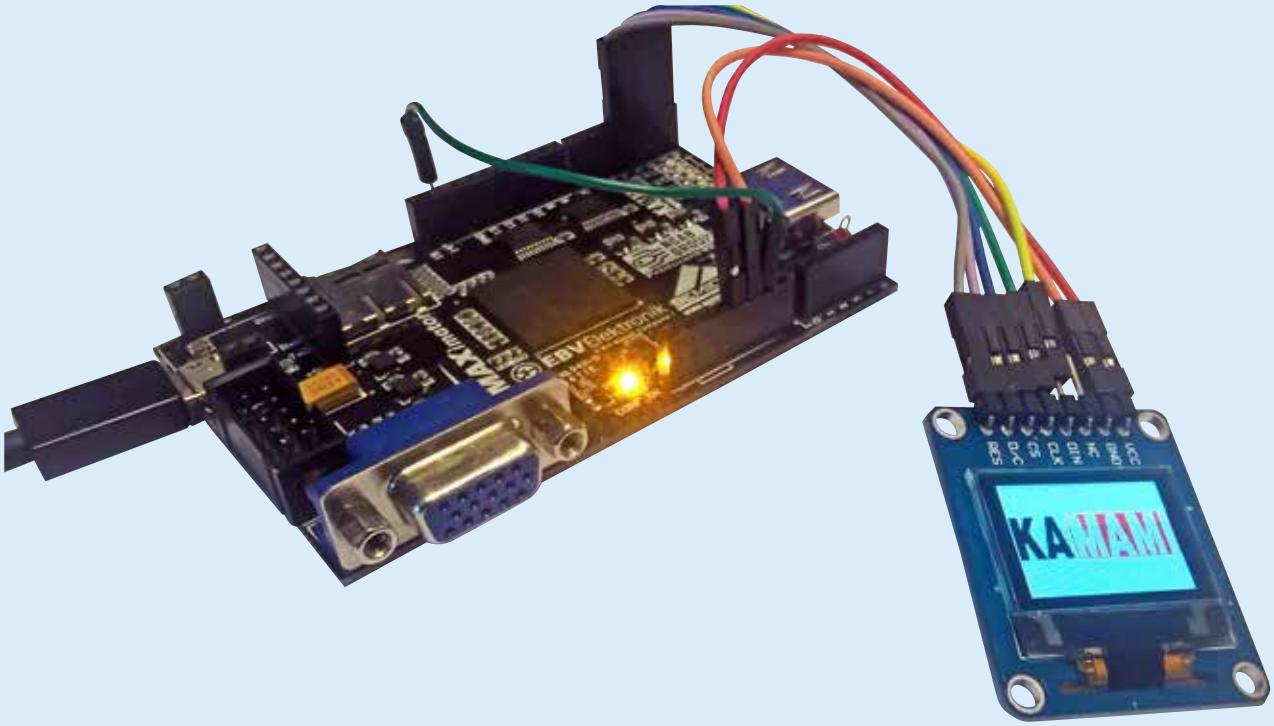
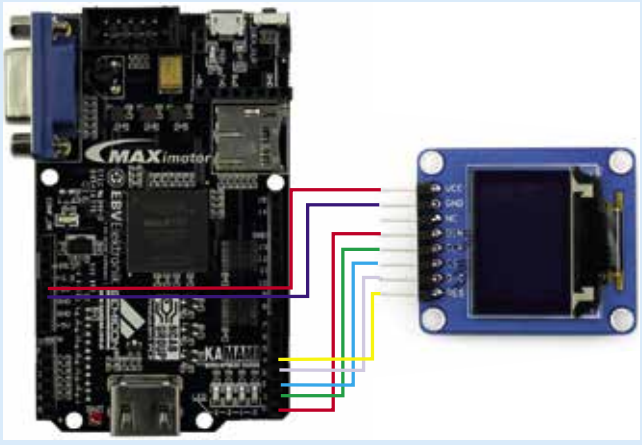


Pierwsze kroki z FPGA (9)

Obsługa wyświetlacza OLED z kontrolerem SSD1331

Celem projektu było obsłużenie sprzętowego kontrolera kolorowego wyświetlacza OLED za pomocą układu FPGA zestawu maXimator. Zastosowany w przykładzie wyświetlacz wyposażono w kontroler SSD1331, z którym aplikacja użytkownika komunikuje się poprzez interfejs SPI.

