INSTYTUT INFORMATYKI
Wydział IEiT AGH

Ćwiczenie laboratoryjne lub pokaz zdalny

Elementy graficznych interfejsów użytkownika w systemach mikroprocesorowych

Nazwa kodowa: **`lcd`**. Wersja **`20231123.0`**

Ada Brzoza
ada.brzoza@agh.edu.pl

1 Wstęp

1.1 Cel ćwiczenia

Celem ćwiczenia jest zapoznanie się możliwościami oferowanymi przez współcześnie stosowane elementy interfejsów użytkownika: wyświetlacze graficzne, sprzętowe moduły wspomagające wyświetlanie obrazu, panele dotykowe.

1.2 Wymagania wstępne

Wymagania wstępne do przeprowadzenia ćwiczenia są następujące:

- umiejętność posługiwania się językiem C,
- rozumienie pojęcia kompilacji skrośnej (*cross-compiling*) i umiejętność posługiwania się narzędziami do tego rodzaju kompilacji,
- umiejętność rozróżnienia typowych interfejsów zewnętrznych komputera PC,
- umiejętność rozróżniania rejestrów wewnętrznych mikroprocesora oraz rejestrów układów peryferyjnych mikrokontrolera lub systemu mikroprocesorowego,
- umiejętność posługiwania się narzędziami i elementami projektu poznanymi w czasie poprzednich ćwiczeń laboartorjnych.

1.3 Stanowisko testowe i plan pracy

Ćwiczenie przeprowadzone jest z wykorzystaniem nplatformy sprzętowej **STM32F746G-DISCO** z mikrokontrolerem z rdzeniem ARM Cortex-M7, graficznym wyświetlaczem LCD o rozdzielczości 480x272 pikseli i pojemnościowym panelem dotykowym. Od strony programowej użyjemy:

- środowiska **STM32CubeIDE w wersji 1.7.0**,
- programu terminalowego **PuTTY**.

1.4 Materiały pomocnicze

Do wykonania ćwiczenia niezbędne lub przydatne są następujące materiały:

- jeśli pracujemy w laboratorium: projekt startowy ***F7_LCD_HID***,
- nota katalogowa (*datasheet*) mikrokontrolera STM32F746,
- instrukcja obsługi (*reference manual*) mikrokontrolera STM32F746,
- instrukcja obsługi (user manual) zestawu STM32F746G-DISCO, dokument UM1907,
- zestaw przykładów dla STM32F746G-DISCO, np.:
  - **`~/STM32Cube/Repository/STM32Cube_FW_F7_V1.16.1/Projects/STM32746G-Discovery/Examples/BSP/Src`**
  - **`~/STM32Cube/Repository/STM32Cube_FW_F7_V1.16.1/Projects/STM32746G-`**

`Discovery/Applications/Display/LTDC_PicturesFromSDCard/`

Uwaga: jeśli na dysku komputerów w laboratorium nie ma zestawu sterowników/kodu w wersji *STM32Cube_FW_**F7_V1.16.1***, wystarczy odnaleźć wersję najbliższą dostępną lokalnie lub pobrać 1.16.1 na dysk ze strony producenta – teraz korzystamy bowiem jedynie z przykładów jako referencji.

2 Opis stanowiska

2.1 Zastosowany sprzęt

Ćwiczenie wykonywane jest na płytce testowej STM32F746G-DISCO. Została ona wyposażona w m.in. w kolorowy wyświetlacz graficzny o rozdzielczości 480x272 pikseli z panelem dotykowym typu pojemnościowego. Wyświetlacz jest kontrolowany przez specjalizowany układ peryferyjny **LTDC** (*LCD-TFT Controller*) mikrokontrolera. Za wspomaganie (przyspieszanie) generowania obrazu do wyświetlenia odpowiada moduł **DMA2D**. Do obsługi większości zaawansowanych układów peryferyjnych na płytce testowej można użyć dostarczonych przez producenta bibliotek wyższego poziomu BSP (*Board Support Package*). Kod BSP został napisany w taki sposób, aby dla uzyskania podstawowej funkcjonalności można było go relatywnie łatwo dołączyć do projektu nie wnikając w szczegóły działania poszczególnych układów peryferyjnych. Mimo wszystko kod sterowników z BSP jest otwarty i można przeanalizować, w jaki sposób korzysta z niższych warstw oprogramowania oraz ew. rozszerzyć jego funkcjonalność wg indywidualnego zapotrzebowania.

