Systemy mikroprocesorowe

Instrukcja laboratoryjna

Spotkanie 2

Obsługa wyświetlaczy przez mikrokontroler

Autorzy: dr inż. Paweł Dąbal

Ostatnia aktualizacja: 23.11.2021 r.

Wersja: 1.0.0

Warszawa 2021

Spis treści

1. Wprowadzenie

Instrukcja ta zawiera zadania związane z tworzeniem programów obsługujących dwa popularne typy wyświetlaczy: siedmiosegmentowy oraz tekstowy LCD. W trakcie zajęć student będzie korzystał z środowiska programistycznego *STM32CubeIDE*, rozbuduje dostarczony projekt aplikacji dla mikrokontrolera STM32L496ZGT6 umieszczonego na płycie uruchomieniowej *KAmeLeon* o funkcjonalność umożliwiającą obsłużenie wyświetlaczy. Ponadto będzie potrafił obsługiwać narzędzia wspomagające kontrolę wersji oprogramowania w celu dokumentowania własnych postępów.

1.1. Cel ćwiczenia

Ćwiczenie laboratoryjne ma na celu:

- nabycie umiejętności konfiguracji stanowiska pracy w oparciu o oprogramowanie *STM32CubeIDE 1.7.0*;
- poznanie płyty uruchomieniowej *KAmeLeon* z mikrokontrolerem *STM32L496ZGT6*;
- przypomnienie i usystematyzowanie wiedzy i umiejętności z zakresu posługiwania się językiem *C*;
- przypomnienie wiedzy z zakresu budowy wyświetlaczy siedmiosegmentowych oraz tekstowych LCD;
- praktyczne korzystanie z systemu kontroli wersji *Git* i serwisu *GitHub*.

1.2. Wymagania wstępne

Przed przystąpieniem do wykonywania ćwiczenia laboratoryjnego należy :

- zapoznać się z schematem płyty uruchomieniowej *KAmeLeon* oraz dokumentacją mikrokontrolera *STM32L496ZGT6* (*Product Specifications*, *Reference Manuals*);
- zapoznać się z składnią języka C – pojęcie zmiennej, stałej, funkcji, prototypu funkcji, parametru funkcji, wyrażenia warunkowego, pętli, wskaźnik, struktury i tablicy;
- utworzyć konto na platformie *GitHub*;
- zapoznać się z dokumentacją środowiska *STM32CubeIDE* i biblioteki *STM32CubeL4*.

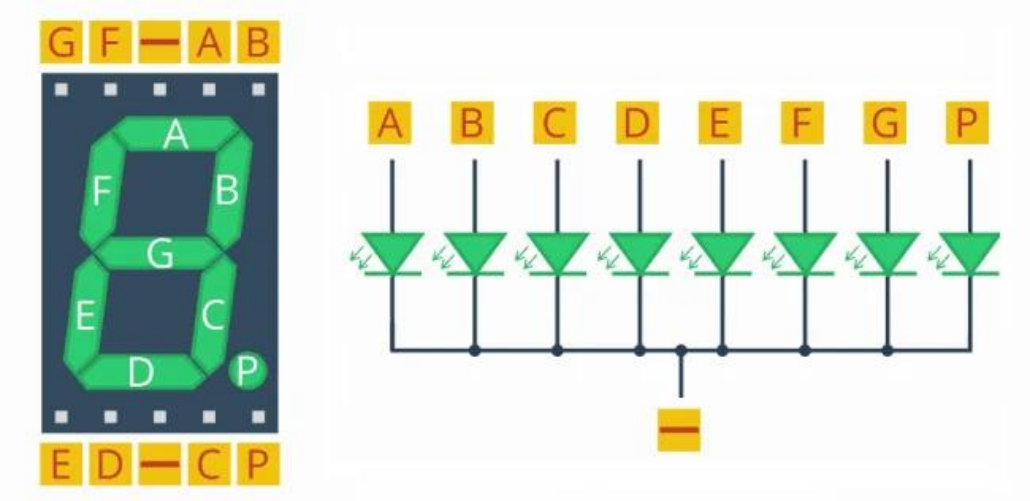
1.3. Opis stanowiska laboratoryjnego

Stanowisko do przeprowadzenia zajęć składa się z komputera PC z zainstalowanym oprogramowaniem koniecznym do realizacji zajęć: *STM32CubeIDE*, dedykowaną do układu biblioteką *STM32CubeL4* i systemem *GIt* oraz płyty *KAmeleon* podłączonej do komputera za pomocą przewodu USB do złącza programatora *SWD PRG/DBG/vCOM* płyty uruchomieniowej. Połączenie to również odpowiada za zasilanie płyty. Ponadto w trakcie zajęć użyty może być oscyloskop cyfrowy.