Więcej informacji:
Projekt powstał w Katedrze Elektroniki Wydziału Informatyki, Elektroniki i Telekomunikacji AGH, pod kierunkiem dr inż. Pawła Rajdy i dr inż. Jerzego Kasperka.



Rysunek 1. Schemat połączeń płytki maXimatora z wyświetlaczem

Prezentowany w artykule projekt wykonano z użyciem bezpłatnego oprogramowania narzędziowego Intel Quartus Prime, które można pobrać ze strony internetowej ***https://goo.gl/kuHmWj***. Moduł z kolorowym wyświetlaczem OLED-RGB dołączono do maXimatora zgodnie ze schematem pokazanym na **rysunku 1** i opisem umieszczonym w **tabeli 1**. Dodatkowo, wyprowadzenie D14 maXimatora jest używany jako wejście zerujące o aktywnym poziomie wysokim – podczas normalnej pracy ta linia musi być dołączona do masy zasilania.

Jednostka TOP LEVEL

Pokazana na rysunku 2 jednostka *TOP LEVEL* składa się z następujących elementów:

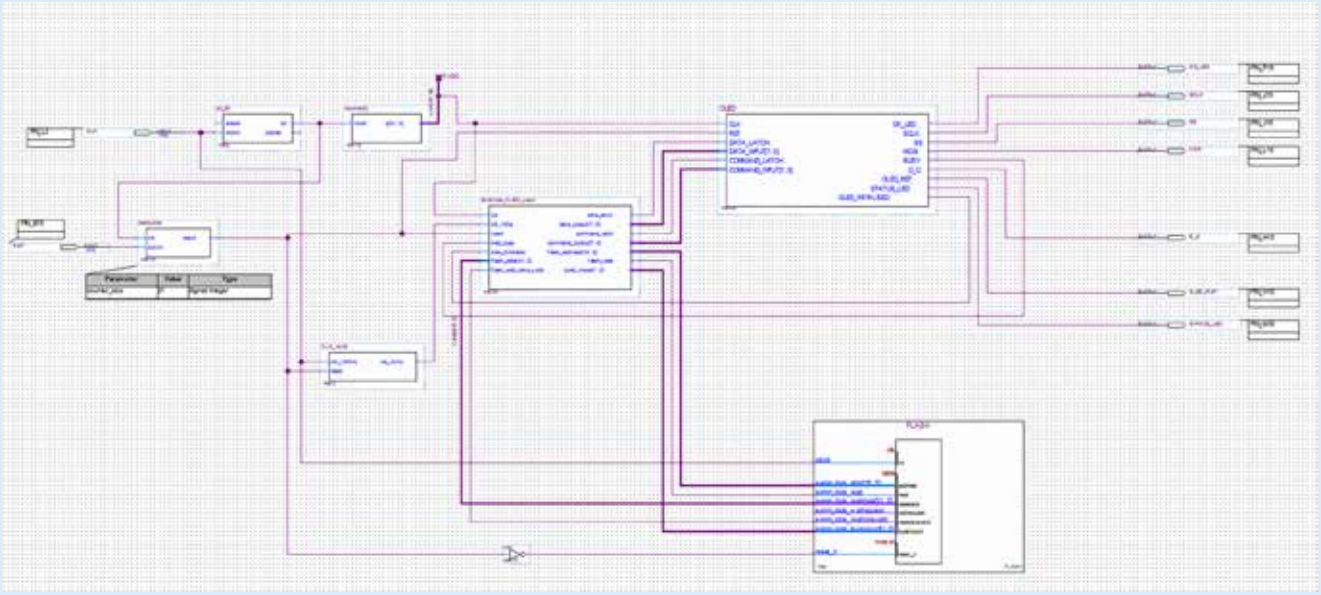
Bloku **OLED**, który jest odpowiedzialny za inicjalizację oraz przesyłanie komend i danych do kontrolera wyświetlacza.