2.2 Projekt bazowy

Projekt bazowy ***F7_LCD_HID*** zapewnia już po rozpakowaniu właściwie działający kod, jednak bez kilku elementów, które będzie należało zmodyfikować lub dopisać samodzielnie realizując ćwiczenie. Kompilacja projektu, programowanie pamięci Flash mikrokontrolera oraz usuwanie plików wynikowych mogą być przeprowadzone przy pomocy poleceń wydawanych z poziomu środowiska STM32CubeIDE. Projekt był tworzony i sprawdzany w środowisku **STM32CubeIDE w wersji 1.7.0**. Taka wersja jest także zalecana do pracy nad ćwiczeniem w laboratorium.

Funkcja **`main`** znajduje się w pliku main.c w podkatalogu *./Src* projektu. Pliki nagłówkowe części konfigurowalnej projektu znajdziemy w *./Inc*.

W nowym projekcie możemy wysyłać przez łącze szeregowe komunikaty do wyświetlenia programie terminalowym. Komunikaty te możemy wysyłać zarówno przy pomocy funkcji **`printf`** jak i **`xprintf`**. Kod funkcji **`xprintf`** znajduje się w module **`Src/term_io.c`**. Może ona być przydatna do wyświetlania komunikatów bez znaku końca linii.

Uwaga: W wersji mikrokontrolerów STM32F746 zastosowanych do budowy płytki STM32F746G-DISCO (konkretnie w używanej przez nas ich implementacji rdzenia Cortex-M7) wystąpił błąd sprzętowy, który utrudnia debugowanie: nie jest możliwe krokowe wykonywanie kodu, tj. po wydaniu poleceń **`next`** lub **`step`** w GDB, program zaczyna zachowywać się w sposób niezgodny z oczekiwaniami. Zatrzymywanie działania programu, a także analiza stosu i rejestrów powinna działać poprawnie. W najnowszych implementacjach rdzenia Cortex-M7 błąd ten został poprawiony.

W projekcie bazowym zostały przygotowane bufory do przechowywania danych graficznych do wyświetlenia na LCD. Są to dwie tablice: **`lcd_image_fg`**, **`lcd_image_bg`** zdefiniowane w module **`main.c`**.

```
static volatile uint32_t lcd_image_fg[LCD_Y_SIZE][LCD_X_SIZE]
__attribute__((section(".sdram")));

static volatile uint32_t lcd_image_bg[LCD_Y_SIZE][LCD_X_SIZE]
__attribute__((section(".sdram")));
```

Definicje tych tablic mają dodatkowe atrybuty informujące narzędzia kompilacji, w którym obszarze przestrzeni adresowej (sekcji) mają się znaleźć oraz atrybut mówiący o tym, że zmienne te mogą być nieużywane i nie wygeneruje to ostrzeżenia kompilatora. Sekcja *.sdram* została zdefiniowana w skrypcie linkera *STM32F746NGHX_FLASH.ld* i odpowiada obszarowi **zewnętrznej pamięci SDRAM dołączonej do mikrokontrolera**. Dzięki wbudowanemu kontrolerowi pamięci zewnętrznej, SDRAM jest widziana w przestrzeni adresowej tak samo, jak pozostałe, wewnętrzne pamięci mikrokontrolera.

2.2.1 System operacyjny

Projekt startowy kompilowany jest razem z systemem operacyjnym FreeRTOS. W niniejszym ćwiczeniu korzystamy jedynie z podstawowej funkcjonalności oferowanej przez FreeRTOS: włączamy tylko jedno zadanie użytkownika o nazwie **`mainTask`** w module **`main.c`**. W tym zadaniu będziemy implementować większość realizowanej funkcjonalności. Z innymi podstawowymi możliwościami systemu operacyjnego FreeRTOS zapoznamy się realizując kolejne ćwiczenie.