1.4. Wyświetlacze 7-segmentowe

Wyświetlacze tej rodziny są jednymi z najstarszych i najpopularniejszych rozwiązań. Są one montowane głównie w zegarach i cyfrowych multimetrach. Wyświetlacze siedmiosegmentowe mogą być wykonane w różnych technologiach – najczęściej jest to zestaw oddzielnych diod LED. Buduje się też urządzenia mechaniczne – zespół klapek w kolorze tła przysłaniających kontrastowe kreski (ang. *vane displays*) – zaletą układów mechanicznych jest możliwość prezentacji cyfr bez zasilania (niektóre wykonania). Nazwa pochodzi od siedmiu kresek, z których można zbudować znak. Najczęściej wyświetlają cyfry, ale mogą to być również niektóre litery. Segmenty w prezentowanym wyświetlaczu oznaczone są literami od A do G. Dodatkowo

występuje również kropka DP, oddzielający często część ułamkową liczby od części całkowitej. Wyświetlenie określonego znaku polega na załączeniu odpowiednich segmentów. Przykładowo cyfra 1 wymaga załączenia segmentów B i C, a cyfra 2 segmentów A, B, D, E, G. Każdy segment zawiera diodę (LED). Jednak wyprowadzenie każdej z nich oddzielnie wymagałoby użycia aż 14 nóżek (z kropką - 16). Dlatego producenci łączą diody wewnątrz wyświetlacza. Schemat wewnętrzny wyświetlacza OPD-S5621UPG-BW przedstawiony został poniżej. Wszystkie katody zostały połączone razem, do jednej nóżki. Właśnie tak jest zbudowany wyświetlacz ze wspólna katodą. Analogicznie, po połączeniu anod, powstałby wyświetlacz ze wspólną anodą. W związku z tym, że wyświetlacz to zbiór diod LED ich sterowanie niczym się nie różni od sterowania zwykłymi diodami. Jednakże dla wyświetlania cyfr i niektórych liter równolegle zapalanych jest wiele segmentów. Istnieją również 14- i 16- wyświetlacze segmentowe jednakże z upowszechnieniem się wyświetlaczy LCD są coraz rzadziej stosowane. W zależności od potrzeb stawianych przez system mikroprocesorowy, wyświetlacze mogą występować w formie 1-, 2- ,4-, a nawet 8-cyfrowej. Niezależnie od liczby cyfr, występujących w wyświetlaczu zasada działania w każdym przypadku jest taka sama.



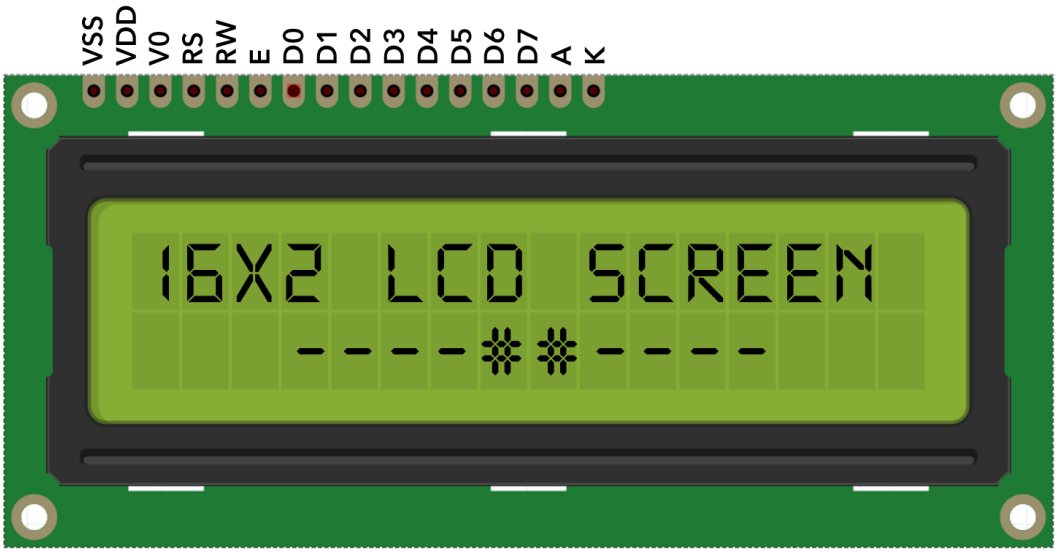
Dostępne są układy scalone sterowników dla tego typu wyświetlaczy np. CD4026, SN7448, które pozwalają na ograniczenie liczby wymaganych wyjść mikrokontrolera. Inny sposób to użycie rejestru przesuwnego np. 74HC595 kiedy to liczba wyjść mikrokontrolera zredukowana jest do trzech.

