 kΩ. Dla ułatwienia, w **tabeli 1** wymieniono niezbędne połączenia.

Krok czwarty: dołączanie do projektu zewnętrznych bibliotek

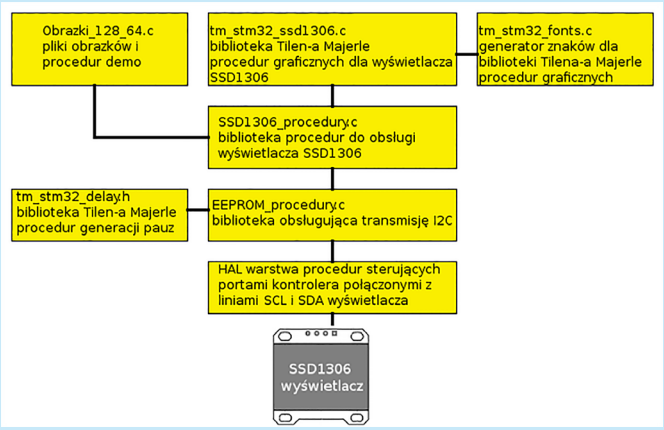
Dla wyświetlacza z kontrolerem SSD1306 potrzebne są procedury inicjujące jego pracę obraz przesyłające do niego dane. Nie ma potrzeby pisania ich od podstaw, ponieważ w części lub całości są one dostępne w postaci bibliotek. Można je znaleźć udostępnione w Internecie na zasadach licencji otwartej. Dobrym przykładem jest strona *http://goo.gl/JBS3uH* autorstwa Tilen-a Majerle. Zamieszczono na niej ciekawe i wartościowe biblioteki procedur oraz tutoriale (przewodniki) poświęcone programowaniu



Rysunek 3. Ustawienie pozostałych opcji projektu



Rysunek 4. Połączenia pomiędzy wyświetlaczem a płytką



Rysunek 5. Schemat blokowy pokazujący zależności pomiędzy bibliotekami

mikrokontrolerów STM32F4. Biblioteki można wykorzystać bezpośrednio lub po zmianach dopasować do potrzeb swoich projektów. Tilen Majerle opracował bibliotekę także dla wyświetlacza SSD1306. Pewne jej fragmenty zostaną wykorzystane w opisywanym projekcie.

Środowisko *AC6 System Workbench* ma mechanizmy dołączania nowych plików bibliotek do już utworzonego projektu. Zademonstruję to na przykładzie biblioteki Tilen-a Majerle zawierającej procedury odmierzające opóźnienia. Pozwala ona na tworzenie milisekundowych i mikrosekundowych opóźnień w pracy programu bez wykorzystywania modułów czasowo-licznikowych.

Posługując adresem *http://goo.gl/BwuMd5* należy pobrać spakowane pliki wszystkich bibliotek przeznaczonych do użycia ze sterownikami HAL: *TM STM32 Libraries STM32 libraries based on STM32Fxxx HAL drivers*. Po ściągnięciu, plik należy rozpakować w wydzielonym katalogu. Wśród wielu innych będą tam także znajdować się pliki o nazwach *tm_stm32_delay.c* i *tm_stm32_delay.h*. Są to właśnie biblioteki procedur opóźnień. Następnie należy otworzyć środowisko *AC6* z wygenerowanym wcześniej przez *STM32CubeMX* szkieletem projektu (przykładowym NUC_SSD1306). Na pulpicie z lewej strony znajduje się zakładka *Project Explorer* z nazwą otwartego projektu. Należy lewym przyciskiem myszki kliknąć na nazwę otwierając w ten sposób drzewo katalogów projektu. Nowe pliki najsensowniej dodawać do podkatalogu o nazwie *Application*. Klikamy prawym przyciskiem myszki na *Application* → *New* → *Folder* i w polu *Folder Name* wpisujemy nazwę folderu dodawanego dla plików biblioteki pauzy np. *Procedury_Pauzy*. Następnie, klikając prawym przyciskiem myszki na nazwę utworzonego folderu, wybieramy *Import* → *File System* → *Next*. Wskazujemy katalog, w którym znajdują się rozpakowane pliki bibliotek Tilen-a Majerle. Zaznaczamy pozycje *tm_stm32_delay.c* i *tm_stm32_delay.h* oraz naciskamy *Finish*. Pliki zostaną przekopiowane do utworzonego w projekcie podkatalogu *Procedury_Pauzy*.