2.2.2 Biblioteki sterowników niskopoziomowych

Przydatne w niniejszym ćwiczeniu sterowniki dla układów peryferyjnych mikrokontrolera znajdują się w podkatalogu **`Drivers`** projektu startowego. W kolejnych podkatalogach mamy dostępne następujące komponenty:

- **`STM32F7xx_HAL_Driver/`** – podkatalog ze sterownikami zapewniającymi warstwę abstrakcji (HAL) dla układów peryferyjnych mikrokontrolera,
- **`CMSIS/`** – elementy ujednoliconego frameworku programowego dla rdzeni mikroprocesorów Cortex-M (*Cortex Microcontroller Software Interface Standard*),
- **`Drivers/BSP/STM32746G-Discovery/`** – podkatalog zawierający sterowniki specyficzne dla płytki testowej (*Board Support Package*); najciekawszymi z naszego punktu widzenia plikami z tego podkatalogu mogą być:
  - ***stm32746g_discovery_lcd.c* / *.h***,
  - dodatkowo również ***stm32746g_discovery.c* / *.h***.

2.2.3 Watchdog

Watchdog to układ czasowo-licznikowy (timer), którego używa się w systemach mikroprocesorowych m.in. do uzyskania zabezpieczenia przed sytuacją, w której urządzenie stanie się nieresponsywne, tj. „zawiesi się” (właściwie znajdzie się w stanie, z którego nie będzie można łatwo kontynuować biegu programu). Przykładem takiej sytuacji jest wystąpienie wyjątku, który nie

jest w pełni obsługiwany inaczej niż przez wejście do nieskończonej pętli. Bez układu watchdog, takie „zawieszenie się” wymagałoby zewnętrznej ingerencji użytkownika, aby przywrócić działanie tego urządzenia, np. przez wciśnięcie przycisku zerującego (reset).

Typowe działanie układu watchdog polega na tym, że gdy „odliczy” on zadany interwał czasu, to automatycznie wykona zerowanie systemu mikrprocesorowego. W przypadku stanowiska laboratoryjnego do niniejszego ćwiczenia jako watchdoga używamy układu peryferyjnego IWDG. Watchdog jest włączany przy inicjalizacji, w funkcji ***MX_IWDG_Init***. Po inicjalizacji musi on być odpowiednio często „odświeżany” przez wywołanie funkcji ***HAL_IWDG_Refresh*** – zostało to wstępnie zaimplementowane w pętli głównej zadania ***StartDefaultTask***. Jeśli zaniechamy odświeżania watchdoga przez wywołanie *HAL_IWDG_Refresh*, to mikrokontroler samoczynnie wyzeruje się. Sterowniki IWDG wraz z dokumentacją w komentarzach znajdziemy projekcie w plikach ***stm32f7xx_hal_iwdg.c / .h*** w podkatalogach ***Drivers/ STM32F7xx_HAL_Driver/Src*** i ***Inc*** .

2.2.4 Obsługa karty pamięci SD

Jeśli w gnieździe *CN3 µSD* płytki testowej umieścimy poprawnie sformatowaną kartę pamięci SD/SDHC, to będziemy mieć możliwość zapisu i odczytu plików. Karta powinna być sformatowana w systemie plików FAT. W funkcji **`serialTestComm`** w module main.c mamy dostępne przykładowe fragmenty kodu wywoływane przez wysyłanie z terminala puTTY znaków ASCII. Do obsługi testowego zapisu i odczytu plików możemy uruchomić przykładowe procedury:

- przykład zapisu do pliku sprawdzimy po wysłaniu **`W`** (*upper case*)
- przykład odczytu z pliku w formacie tekstowym lub wyglądającym jak „zrzut” pamięci w edytorze hex sprawdzimy wysyłając odpowiednio znak **`R`** lub **`H`** (*także upper case*).

2.2.5 Obsługa urządzeń USB klasy HID: myszy i klawiatury

W projekcie startowym mamy dostępny przykładowy kod umożliwiający przesuwanie kursora po ekranie wyświetlacza przy pomocy dołączonej do płytki testowej myszy albo klawiatury z interfejsem USB. Dane na temat wciśniętych klawiszy, przycisków lub ruchu myszy zostają wyświetlane na terminalu. Samodzielna analiza tych fragmentów kodu nie powinna sprawić wiele problemu. Dodatkowe urządzenie wejściowe może być przydatne przy realizacji „zadania kreatywnego” z dalszej części instrukcji.

2.2.6 Obsługa serwera WWW

Serwer WWW może działać podobnie jak w poprawnie wykonanym ćwiczeniu debug-devel. Do oprogramowania serwera WWW potrzebna jest inicjalizacja biblioteki LwIP. Aby uaktywnić kod biblioteki LwIP, należy „odblokować” jej inicjalizację zmieniając w pliku **`main.h`** makrodefinicję:

```
#define CFG_DISABLE_LWIP    1
```

na

```
#define CFG_DISABLE_LWIP    0
```

Inicjalizacja LwIP została „zablokowana” ponieważ domyślnie nie korzystamy z LwIP, a uruchamianie jej kodu znacznie wydłuża całkowity czas inicjalizacji firmware.