1.5. Wyświetlacze tekstowe LCD

Wyświetlacz ciekłokrystaliczny LCD pozwala na wyświetlanie oprócz cyfr, również liter oraz znaków graficznych. Każda litera składa się z wielu pikseli (w układzie 5x8), które poukładane są w prostokątach 16 w każdym, z dwóch wierszy. To jest właśnie wyświetlacz tekstowy LCD 16x2. Zapis ten oznacza, że w jednej chwili na ekranie możemy wyświetlić po 16 znaków w 2 wierszach. Jest to ograniczenie typowe dla wyświetlaczy tekstowych, służących do pokazywania napisów (i małych symboli). Inaczej byłoby w przypadku wyświetlaczy graficznych, tam mamy większą swobodę, ponieważ wszystkie piksele ułożone są w jednym prostokącie np.: 128x32. Dzięki temu możliwe jest dodatkowo rysowanie np.: linii, czy okręgów. Litery składają się z pixeli. Sterowanie każdym z osobna generowałoby dużą, a wręcz olbrzymią ilość linii sterujących. Oczywiście wszystko działa znacznie prościej, ponieważ wyświetlacze wyposażone są we wbudowane sterowniki. Najpopularniejszy z nich to HD44780.

Jak w takim razie należy przesyłać tekst do wyświetlacza? Konieczne jest podłączenie około 12 przewodów. Oczywiście tylko część z nich służy do komunikacji, pozostałe to zasilanie oraz inne sygnały, które nie zmieniają

się podczas pracy wyświetlacza. Najczęściej, ekran taki, jest wyposażony w 16-pinowe złącze. Piny (numerowane od lewej) od 1 do 3 służą do zasilania układu, od 4 do 14 do sterowania, zaś pod 15 i 16 znajduje się wewnętrzna dioda święcąca, która podświetla ekran. Wyświetlacze kompatybilne ze sterownikiem HD44780 mogą komunikować się z otoczeniem w trybie 4-bitowym oraz 8-bitowym. W pierwszym z nich konieczne jest 7 połączeń mikrokontrolera z wyświetlaczem. Natomiast w przypadku trybu 8-bitowego należy zrobić aż 11 połączeń. Korzystając z mniejszej ilości wyprowadzeń zachowamy praktycznie wszystkie opcje wyświetlacza.



No	Symbol	Function
1	VSS	Ground
2	VDD	5V +
3	V0	Contrast
4	RS	Register
5	RW	Read/Write
6	E	Enable
7	D0	Data bus
8	D1	Data bus
9	D2	Data bus
10	D3	Data bus
11	D4	Data bus
12	D5	Data bus
13	D6	Data bus
14	D7	Data bus
15	A	Anode (5V+)
16	K	Cathode (GND)

1.6. Sposób realizacji ćwiczenia laboratoryjnego

Każde zajęcia składać będą się z następujących elementów:

- zadań obowiązkowych - do wykonania krok po kroku na podstawie instrukcji – do uzyskania od 0 do 12 pkt.;
- zadań uzupełniających – do samodzielnego zbudowania i napisania programu – do uzyskania od 0 do 14 pkt;
- pytań sprawdzających rozumienie działania programu – zadawane przez prowadzącego przyjmującego wykonanie zadania – odpowiedź wpływa na punktację z zadań obowiązkowych i uzupełniających, tzn. czy przyznać maksymalną możliwą liczbę punktów za zadanie czy tylko część przy ewidentnym braku zrozumienia problemu.

Po zrealizowaniu każdego z zadań należy poprosić prowadzącego o sprawdzenie i przyznanie punktów. Na koniec zajęć wystawiana jest ocena na podstawie sumy uzyskanych punktów: **2,0** <0; 10), **3,0** <10; 13), **3,5** <13; 16), **4,0** <16; 19), **4,5** <19; 23), **5,0** <23; 26>. W każdych zajęciach należy uczestniczyć. Przy każdym zadaniu określona jest liczba punktów jakie można uzyskać. W przypadku zadań uzupełniających premiowana jest **jakość** i **czas realizacji** zaprezentowanego rozwiązania. Ocena końcowa z laboratorium to średnia arytmetyczna ocen z każdego spotkania.

2. Zadania podstawowe do realizacji (12 pkt.)

W tej części instrukcji zamieszczone są treści, z którymi obowiązkowo należy się zapoznać i praktycznie przećwiczyć. Ważne jest aby zapamiętać wykonywane przedstawione czynności aby móc na kolejnych zajęciach wykonywać je na kolejnych zajęciach bez potrzeby sięgania do niniejszej instrukcji.