Aby procedura mogła być „widziana" przez inne pliki programu, należy w projekcie zadeklarować ścieżkę dostępu do pliku nagłówkowego *tm_stm32_delay.h*. Należy prawym przyciskiem kliknąć na nazwę projektu i dalej: *Properities* → *C/C++ Build* → *Tool Settings* → *Includes*. Po naciśnięciu *Add* → *Workspace* należy wskazać nazwę folderu gdzie znajduje się plik nagłówkowy, czyli *Procedury_Pauzy*. Po zatwierdzeniu wyboru nowa ścieżka dostępu zostanie dodana do listy. Można się o tym przekonać rozwijając w drzewie katalogu projektów pozycję *Includes*.

Ostatnia poprawka dotyczy zawartości pliku tm_stm32_delay.h. Należy dokonać następujących zmian:

- jest:

```
#include „stm32fxxx_hal.h"
#include „defines.h"
#include „stdlib.h"
```

- powinno być:

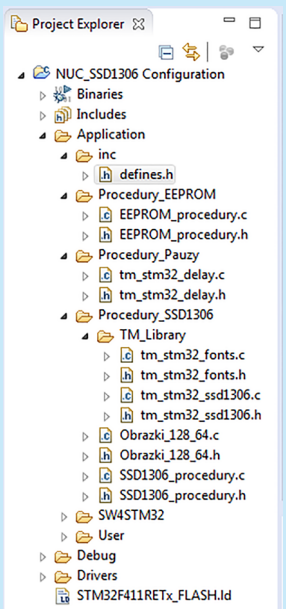
```
#include „stm32f4xx_hal.h"
// #include „defines.h"
#include „stdlib.h"
```

Po tych zmianach projekt powinien się skompilować bez komunikatów o błędach.

Biblioteki sterujące wyświetlaczem

Aby pokazać obraz na wyświetlaczu, trzeba zaprząc do pracy kilka bibliotek. Schemat blokowy pokazujący zależności pomiędzy nimi pokazano na **rysunku 5**. Omówmy je pokrótce:

- **Interfejs HAL.** Za sterowanie wyprowadzeniami interfejsu I²C odpowiada interfejs HAL. Pliki interfejsu zostały automatycznie dodane do projektu na etapie generowania szkieletu oprogramowania przez *STM32CubeMX*.
- **Obsługa sprzętowego interfejsu I²C.** *EEPROM_procedury.c*, *EEPROM_procedury.h* to pliki biblioteki do obsługi sprzętowego interfejsu I²C. Wykonano ją na bazie biblioteki służącej do komunikacji z pamięciami EEPROM wyposażonymi w interfejs I²C.
- **Bibliotek opóźnień.** Pliki *tm_stm32_delay.c*, *tm_stm32_delay.h*, to biblioteka opóźnień Tilen-a Majerle wykorzystywana przez procedury transmisji I²C.
- **Bibliotek procedur inicjujących wyświetlacz.** Pliki *SSD1306_procedury.c*, *SSD1306_procedury.h* to biblioteka procedur inicjujących wyświetlacz oraz procedur przesyłania danych i rozkazów do wyświetlacza. Umieszczono tu także procedury do wyświetlania obrazów zapisanych w plikach o formacie XBM.
- **Bibliotek do wyświetlania figur geometrycznych.** Pliki *tm_stm32_ssd1306.c*, *tm_stm32_ssd1306.h* to zmodyfikowana biblioteka Tilen-a Majerle do wyświetlania figur geometrycznych oraz napisów.
- **Wzorce znaków.** Pliki *tm_stm32_fonts.c*, *tm_stm32_fonts.h* zawierają biblioteki znaków (fontów) wykonanych przez Tilen-a Majerle. Są one niezbędne do wyświetlania napisów.