2.2.7 Edycja projektu bazowego

Projekt startowy do ćwiczenia został przygotowany przy pomocy **`STM32CubeMX w wersji 6.3.0`** oraz przy zastosowaniu kodu sterowników (*Firmware Package*) w wersji **`1.16.1`**. Jeśli zajdzie potrzeba edycji projektu, to zalecane jest zastosowanie podanych tutaj wersji narzędzi.

3 Zadania do wykonania

W ramach sprawozdania z realizacji ćwiczenia, należy zaprezentować osobie prowadzącej uzyskane efekty realizacji ćwiczeń, a na platformie UPEL proszę umieścić:

- kopię pliku *main.c* powstałą w trakcie realizacji ćwiczenia aż do etapu uzyskania kursora wg opisu (czyli do punktu 3.5 włącznie); **nazwę kopii pliku *main.c* proszę zmienić na *main_kursor.c***,
- Zdjęcie ekranu wyświetlającego kursor jak wyżej,
- kopię pliku main.c powstałą w wyniku realizacji kreatywnego zadania; **nazwę kopii *main.c* proszę zmienić na *main_kreatywne.c***,
- zdjęcie ekranu z widocznym efektem realizacji zadania kreatywnego.

Część zadań do wykonania w tym ćwiczeniu polega na świadomej analizie dostępnego kodu oraz na przeprowadzeniu własnych eksperymentów. Zamiast przepisywać kod z instrukcji, należy zapoznać się z wewnętrznymi mechanizmami działania:

- obsługi wyświetlacza:
  - funkcja **`lcd_start`** w *main.c*
  - sterowniki LCD zawarte w plikach **`Drivers/BSP/STM32746G-Discovery/stm32746g_discovery_lcd.c / .h`**.
- kodowania kolorów:
  - W jaki sposób kodowane są kolory w projekcie przy domyślnych ustawieniach i zestawie sterowników? Warto wiedzieć jakimi liczbami w systemie szesnastkowym zakodować m.in.:
    - podstawowe RGG: czerwony, zielony, niebieski,
    - CMY: cyjan, fuksja (magenta), żółty,
    - biały,
    - czarny,
    - odcienie kolorów o jasnościach: 12,5%, 25%, 50%, 75%, 87,5% (przyjmijmy, że 25% jest ciemniejszy niż 75%), np. szary o jasności 12,5% lub ciemnozielony o jasności 25%,
  - W jaki sposób kodowana jest przezroczystość koloru?
- (dotyczy realizacji zdalnej) obsługi kursora i zdalnego dostępu przez port szeregowy – głównie w *main.c*:

- **`putcursor`** – funkcja rysująca kursor na współrzędnych *cursorX* i *cursorY*, jeśli wcześniej została ustawiona zmienna *cursorUpd*;
- **`moveCursor`** – funkcja przesuwająca kursor o zadaną liczbę pikseli na X i Y; zawiera podstawowe zabezpieczenia przed przekroczeniem obszaru obrazu na wyświetlaczu.
- spróbujmy także sprawdzić, które klawisze wysyłane z terminala odpowiadają za przesuwanie kursora w górę, w dół, w lewo i w prawo oraz w którym miejscu znajduje się kod odpowiedzialny za odczyt „kliknięcia” spacją na aktualnych współrzędnych?