Bloku **EXAMPLE_OLED_Logic**, w którym pokazano użycie przykładowych komponentów do sterowania wyświetlaczem:

Tabela. 1. Sposób dołączenia modułu OLED-RGB

Pin OLED	maXimator
VCC	+5V
GND	GND
DIN	D0
CLK	D1
CS	D2
D/C	D3
RES	D4

Poprzednie części kursu i dodatkowe materiały dostępne są na FTP: **ftp://ep.com.pl**, user: **77322**, pass: **8qxonzsb**



Rysunek 2. Wygląd projektu

CLS_Command – czyści wyświetlacz,

SetPixel_Commmand – ustawia wybrany piksel wyświetlacza na wybrany kolor,

DrawImageFull_Commmand – rysuje cały obrazek uprzednio pobierając go z pamięci.

Bloku **Flash**, który implementuje blok pamięci typu Flash zainicjalizowanej plikiem *onchip_flash.dat*. Przechowane są w niej mapy bitowe dwóch obrazów, które będą przesłane do wyświetlacza.

Oprócz wyżej wymienionych, na schemacie blokowym możemy znaleźć jeszcze blok odpowiedzialny za *debouncing* przycisku zerowania oraz blok generatora pętli PLL, na którego na wyjściu występuje przebieg o częstotliwości 50 MHz.

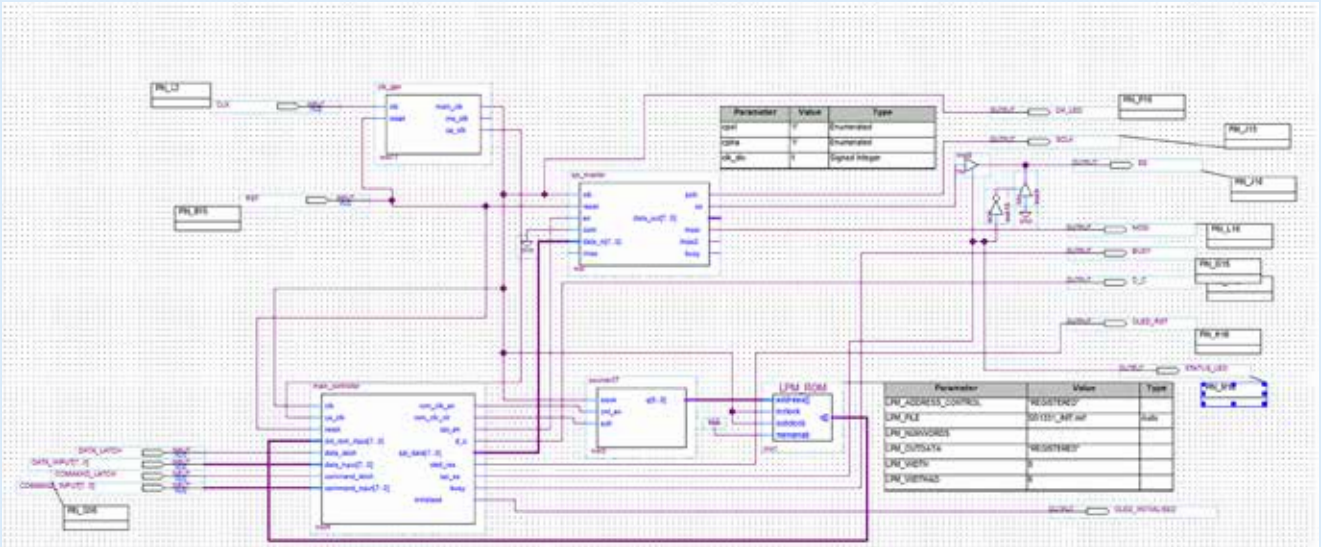
Moduł OLED

Moduł OLED (**rysunek 2**) jest głównym elementem projektu. Odpowiada on za komunikację za pomocą interfejsu SPI, inicjalizację wyświetlacza oraz późniejsze przekazywanie komend i danych. Sygnały modułu OLED opisano w **tabeli 2**.