Rysunek 6. Drzewo projektu

Listing 2. Inicjowanie wyświetlacza z kontrolerem SSD1306

```
//procedura inicjacji SSD1306
//wy: status TRUE -sukces, FALSE -błąd
char SSD1306_Inicjacja(void)
{
char status=TRUE;
    Inicjacja_EEPROM();  //inicjowanie procedur I2C
    Delayms(100);
    /* Inicjowanie LCD */
    status =status & Write_Command(0xAE); //display off
    status =status & Write_Command(0x20); //Set Memory Addressing Mode
                                          //Mode;10,Page Addressing Mode (RESET);11,Invalid
    status =status & Write_Command(0x10); //00,Horizontal Addressing Mode;01,Vertical Addressing
    status =status & Write_Command(0xB0); //Set Page Start Address for Page Addressing Mode,0-7
    status =status & Write_Command(0xC8); //Set COM Output Scan Direction
    status =status & Write_Command(0x00); //---set low column address
    status =status & Write_Command(0x10); //---set high column address
    status =status & Write_Command(0x40); //--set start line address
    status =status & Write_Command(0x81); //--set contrast control register
    status =status & Write_Command(0xFF);
    status =status & Write_Command(0xA1); //--set segment re-map 0 to 127
    status =status & Write_Command(0xA6); //--set normal display
    status =status & Write_Command(0xA8); //--set multiplex ratio(1 to 64)
    status =status & Write_Command(0x3F); //
                                          //RAM content
    status =status & Write_Command(0xA4); //0xa4,Output follows RAM content;0xa5,Output ignores
    status =status & Write_Command(0xD3); //-set display offset
    status =status & Write_Command(0x00); //-not offset
                                          //frequency
    status =status & Write_Command(0xD5); //--set display clock divide ratio/oscillator
    status =status & Write_Command(0xF0); //--set divide ratio
    status =status & Write_Command(0xD9); //--set pre-charge period
    status =status & Write_Command(0x22); //
    status =status & Write_Command(0xDA); //--set com pins hardware configuration
    status =status & Write_Command(0x12);
    status =status & Write_Command(0xDB); //--set vcomh
    status =status & Write_Command(0x20); //0x20,0.77xVcc
    status =status & Write_Command(0x8D); //--set DC-DC enable
    status =status & Write_Command(0x14); //
    status =status & Write_Command(0xAF); //--turn on SSD1306 panel
    return status;
}
```

Listing 3. Zapisanie komendy lub danej do wyświetlacza

```
 //zapisanie komendy do SSD1306
//we: Command -kod komendy
//wy: status TRUE -sukces, FALSE -błąd
char Write_Command(unsigned char Command)
{
char bufor_komendy[2], status;
    bufor_komendy[0] =0x00;        //write command
    bufor_komendy[1] =(char) Command;
    status =EepZapis(SSD1306_SLAVE_ADR, ADRES_I2C_0BAJT, 0, 2, &bufor_komendy[0]);
    return status;
}

// zapis danej do SSD1306
//we: Data -dana
//wy: status TRUE -sukces, FALSE -błąd
char Write_Data(unsigned char Data)
{
char bufor_danych[2], status;
    bufor_danych[0] =0x40;         //write data
    bufor_danych[1] =(char) Data;
    status =EepZapis(SSD1306_SLAVE_ADR, ADRES_I2C_0BAJT, 0, 2, &bufor_danych[0]);
    return status;
}
```

- **Pliki obrazów.** *Obrazki_128_64.c*, *Obrazki_128_64.h*, to pliki przykładowych obrazków w formacie XBM i procedur demonstracyjnych.
- **defines.h** – plik nagłówkowy definicji stałych używanych, przez procedury programu.

Wszystkie wymienione pliki należy dołączyć do projektu w sposób opisany w czwartym kroku. Po dołączeniu plików drzewo projektu powinno wyglądać podobnie jak na **rysunku 6**.

Biblioteka sprzętowej obsługi transmisji I²C

Przed pierwszym użyciem biblioteka powinna być zainicjowana wywołaniem procedury *Inicjacja_EEPROM()*.