3.1 Funkcja inicjalizująca LCD

1. W pliku **`main.c`** mamy już utworzoną funkcję **`lcd_start`**, w której zostaje przeprowadzona inicjalizacja i konfiguracja wyświetlacza LCD na płytce STM32F746G-DISCO.
2. Korzystając dostępnego kodu funkcji, aplikacji demonstracyjnych, źródeł BSP: **`stm32746g_discovery_lcd.c`** i **`stm32746g_discovery_lcd.h`** oraz analizując dostępny kod proszę sprawdzić, co się dzieje w funkcji **`lcd_start`**; Przede wszystkim warto wiedzieć:
    - W jaki sposób wykonać inicjalizację sprzętu obsługującego wyświetlacz?
    - W jaki sposób zostaje wykonana inicjalizacja warstw obrazu i gdzie wskazywane są tablice **`lcd_image_fg`** i **lcd_image_bg** jako miejsca przechowywania danych obrazu warstwy przedniej (*foreground*) oraz tła (*background*)?
    - Które liczby odpowiadają indeksom warstw przedniej i tła?
    - Jak włączyć wyświetlacz LCD?
    - W jaki sposób dla warstwy przedniej oraz tła wykonać czyszczenie zawartości wyświetlacza oraz ustawienie koloru tła na biały?
    - W jaki sposób ustawić poziom przezroczystości tła oraz warstwy przedniej na zadane wartości i co one oznaczają, tj. jaka liczba oznacza brak przezroczystości, a jaka pełną przezroczystość?
    - Na jaki poziom przezroczystości ustawione są warstwy w dostępnym przykładzie?

3.2 Przygotowanie tła

Rysowanie tła powinno odbywać się w funkcji **`draw_background`** wywoływanej po *lcd_start* i dostarczonej jako pusta funkcja do uzupełnienia. W funkcji *draw_background* na warstwie tła proszę utworzyć kod rysujący obrazek składający się z kilku figur geometrycznych wypełnionych kolorem. Jako minimum proszę narysować i napisać:

- jeden lub więcej wypełniony kolorem prostokąt,
- jeden lub więcej wypełniony kolorem trapez,
- koło,
- dowolny, czytelny tekst.

Wskazówka: o kolorze rysowanej figury i tekstu decyduje parametr przekazany do funkcji **`BSP_LCD_SetTextColor`**.

3.3 Obsługa akcji kursora (dotyczy realizacji zdalnej)

W ramach tego punktu należy napisać fragment kodu w funkcji **`cursorPressAction`** pomiędzy

`//start performing a sample action`

a

`//end of the sample action.`

W tym fragmencie program powinien:

- uaktywnić/wybrać warstwę tła (używając np. stałej/makra **`LCD_LAYER_BG`**)
- ustawić kolor ciemnoniebieski (*dark blue*) jako aktywny kolor obiektu korzystając ze stałej dostępnej w pliku nagłówkowym sterownika LCD (**`stm32746g_discovery_lcd.h`**),
- narysować koło o środku na współrzędnych **`cursorX`**, **`cursorY`** i o promieniu wynoszącym połowę stałej **`CURSOR_MARGIN`** zdefiniowanej w **`main.c`**.

3.4 Dotyczy wyłącznie ćwiczenia w wersji stacjonarnej: obsługa panelu dotykowego

Inicjalizacja obsługi panelu dotykowego odbywa się w głównym zadaniu, we fragmencie:

```
uint8_t status = BSP_TS_Init(LCD_X_SIZE, LCD_Y_SIZE);
if( status != TS_OK )
  xprintf("TS init error\n");
```

Odczyt współrzędnych należy wykonywać cyklicznie, w pętli głównej zadania *mainTask*. Sugerowanym miejscem do implementacji odczytu panelu dotykowego jest linia komentarza:

```
  //a recommended line to read the touch screen :)
```

Przykładowy kod odczytu panelu dotykowego i następujące po nim uaktualnienie położenia kursora na wyświetlaczu mogą być zaimplementowane w następujący sposób:

```
  TS_StateTypeDef TS_State;
  BSP_TS_GetState(&TS_State);
  if( TS_State.touchDetected )
  {
    cursorX = TS_State.touchX[0];
    cursorY = TS_State.touchY[0];
    cursorUpd = 1;
  }
  putcursor();
```

Informacje o fakcie i miejscu dotknięcia panelu przekazywane są przez strukturę **`TS_State`**. Flaga **`cursorUpd`** (globalna) zostaje ustawiona, aby poinformować inne moduły, w tym funkcję wyświetlającą kursor, że zajdzie potrzeba przemieszczenia kursora. Współrzędne dotknięcia w powyższym kodzie zostają wpisane do zmiennych **`cursorX`** i **`cursorY`**. Należy ich użyć do wyświetlenia figury geometrycznej, która stanie się kursorem wskazującym, w którym miejscu

wyświetlacza nastąpiło dotknięcie panelu. Wspomnianym kursorem może być np. koło o środku znajdującym się na odczytanych z panelu współrzędnych i promieniu ok. 40 pikseli. Kursor należy wyświetlić na przedniej warstwie wyświetlacza. Przykładowa funkcja wyświetlająca kursor znajduje się także w funkcji **`putcursor`**.