Moduł OLED zawiera w sobie następujące bloki (**rysunek 3**):

Tabela 2. Wyprowadzenie modułu OLED

Port	Funkcja
CLK	Sygnał zegarowy
RST	Sygnał reset
DATA_INPUT[7..0]	Wejście 8-bitowe dla danych
DATA_LATCH	Wejście zatrzaskujące wejście DATA_INPUT[7..0]
COMMAND_INPUT[7..0]	Wejście 8-bitowe dla komend
COMMAND_LATCH	Wejście zatrzaskujące wejście COMMAND_INPUT[7..0]
SCLK	Sygnał zegarowy SPI
SS	Slave Select wybór urządzenia slave SPI
MOSI	Master Output Slave Input interfejsu SPI
D_C	Linia informująca kontroler o tym czy przesyłamy dane czy komendy
OLED_RST	Sygnał służący resetowaniu wyświetlacza OLED
OLED_INITIALISED	Sygnał informujący o tym, że wyświetlacz został zainicjalizowany
BUSY	Sygnał informujący o tym, że moduł OLED jest w trakcie przetwarzania komendy, danych. Nowe komendy/dane należy przesyłać wtedy, gdy stan tego pinu jest niski.
STATUS_LED	Sygnał sterujący LED



Rysunek 3. Bloki zawarte w module OLED

spi_master będący sprzętową implementacją sterownika SPI, komunikującego się bezpośrednio z kontrolerem wyświetlacza OLED. Funkcje jego wyprowadzeń zebrano w **tabeli 3**.

LPM_ROM to gotowy IP będący pamięcią ROM. Zapisany w nim plik *.mif* (*SD1331_INIT.mif*) zawiera sekwencję komend służących do inicjalizacji wyświetlacza. Dokładne znaczenie poszczególnych bajtów znajduje się w dokumentacji kontrolera SSD1331. Zawartość pamięci bloku LMP_ROM opisano w **tabeli 4**.

main_controller to blok ten jest odpowiedzialny za odczyt sekwencji inicjalizującej z bloku LPM_ROM, a także sterowaniem blokiem *spi_master*. Funkcje jego wyprowadzeń opisano w **tabeli 5**.

clk_gen jest dzielnikiem sygnału taktującego, na wyjściu *main_clk* występuje wejściowy sygnał zegarowy o częstotliwości dzielonej przez 2, a na wyjściu *us_clk* – sygnał zegarowy o częstotliwości dzielonej przez 6.

Tabela 3. Wyprowadzenie modułu spi_master

Port	Funkcja
clk	Sygnał zegarowy
reset	Sygnał reset
en	Sygnał inicjujący transakcję
cont	Sygnał określający, czy mamy do czynienia z transakcją ciągłą czy pojedynczą
data_in[7..0]	Bajt który zostanie przesłany poprzez SPI
miso	Wejście Master Input Slave Output (nie używane)
sclk	Sygnał zegarowy SPI
ss	Slave Select
data_out[7..0]	Bajt otrzymany od urządzenia slave
mosi	Wyjście Master Output Slave Input
busy	Sygnał informujący o zajętości modułu spi_master

Tabela 5. Funkcje wyprowadzeń bloku main_controller

Port	Funkcja
clk	Sygnał zegarowy
us_clk	Drugi sygnał zegarowy
reset	Sygnał reset
init_rom_input[7..0]	Tryb ciągłego przesyłania danych (nie używany)
data_latch	Zatrzask danych
data_input[7..0]	Dane wejściowe
command_latch	Zatrzask komendy
command_input[7..0]	Komenda
Rom_clk_en	Sygnał aktywujący licznik sterujący adresacją pamięci ROM
Rom_clk_clr	Sygnał służący wyzerowaniu ww. licznika
Spi_en	Sygnał aktywujący moduł SPI
D_c	Sygnał określający, czy przesyłamy dany czy komendę
Spi_data[7..0]	Dane przesyłane do modułu SPI
Oled_res	Sygnał zerujący sterownik wyświetlacza
Spi_ss	Sygnał Slave Select
busy	Sygnał informujący o zajętości modułu
initialized	Sygnał informujący o tym, że wyświetlacz został zainicjalizowany