Uwaga: Załączone wycinki z ekranu są poglądowe i pomagają jedynie w wskazaniu lokalizacji elementów interfejsu. Należy używać wartości podanych w tekście.

2.1. Zagadnienia wstępne

Przed przystąpieniem do pracy należy skonfigurować zainstalowane oprogramowanie i uzyskać dostęp do repozytorium, tzn. system kontroli wersji *Git*, instalacji biblioteki do obsługi mikrokontrolerów rodziny STM32L4 w środowisku *STM32CubeIDE* oraz sklonować repozytorium dla zajęć.

2.1.1. Konfiguracja lokalna oprogramowania *Git*

Przed wykonaniem pierwszych operacji z użyciem systemu *Git* należy skonfigurować dane autora wydając poniższe polecenia w wierszu poleceń systemu (np. *cmd.exe* w systemie Windows):

- ***git config --global user.name "Imię Nazwisko"*** – polecenie ustawia nazwę użytkownika;
- ***git config --global user.email "imie.nazwisko@student.wat.edu.pl"*** – polecenie ustawia adres e-mail.

Brak konfiguracji będzie widoczny jako podpisanie migawki (ang. *commit*) przez inną osobę niż ta która realizuje postawione w instrukcji zadania. W celu uniknięcia niejednoznaczności autorstwa przy rozliczaniu pracy należy pamiętać o sprawdzeniu ustawień (*git config—global user.name*, *git config—global user.email*).

2.1.2. Dołączenie do klasy i pozyskania repozytorium bazowego dla zadania

Na zajęciach należy dołączyć do wirtualnej grupy w ramach *Classrom GitHub* za pomocą udostępnionego odnośnika prowadzącego do zadania, po otwarciu którego tworzone jest zadanie – indywidualne repozytorium. Z listy należy wybrać swój adres e-mail/numer albumu i potwierdzić przyjęcie zadania (ang. *assignment*). Automatycznie zostanie utworzone prywatne repozytorium indywidualnie dla każdego studenta na podstawie przygotowanego repozytorium-szablonu. Na pierwszych zajęciach jest to wersja minimalna, a na kolejnych będzie zawierała już wstępnie skonfigurowane projekty. W sytuacji jeżeli student nie może odszukać się na liście (np. ktoś inny podłączył się pod daną osobę) proszę zgłosić to prowadzącemu zajęcia. W celu skorygowania nieprawidłowości. Zaakceptowanie zadania wiąże się również z dołączeniem do organizacji – wirtualnego konta organizacji w ramach którego tworzone są indywidualne repozytoria do zadań, z którego prowadzący zajęcia mają dostęp do wszystkich repozytoriów tworzonych w ramach zajęć.

W celu umożliwienia obsługi połączenia pomiędzy serwisem *GitHub* a aplikacją *STM32CubeIDE* potrzebne jest stworzenie osobistego klucza dostępu (ang. *personal access token*). Sposób jego przygotowania opisany został w Creating a personal access token. W kroku 6 należy podać nazwę tokena: *STM32CubeIDE*. Następnie czas wygaśnięcia zmienić na 30.02.2022 r. W kroku 8 należy zaznaczyć tylko pozycję *repo*.

Wygenerowany token należy zapisać w bezpiecznym miejscu tak aby w razie potrzeby móc go użyć np. na kolejnych zajęciach. Odświeżenie strony spowoduje, że nie będzie możliwości uzyskania dostępu do tokena.

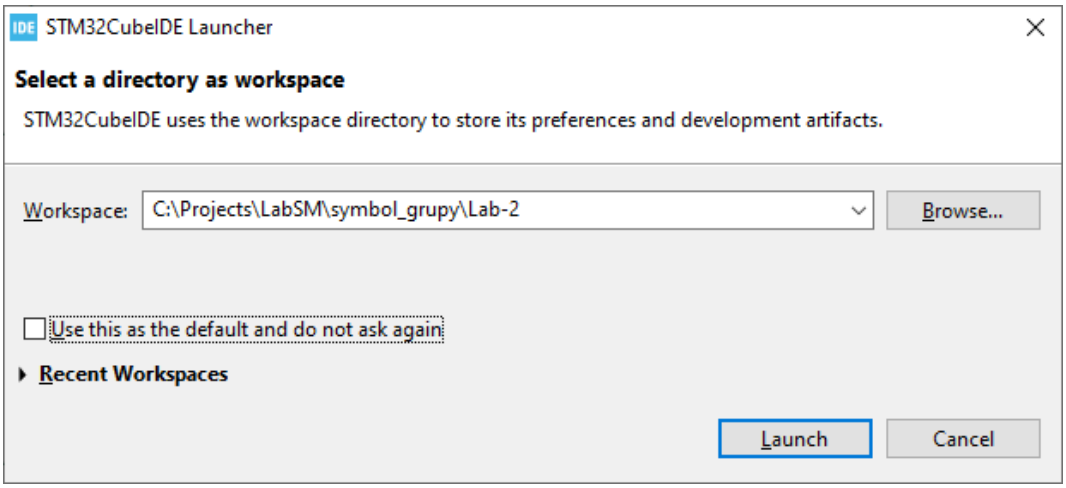
W wierszu poleceń systemu należy podać komendę inicjującym kopiowanie repozytorium z serwera GitHub na lokalny komputer we wskazanej lokalizacji:

git clone https://github.com/ztc-wel-wat/sm-lab-2-seg7-lcd-login.git C:\Projects\LabSM\symbol_grupy\Lab-2

W adresie repozytorium należy zwrócić uwagę na ***login*** gdyż dla każdego będzie on zgodny z nazwą użytkownika w serwisie *GitHub* oraz na *symbol_grupy*, które należy zmienić na własne. Należy podać login do serwisu i podać token PAT jako hasło.

2.1.3. Konfiguracja środowiska STM32CubeIDE

Po uruchomieniu środowiska programistycznego *STM32CubeIDE* pojawi się pytanie o wskazanie katalogu, który będzie pełnił rolę przestrzeni roboczej. Należy wskazać na katalog skopiowanego w poprzednim kroku repozytorium (w przykładzie: *C:\Projects\LabSM\symbol_grupy\Lab-2*). Wybór katalogu zatwierdzamy przyciskiem *Launch*. Można używać wielu różnych przestrzeni roboczych w przypadku pracy z różnymi projektami. Po załadowaniu środowiska należy zamknąć zakładkę *Information Center.*



W celu zarządzania repozytorium dla zadania należy w środowisku *STM32CubeIDE* należy otworzyć widok (ang. *perspective*) zarządzania repozytorium. W tym celu z menu wybieramy: *Window → Perspective → Open Perspective → Other…*, gdzie z listy należy wybrać *Git.* Nastąpi przełączenie widoku i po lewej stronie pojawi się zakładka *Git Repositories* z listą dostępnych repozytoriów.

Mechanizm migawek pozwala na zapisywanie chwilowego stanu projektu. Można cofnąć się do wcześniejszej fazy projektu, tworzyć rozgałęzienia, łączyć rozgałęzienia. W trakcie zajęć ograniczymy się do prostej liniowej struktury migawek wykonywanych po zakończeniu każdego z zadań instrukcji opatrzonych stosownym komentarzem. W dalszej części instrukcji będą podane treści komentarzy jakimi należy opatrzyć realizowane migawki. Powstanie zatem coś w rodzaju sprawozdania z zajęć, które będzie *przechowywane* w serwisie *GitHub*.

2.1.4. Opis projektu bazowego

W sklonowanym repozytorium znajduje się projekt bazowy korzystający z biblioteki HAL o nazwie *BasicDisplayHAL*. Posiada on skonfigurowane wyprowadzenia mikrokontrolera do których dołączone są diody LED (LED0 … LED7) oraz pięć styków joysticka joystick (*SW_RIGHT, SW_ LEFT, SW_ DOWN, SW_UP, SW_OK*). W pliku *gpio.c* zdefiniowane zostały podstawowe funkcję do obsługi diod LED oraz odczytu stanu joysticka.

2.2. Obsługa wyświetlacza 7-segmentowego (6 pkt.)

Do obsługi wyświetlacza 7-segmentowego posłużą wyprowadzenia mikrokontrolera PG0 (*SEG7_A*), PG1 (*SEG7_B*), PG2 (*SEG7_C*), PG3 (*SEG7_D*), PG4 (*SEG7_E*), PG5 (*SEG7_F*), PG6 (*SEG7_G*), PG9 (*SEG7_DP*),

PB2 (*SEG7_DIG1*), PB3 (*SEG7_DIG2*), PB4 (*SEG7_DIG3*), PB5 (*SEG7_DIG4*). Należy je skonfigurować do pracy jako cyfrowe wyjście i nadać etykiety zgodnie z opisem podanym w nawiasach. Poprawne nadanie etykiet jest bardzo ważne dla poprawności kompilacji dołączonego w dalszej części instrukcji kodu.