Do wysłania danych do wyświetlacza magistralą I²C służy funkcja *char EepZapis(unsigned char slave_adr, unsigned char ile_bajtow_adresu, unsigned int adres_int, unsigned int ilosc_bajtow, char *p_bufor)*. Należy ją wywołać z następującymi argumentami:

- **slave_adr** to adres urządzenia (dla wyświetlacza 0x78).
- **ile_bajtow_adresu** – tu zawsze 0.
- **adres_int** – nieużywany, można ustawić 0.
- **ilosc_bajtow** – liczba bajtów do przesłania za pomocą I²C.
- ***p_bufor** – wskaźnik do początku bufora zawierającego bajty do przesłania za pomocą I²C.

Funkcja zwraca wartość *True* w przypadku sukcesu i *False* w przypadku wystąpienia błędu.

Biblioteka obsługi wyświetlacza SSD1306

Przed pierwszym użyciem wyświetlacz powinien być zainicjowany. W tym celu należy wywołać procedurę *SSD1306_Inicjacja()*. Funkcja zapisuje do rejestrów wyświetlacza wartości początkowe – przedstawiono ją na **listingu 2**. Biblioteka używa osobnych procedur do wysyłania do wyświetlacza rozkazów i danych do wyświetlenia – pokazano je na **listingu 3**.

Listing 4. Przykład wyświetlenia tekstu 3 różnymi wielkościami czcionki

```
TM_SSD1306_GotoXY(0, 0);
TM_SSD1306_Puts("Test", &TM_Font_16x26, SSD1306_COLOR_WHITE);
TM_SSD1306_GotoXY(0, 28);
TM_SSD1306_Puts("biblioteki", &TM_Font_11x18, SSD1306_COLOR_WHITE);
TM_SSD1306_GotoXY(0, 48);
TM_SSD1306_Puts("Tilen-a Majerle", &TM_Font_7x10, SSD1306_COLOR_WHITE);
TM_SSD1306_UpdateScreen();
```

Listing 5. Rysowanie figur geometrycznych: prostokąta, okręgu, trójkąta

```
TM_SSD1306_Fill(SSD1306_COLOR_BLACK);
TM_SSD1306_DrawRectangle(32, 0, 62, 62, SSD1306_COLOR_WHITE);
TM_SSD1306_DrawCircle(63, 31, 30, SSD1306_COLOR_WHITE);
TM_SSD1306_DrawTriangle(63, 10, 43, 43, 83, 43, SSD1306_COLOR_WHITE);
TM_SSD1306_UpdateScreen();
```

Wyświetlanie obrazków o rozdzielczości 128×64 pikseli w formacie XBM obsługuje procedura *void picture_XBM(const unsigned char *p_obrazek)*. Jako parametr procedury należy podać wskaźnik do początku bufora z danymi obrazka.

Biblioteka procedur graficznych Tilen-a Majerle

Za pomocą tej procedury można na ekranie wyświetlacza zapalać pojedyncze piksele, kreślić linie, figury geometryczne takie jak okręgi i wielokąty a także wypisywać teksty o kilku rozmiarach czcionek. Przed pierwszym użyciem należy bibliotekę zainicjować wywołując procedurę *TM_SSD1306_Init()*. Przykład wyświetlenia tekstu 3 różnymi wielkościami czcionki pokazano na **listingu 4**. Na koniec wywołuje się procedurę wysłania utworzonych w buforze obrazów do wyświetlacza *TM_SSD1306_UpdateScreen()*. Kolejny przykład – pokazany na **listingu 5** – służy do zademonstrowania sposobu rysowania na ekranie 3 różnych figur geometrycznych.

Przygotowanie obrazków w formacie XBM

Do wykonania obrazków potrzebny będzie dowolny program pozwalający na konwersję do formatu graficznego XBM. Może to być choćby darmowy GIMP. Obrazek można wykonać samodzielnie albo użyć gotowego rysunku. Obraz należy przekształcić do postaci czarno-białej bez tonów pośrednich. Obrazek należy przeskalować do wymiarów: 128 piksele szerokości i 64 linie wysokości. Po zapisie w formacie XBM otrzymuje się plik tekstowy w formie tablicy, którą trzeba przekopiować do swojego programu. Wyświetlenie obrazka nastąpi po wywołaniu procedury:

```
picture_XBM(&tablica_z_danymi_obrazka[0]).
```

Ryszard Szymaniak, EP