Moduł Example_OLED_Logic

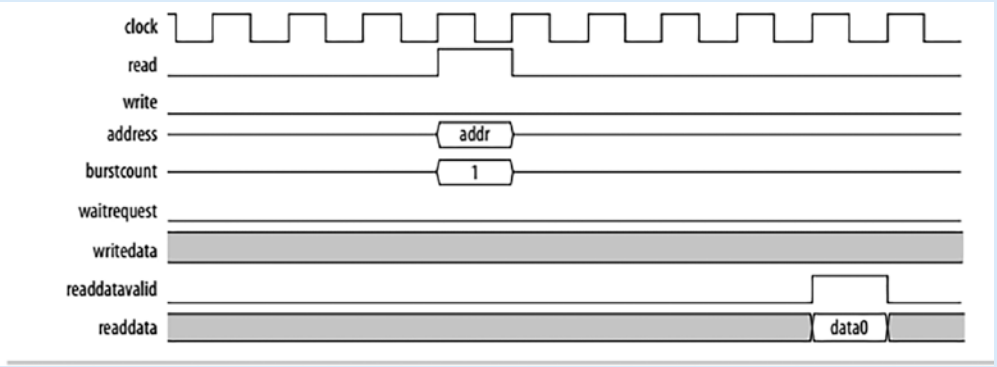
Moduł wykonano, aby pokazać, jak należy sterować modułem OLED poprzez odpowiednie komendy. W jego architekturze zawarto trzy komponenty:

CLS_Command, który ma za zadanie wyczyścić wyświetlacz, tzn. wypełnić go zadanym kolorem. Funkcje jego wyprowadzeń pokazano w **tabeli 6**.

Tabela 4. Zawartość pamięci LMP_ROM

Bajt	Znaczenie
0xAE	Wyłączenie wyświetlacza OLED
0x75	Ustawienie adresu wiersza
0x00	
0x3F	
0x15	Ustawienie adresu kolumny
0x00	
0x5F	
0xA0	Ustawienie formatu danych
0x72	RGB 565
0xA1	Ustawienie wiersza startowego RAM
0x00	
0xA2	Ustawienie offsetu wyświetlacza
0x00	
0xA4	Ustawienie trybu wyświetlacza
0xA8	Ustawienie multipleksera
0x3F	
0xAD	Ustawienia ogólnej konfiguracji
0x8F	
0xB0	Konfiguracja trybu oszczędzania energii
0x1A	
0xB1	Ustawienie okresu fazy 1 oraz 2
0x74	
0xB3	Ustawienie stosunku dzielnika zegara do częstotliwości oscylatora
0xD0	
0x8A	Ustawienie poziomu Second Pre-charge Speed dla koloru A
0x81	
0x8B	Ustawienie poziomu Second Pre-charge Speed dla koloru B
0x82	
0x8C	Ustawienie poziomu Second Pre-charge Speed dla koloru C
0x83	
0xBB	Ustawienie wartości Pre-charge
0x3E	
0xBE	Ustawienie poziomu VCOMH
0x3E	
0x87	Konfiguracja wartości natężenia prądu
0x0F	
0x81	Konfiguracja kontrastu A
0x80	
0x82	Konfiguracja kontrastu B
0x80	
0x83	Konfiguracja kontrastu C
0x80	
0xAF	Uruchomienie wyświetlacza

Aplikacje maXimatora
Pod adresem ***https://goo.gl/f7NsHq*** są dostępne przykładowe projekty przygotowane za pomocą bezpłatnego programu Intel Quartus Prime dla zestawu maXimator. Dostępne jest tam także kompletny zestaw plików źródłowych projektu prezentowanego w EP.



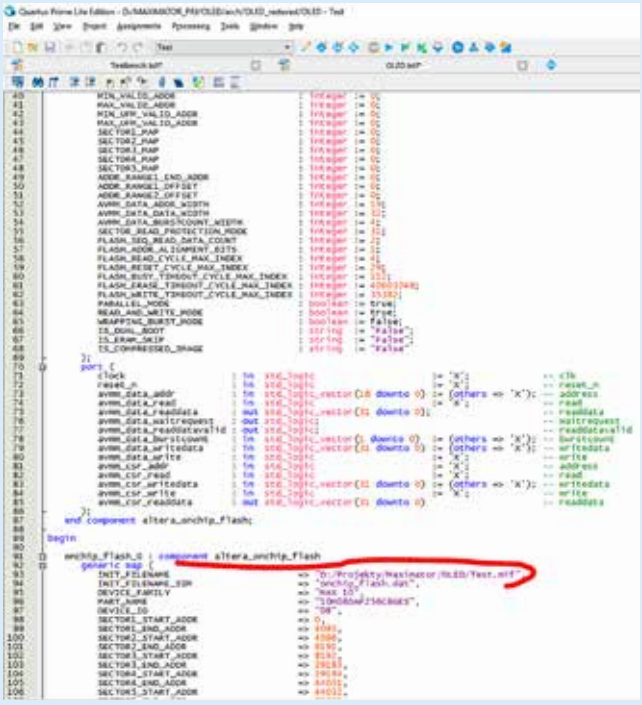
Rysunek 4. Przebiegi sygnałów w module Flash podczas odczytu