3.5 Zmiana poziomu przezroczystości

Wskazówka: warto zacząć od sprawdzenia/dowiedzenia się, na czym polega technika *color keying*.

3.5.1 Wstępne informacje

Przy poziomie przezroczystości warstwy przedniej (*foreground*) ustawionym na „nieprzezroczysty" (255), proszę zmienić sposób wyświetlania kursora na „częściowo przezroczysty", np. o poziomie przezroczystości wynoszącym 50%. Proszę spróbować zastanowić się i ew. sprawdzić w praktyce, czy potrafimy uzyskać zadane przezroczystości kursora wynoszące: 25%, 50%, 75%, ok. 87,5% (zakładamy, że 100% przezroczystości to kursor całkowicie przezroczysty, więc także niewidoczny, natomiast 0% przezroczystości oznacza kursor całkowicie zasłaniający tło).

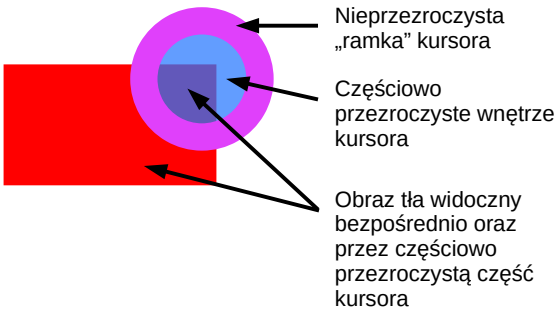
Wskazówka: na podstawie zawartości pliku

`/Drivers/BSP/STM32746G-Discovery/stm32746g_discovery_lcd.h`

oraz ew. własnych eksperymentów sprawdzić, w jaki sposób kodowana jest zawartość każdego piksela do wyświetlenia oraz przezroczystość kolorów.

3.5.2 Zadanie do wykonania

Proszę zaimplementować funkcjonalność wyświetlania kursora wg opisu. Kursor powinien być rysowany na warstwie przedniej (*foreground*) i składać się z dwóch elementów: 1) z elementu nieprzezroczystego oraz 2) półprzezroczystego. Przykład takiego kursora z elementem przezroczystym i półprzezroczystym został przedstawiony na rysunku poniżej. Aby uzyskać tego rodzaju wyświetlanie, **warstwa przednia powinna być skonfigurowana jako całkowicie nieprzezroczysta**, natomiast obszary półprzezroczyste powinny być generowane odpowiednim kodowaniem pikseli -częścią „łamigłówki" jest rozpoznanie, jak należy to zrobić ;-)



Na platformie UPEL umieszczamy kopię aktualnego stanu pliku **`main.c`** po modyfikacjach z niniejszego punktu, pod nazwą zmienioną na **`main_kursor.c`**. Ponadto proszę umieścić zdjęcie ekranu, na którym widać utworzony kursor nad obrazem tła.

3.6 Realizacja kreatywnego zadania

Używając funkcji dostępnych m.in. w module **`stm32746g_discovery_lcd.c`** należy napisać efektowny program wykorzystujący elementy interfejsu graficznego: wyświetlacz LCD i/lub interfejs sterujący: terminal (opcja zdalna ćwiczenia), panel dotykowy, mysz lub klawiaturę (opcje przy wykonywaniu ćwiczenia lokalnie). W ramach realizacji zadania można napisać np.:

- prostą grę,
- wizualizację prostego, ale ciekawego algorytmu (np. The game of life),
- animację wg zadanych statycznie lub dynamicznie (kursor, panel) parametrów.

Mile widziane jest pozytywne zaskoczenie osoby prowadzącej laboratorium kreatywnością zespołu w realizacji tego zadania.

W katalogu sprawozdania proszę umieścić aktualną kopię pliku **`main.c`** z nazwą zmodyfikowaną do **`main_kreatywne.c`**. Proszę także umieścić zdjęcie ekranu wyświetlającego efekt realizacji zadania kreatywnego.

3.7 Archiwizacja katalogu projektu

Katalog ze zmodyfikowanym projektem należy zarchiwizować, ponieważ może on być przydatny przy realizacji innych ćwiczeń laboratoryjnych oraz do ew. samodzielnych prac. Dla zmniejszenia rozmiaru archiwum należy usunąć pliki wynikowe (***make clean***). Nie jest zalecane pozostawianie projektu na komputerze w pracowni, ponieważ dyski tych komputerów mogą zostać nadpisane pomiędzy zajęciami bez uprzedniego ostrzeżenia.