2.2.1. Obsługa jednego modułu – praca statyczna (3 pkt.)

Poprzez właściwe wysterowanie segmentów od A do G można zapalić je tak aby ich układ miał charakter cyfry. Oczywiście wymaga to stworzenia odpowiedniej tablicy podstawnikowej (*SEG_Bin2Dec*), która dla danej wartości cyfry wskaże jakie segmenty mają być ustawione w stan wysoki, a które w niski. Ponadto Dodać należy tablicę przechowującą numery fyprowadzeń do których są podłączone moduły (SEG_Module). Związku z tym w pliku *mian.c* należy dodać poniższy kod:

```
/* USER CODE BEGIN PV */
const uint8_t SEG_Bin2Dec[] =
{
    SEG7_A_Pin | SEG7_B_Pin | SEG7_C_Pin | SEG7_D_Pin | SEG7_E_Pin | SEG7_F_Pin,    // 0
    SEG7_B_Pin | SEG7_C_Pin,    // 1
    SEG7_A_Pin | SEG7_B_Pin | SEG7_D_Pin | SEG7_E_Pin | SEG7_G_Pin, // 2
    SEG7_A_Pin | SEG7_B_Pin | SEG7_C_Pin | SEG7_D_Pin | SEG7_G_Pin, // 3
    SEG7_B_Pin | SEG7_C_Pin | SEG7_F_Pin | SEG7_G_Pin,  // 4
    SEG7_A_Pin | SEG7_C_Pin | SEG7_D_Pin | SEG7_F_Pin | SEG7_G_Pin, // 5
    SEG7_A_Pin | SEG7_C_Pin | SEG7_D_Pin | SEG7_E_Pin | SEG7_F_Pin | SEG7_G_Pin,    // 6
    SEG7_A_Pin | SEG7_B_Pin | SEG7_C_Pin,   // 7
    SEG7_A_Pin | SEG7_B_Pin | SEG7_C_Pin | SEG7_D_Pin | SEG7_E_Pin | SEG7_F_Pin | SEG7_G_Pin,   // 8
    SEG7_A_Pin | SEG7_B_Pin | SEG7_C_Pin | SEG7_D_Pin | SEG7_F_Pin | SEG7_G_Pin,    // 9
};

const uint16_t SEG_Module[] = { SEG7_DIG4_Pin, SEG7_DIG3_Pin, SEG7_DIG2_Pin, SEG7_DIG1_Pin };
/* USER CODE END PV */
```

Do ustawienia wybranej cyfry na wyświetlaczu należy dodać funkcję *SEG_SetSegment*, która pozwoli na ustawienie wyprowadzeń mikrokontrolera. W pierwszej kolejności należy wygasić wszystkie segmenty, a następnie odpowiednie segmenty zapalić. Do wygaszenia wszystkich elementów posłuży zdefiniowany symbol *SEG7_Msk*. Ponadto potrzebna jest również funkcja *SEG_SetModule*, która aktywuje jeden z czterech (DIG1, DIG2, DIG3, DIG4) modułów wyświetlacza. Należy dodać do pliku *main.c* poniższy kod funkcji:

```
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
#define SEG7_Msk        (SEG7_A_Pin | SEG7_B_Pin | SEG7_C_Pin | SEG7_D_Pin | SEG7_E_Pin | SEG7_F_Pin | SEG7_G_Pin | SEG7_DP_Pin)
#define SEG7_MOD_Msk    (SEG7_DIG4_Pin | SEG7_DIG3_Pin | SEG7_DIG2_Pin | SEG7_DIG1_Pin)

/* USER CODE END PD */
```

```
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
void SEG_SetSegment(uint8_t value){
    value &= 0x0F;
    HAL_GPIO_WritePin(GPIOG, SEG7_Msk, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOG, SEG_Bin2Dec[value], GPIO_PIN_SET);
}

void SEG_SetModule(uint8_t module){
    module &= 0x03;
    HAL_GPIO_WritePin(GPIOB, SEG7_MOD_Msk, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOB, SEG_Module[module], GPIO_PIN_SET);
}
/* USER CODE END 0 */
```

Należy korzystając z przygotowanych funkcji uzupełnić funkcję główną *main* tak aby na wybranym module wyświetlacza 7-segmentowego prezentowały się kolejne cyfry co 500 ms. Przygotowany program należy

skompilować, uruchomić i przedstawić prowadzącemu do ocenienie. Po zatwierdzeniu wykonać migawkę z komentarzem „***Zadanie 2.2.1 – pojedynczy wyświetlacz 7-segmentowy***”.