Tabela 6. Funkcje wyprowadzeń bloku CLS_Command

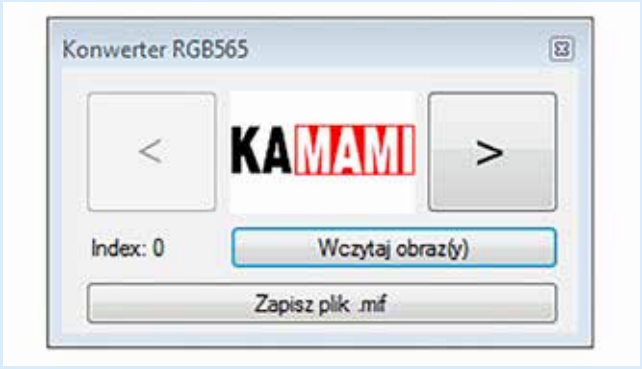
Port	Funkcja
Clk	Sygnał zegarowy
Reset	Sygnał reset
Busy	Sygnał zajętości modułu OLED
Start	Sygnał aktywujący komponent
oled_initialised	Sygnał informujący o zainicjalizowaniu wyświetlacza OLED
data_latch	Zatrzask magistrali danych
data_output[7..0]	Magistrala danych
command_latch	Zatrzask magistrali komend
command_output[7..0]	Magistrala komend
busy_out	Sygnał informujący o zajętości komponentu CLS_Command

Tabela 7. Funkcje wyprowadzeń bloku SetPixel_Command

Port	Funkcja
Clk	Sygnał zegarowy
reset	Sygnał reset
Busy	Sygnał zajętości modułu OLED
start	Sygnał aktywujący komponent
oled_initialised	Sygnał informujący o zainicjalizowaniu wyświetlacza OLED
pos_X	Numer wiersza
pos_y	Numer kolumny
color	Kolor piksela w formacie RGB565
data_latch	Zatrzask magistrali danych
data_output[7..0]	Magistrala danych
command_latch	Zatrzask magistrali komend
command_output[7..0]	Magistrala komend
busy_out	Sygnał informujący o zajętości komponentu DrawPixel_Command



Rysunek 5. Wprowadzenie ścieżki dostępu do plików obrazu



Rysunek 6. Ekran powitalny programu Konwerter_RGB565

SetPixel_Command, którego zadaniem jest zaświecenie w zadanym kolorze pojedynczego piksela o ustalonym adresie. Funkcje jego wyprowadzeń pokazano w **tabeli 7**.

DrawImageFull_Command, który pobiera zapisany w pamięci Flash obraz w celu wyświetlenia go na ekranie. Funkcje jego wyprowadzeń pokazano w **tabeli 8**.

Moduł Flash

Ten moduł służy do implementacji pamięci Flash (UFM), która może być zainicjalizowana odpowiednio przygotowanym plikiem *hex* lub *mif* (*memory initialization file*). Na **rysunku 4** pokazano przebiegi sygnałów podczas odczytu. Pojemność pamięci w układzie MAX10 znajdującym się w zestawie MAXimator pozwala na zapamiętanie dwóch obrazków o rozdzielczości 96×64 pikseli w formacie RGB565. Aby wybrać plik, który ma zostać wgrany należy w pliku *Flash.vhd* w linii 93 (**rysunek 5**) podać ścieżkę bezwzględną np: *INIT_FILENAME => „D:/Projekty/Maximator/OLED/Test.mif”*.