4 Zagadnienia

Wymienione w niniejszej sekcji zagadnienia podsumowujące mają służyć do utrwalenia zdobytej wiedzy i umiejętności oraz do ułatwienia przygotowania się do sprawdzianu, zależnie od informacji organizacyjnych przekazanych przez osoby prowadzące. Zagadnienia można opracować samodzielnie na podstawie dostępnych materiałów i doświadczeń zdobytych na laboratorium.

1. **Warstwa sprzętowa – teoria i praktyka**. Zagadnienia technologiczne i sprzętowe związane z tematyką ćwiczenia (wskazówka: przydatne może być zapoznanie się z dokumentem UM1907 User manual, Discovery kit for STM32F7 Series with STM32F746NG MCU):
    a. Charakterystyka warstwy fizycznej interfejsu danych (obrazu) dla wyświetlacza LCD:
        - sygnały interfejsu: HSYNC, VSYNC, CLK – do czego służą, jaka jest koncepcja ich zastosowania?
        - przesyłanie danych graficznych: szerokości magistral dla poszczególnych kolorów oraz sposób przesyłania danych, m.in.
            1. równoległy czy szeregowy?
            2. synchroniczny czy asynchroniczny?

- sposób transmisji i synchronizacji danych: pikseli, linii, ramki przy tego typu interfejsach

2. **Część programowa – teoria i praktyka**. Rozumienie *co się dzieje* w programie. Znajomość poniższych przykładowych zagadnień na podstawie samodzielnej analizy kodu m.in. z udostępnionego projektu startowego, kodu napisanego w ramach realizacji ćwiczeń oraz dostępnej dokumentacji:
   a. Które moduły mikrokontrolera umożliwiają i wspomagają wyświetlanie obrazu na wyświetlaczu LCD w niniejszym ćwiczeniu przy zastosowaniu projektu *F7_LCD_HID*?
      - Jakie są podstawowe właściwości i funkcjonalność tych modułów?
   b. Gdzie (fizycznie w której pamięci i/lub w których zmiennych) w projekcie programowym *F7_LCD_HID* przechowywane są dane stanowiące obraz wyświetlany na LCD?
   c. Co robimy w funkcji inicjalizującej sterownik wyświetlacza? Umiejętność wskazania i zwięzłego opisania kilku czynności na podstawie zawartości następujących funkcji:
      - wysokopoziomowej funkcji **`lcd_start`** opracowanej w ramach realizacji ćwiczenia,
      - funkcji BSP wywoływanych w **`lcd_start`**.
   d. Format piksela i kodowanie barw w ćwiczeniu,
   e. Umiejętność scharakteryzowania oraz wskazania istotnych różnic pomiędzy sterownikami BSP i HAL,
   f. Sposoby uzyskania częściowej przezroczystości dla całej warstwy oraz dla fragmentu obrazu,

3. **Wiedza ogólna na temat wyświetlania obrazu cyfrowego**
   a. Umiejętność uzyskiwania zadanych barw i przezroczystości przy wskazanym kodowaniu formatu piksela w zakresie typowych kolorów: czerwony (R), zielony (G), niebieski (B), cyjan (C), magenta (M), żółty (Y), czarny oraz w skali szarości, w tym czarny i biały. Oczywiście nie ma konieczności rozpoznawania kolorów, należy przede wszystkim opanować (przynajmniej pamięciowo), z których składowych R, G, B i w jakich proporcjach utworzone są poszczególne kolory R, G, B, C, M, Y, czarny, biały i różne jasności koloru szarego. Nie musimy wiedzieć, jak wyglądają poszczególne kolory, a tym bardziej nie musimy także umieć ich rozpoznawać. Należy jednak wiedzieć, z jakich składają się komponentów R, G i B.
   b. Color keying – co to jest, do czego służy oraz w jaki sposób i do czego była wykorzystywana ta technika w realizacji ćwiczenia?
   c. Stosowanie więcej niż jednej warstwy w wyświetlaniu obrazu na LCD oraz przykładowe scenariusze użycia takiej konfiguracji: wyjaśnienie oraz umiejętność kreatywnego wymyślenia mających sens zastosowań dla dwóch warstw obrazu.