2.2.2. Obsługa wszystkich modułów – praca multipleksowana (3 pkt.)

Z racji, że segmenty modułów wyświetlacza są ze sobą podłączone równolegle ustawienie stanu wysokiego na więcej niż jednym wyjściu *SEG7_DIG* powoduje wyświetlenie tej samej cyfry na każdym module. Aby temu zaradzić konieczne jest zaimplementowanie funkcjonalności ciągłego przełączania aktywnego segment. Umożliwia to ograniczenie liczby połączeń kosztem ograniczenia jasności i potencjalnego wystąpienia efektu migotania segmentów. Zasada działania wyświetlania multipleksowanego polega na cyklicznym oraz szybkim przełączaniem modułów, gdzie w danej chwili czasu włączony jest tylko jeden wyświetlacz spośród wszystkich dostępnych. Cykliczne oraz szybkie przełączanie modułów dzięki własności bezwładności wzroku, daje złudzenie jednocześnie załączonych wszystkich modułów. Obraz jest tym bardziej stabilny im szybciej przełączane (multipleksowane) są wyświetlacze. Przyjmuje się, że minimalna oraz akceptowalna częstotliwość przełączania każdego wyświetlacza powinna wynosić przynajmniej 25 Hz. Jeżeli natomiast zespół wyświetlaczy liczy 4 moduły 7-segmentowe, to częstotliwość przełączania powinna zostać zwiększona przynajmniej dwukrotnie. Aby móc przełączyć aktywny moduł i wysterować jego segmenty można użyć funkcji biblioteki *HAL_SYSTICK_Callback*. Jest ona domyślnie wykonywana przez mikrokontroler co 1 ms w przerwaniu od licznika *SysTick*. Każde upłynięcie okresu zdefiniowanego przez symbol *SEG_RefreshPeriodMs* powoduje wywołanie funkcji *SEG_Mux*, która ustawia segmenty dla danej cyfry (linia 93) oraz ustawia aktywny moduł (linia 94). Cyfry jakie mają być wyświetlone przechowywane są w tablicy *SEG_Value*.

```
88  #define SEG_RefreshPeriodMs     (5)
89
90  uint8_t SEG_Value[] = {0, 0, 0, 0};
91
92  void SEG_Mux(void){
93      static uint32_t module = 0;
94      SEG_SetSegment(SEG_Value[module]);
95      SEG_SetModule(module++);
96  }
97
98  void HAL_SYSTICK_Callback(void){
99      static uint16_t SEG_Delay = 0;
100     if((++SEG_Delay) == SEG_RefreshPeriodMs){
101         SEG_Mux();
102         SEG_Delay = 0;
103     }
104 }
```

W celu wyświetlenia liczby na kilku segmentach potrzebne jest jej rozłożenie na części: jedności, dziesiątki, setki i tysiące. W tym celu przygotowana została funkcja *SEG_DisplayDec*, która uzupełnia tablicę *SEG_Value* odpowiednimi cyframi. Jej kod przedstawiono poniżej.

```
106 void SEG_DisplayDec(uint16_t value){
107     if ((value >= 0) && (value < 10000)){
108         SEG_Value[0] = (value % 10);
109         value /= 10;
110         SEG_Value[1] = value ? (value % 10) : 0;
111         value /= 10;
112         SEG_Value[2] = value ? (value % 10) : 0;
113         value /= 10;
114         SEG_Value[3] = value ? (value % 10) : 0;
115     }
116 }
```

Należy korzystając z przygotowanych funkcji uzupełnić funkcję główną *main* tak aby wyświetlacza 7-segmentowego prezentowały kolejno różne liczby co 500 ms. Przygotowany program należy skompilować, uruchomić i przedstawić prowadzącemu do ocenienie. Po zatwierdzeniu wykonać migawkę z komentarzem „***Zadanie 2.2.2 – 7-segmentowy wyświetlacz multipleksowany***”.

2.3. Obsługa wyświetlacza tekstowego LCD (6 pkt.)

Biblioteka HAL nie dysponuje dedykowanymi funkcjami do obsługi wyświetlaczy LCD. Jednakże dostępne są biblioteki do ich obsługi. Udostępniają one szereg funkcji zapewniających pełną obsługę wyświetlacza. Sterowanie wyświetlaczem 16x2 polega na jego pierwotnej inicjalizacji, w której następuję konfiguracja sterownika i trybu jego pracy. Ponieważ wyświetlacz posiada wbudowaną pamięć znaków o rozmiarze przynajmniej 16x2 bajtów, w celu ciągłego wyświetlania znaków, nie jest konieczne cykliczne odświeżanie pamięci. Raz zapisany znak we wskazanym miejscu jest wyświetlany, aż do ponownej zmiany znaku lub zaniku napięcia zasiania. Wyświetlacz 16x2 umożliwia wyświetlenie znaków zgodne z tablicą ASCII [5]. Dodatkowo w zależności od kraju pochodzenia dostępne mogą być dodatkowe znaki (najczęściej chińskie). Ponadto wyświetlacz LCD ze sterownikiem HD44780 pozwala definiować własne znaki (np. ą, ć, ę, itp.), które można w późniejszym czasie używać.