Dodatkowe oprogramowanie

W ramach projektu utworzono aplikację *Konwerter_RGB565*, której zadaniem jest wygenerowanie pliku *.mif* z wybranych przez

Tabela 8. Funkcje wyprowadzeń bloku DrawImage-Full_Command

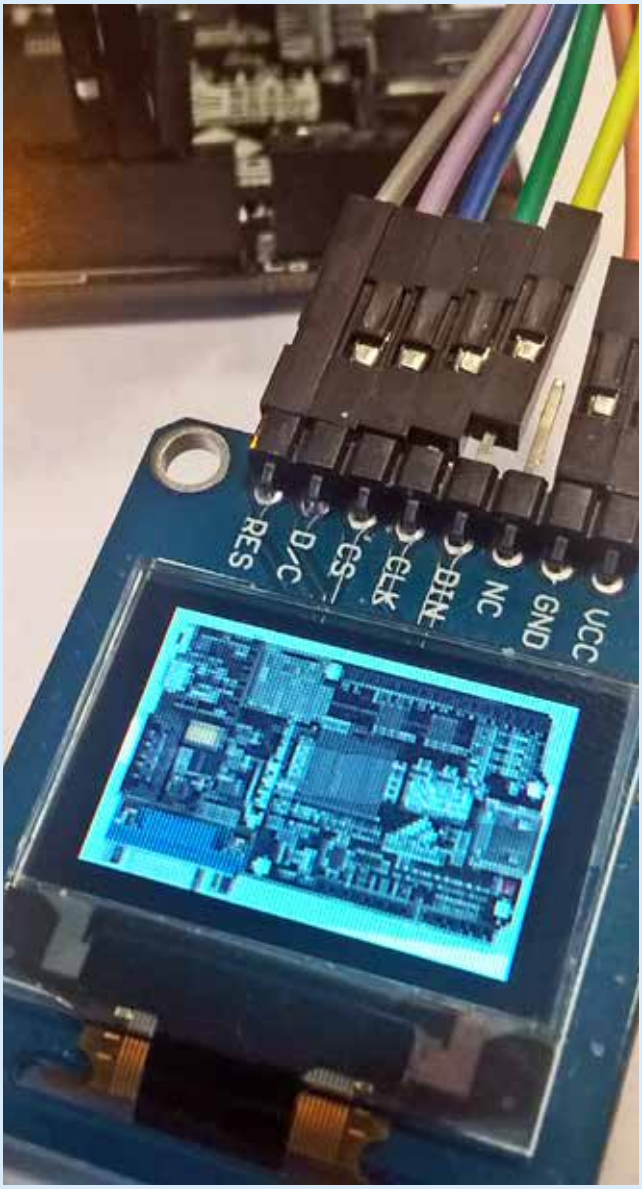
Port	Funkcja
clk	Sygnał zegarowy
reset	Sygnał reset
busy	Sygnał zajętości modułu OLED
start	Sygnał aktywujący komponent
oled_initialised	Sygnał informujący o zainicjalizowaniu wyświetlacza OLED
flash_data	Magistrala danych pamięci Flash
flash_read_data_valid	Sygnał informujący o gotowości danych do odczytu
flash_address_offset	Offset adresu pamięci Flash
flash_read	Sygnał inicjalizujący odczyt danych
flash_address	Adres z którego odczytujemy dane
data_latch	Zatrzask magistrali danych
data_output[7..0]	Magistrala danych
command_latch	Zatrzask magistrali komend
command_output[7..0]	Magistrala komend
busy_out	Sygnał informujący o zajętości komponentu DrawImageFull_Command

użytkownika obrazków w formacie *.bmp* i wymiarach 96×64 piksele (odpowiada matrycy OLED użytego wyświetlacza). Skompilowana aplikacja jest dostępna w portalu ***https://goo.gl/HxTqGr***.

Po uruchomieniu programu (**rysunek 6**) należy kliknąć przycisk *Wczytaj obraz(y)*, po czym zostanie wyświetlone okno wyboru plików do wczytania. Należy zaznaczyć jeden lub dwa obrazy (mając dwa obrazy trzeba je zaznaczyć jednocześnie, np. za pomocą klawisza CTRL). Obrazy powinny mieć format BMP, 24-bity, RGB, wymiary 94×64 piksele, 96 DPI).

Po wybraniu i otwarciu obrazków są wyświetlane ich miniaturki. Przycisk „*Zapisz plik .mif*" jest aktywny. Po jego kliknięciu program zapyta się, gdzie zapisać wygenerowany plik *.mif*. Na **fotografii 7** pokazano przykładowy efekt działania projektu: wyświetlenie obrazka, miniaturowej fotografii zestawu maXimator na wyświetlaczu OLED.

Mateusz Mamala, AGH



Fotografia 7. Wynik działania projektu z przykładu