2.3.1. Inicjalizacja i ustawianie treści do wyświetlenia (3 pkt.)

Niniejsze zadanie polega na wyświetlaniu napisu zawierającego w pierwszej linii swoje *Nazwisko* oraz cyklicznie narastającej wartości licznika w drugiej linii. Licznik ten inkrementowany jest co 500 ms. Należy sprawdzić działanie wyświetlacza w przypadku gdy ustawione zostaną współrzędne w zakresie wykraczającym poza zakres wyświetlacza.

```
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include "lcd.h"
#include <stdio.h>
#include <string.h>
/* USER CODE END Includes */

/* Initialize all configured peripherals */
  MX_GPIO_Init();
  /* USER CODE BEGIN 2 */
  LCD_Init();
  LCD_BacklightOn();
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  char LCD_Buffor[32];
  uint32_t cnt = 0;

  LCD_WriteText((uint8_t*) "Nazwisko");
  while (1) {
      sprintf(LCD_Buffor, "%d", (int) cnt++);
      LCD_WriteTextXY((uint8_t*) LCD_Buffor, strlen(LCD_Buffor), 2, 1);
      HAL_DMA_DeInit(500);
      /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
```

Przygotowany program należy skompilować, uruchomić i przedstawić prowadzącemu do ocenienie. Po zatwierdzeniu wykonać migawkę z komentarzem „***Zadanie 2.3.1 – Inicjalizacja i ustawianie treści do wyświetlenia***”.

2.3.2. Przemieszczanie się napisu na ekranie wyświetlacza (3 pkt.)

Zadanie to polega na przesuwaniu tekstu zawartego w tablicy znaków o nazwie *LCD_Buffor*. Należy zmodyfikować jej zawartość wpisując własne imię bez znaków diakrytycznych. Napis przesuwany jest w górnym wierszu od lewej do prawej strony, a następnie w dolnym wierszu od prawej do lewej. Napis nie może wychodzić poza wyświetlacz i musi zaczynać się zawsze od skrajnego położenia na wyświetlaczu. Dodatkowo, należy zadbać o to, aby przesuwany tekst nie pozostawiał po sobie śladów. Szybkość przesuwania napisu należy ustawić na ok. 500 ms.

```
/* Initialize all configured peripherals */
  MX_GPIO_Init();
  /* USER CODE BEGIN 2 */
  LCD_Init();
  LCD_BacklightOn();
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  char LCD_Buffor[32];
  sprintf(LCD_Buffor, "Imie");

  while (1) {
      for (uint32_t row = 1; row < 3; row++) {
          if (row == 1) {
              for (uint32_t col = 1; col <= (17 - strlen(LCD_Buffor));
                      col++) {
                  LCD_WriteTextXY((uint8_t*) LCD_Buffor, strlen(LCD_Buffor),
                          row, col);
                  HAL_Delay(500);
                  LCD_Clear();
              }
          } else {
              for (uint32_t col = (17 - strlen(LCD_Buffor)); col > 0; col--) {
                  LCD_WriteTextXY((uint8_t*) LCD_Buffor, strlen(LCD_Buffor),
                          row, col);
                  HAL_Delay(500);
                  LCD_Clear();
              }
          }
      }
      /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */
  }
```

Przygotowany program należy skompilować, uruchomić i przedstawić prowadzącemu do ocenienie. Po zatwierdzeniu wykonać migawkę z komentarzem „***Zadanie 2.3.2 – Przemieszczanie się napisu na ekranie wyświetlacza***”.

3. Zadania rozszerzające do realizacji (14 pkt.)

W rozdziale tym przedstawione zostały zadania dodatkowe, za realizację których można podnieść ocenę końcową za wykonane ćwiczenie. Ich realizację należy wykonać w dotychczasowym projekcie.

3.1. Sterowanie wyświetlaczem 7-segmentowym (7 pkt.)

Zadania uzupełniające będą polegać na rozszerzeniu dotychczasowej funkcjonalności przygotowanych wcześniej funkcji.

3.1.1. Prezentacja całkowitych liczb bez wiodącego zera (1 pkt.)

Należy działanie funkcji *SEG_DisplayDec* tak aby umożliwiła prezentowanie liczby bez dodawania zera na bardziej znaczących pozycjach. Należy korzystając z przygotowanych funkcji uzupełnić funkcję główną *main* tak aby zaprezentować funkcjonalność wprowadzonych zmian. Przygotowany program należy skompilować, uruchomić i przedstawić prowadzącemu do ocenienie. Po zatwierdzeniu wykonać migawkę z komentarzem „***Zadanie 3.1.1 – Prezentacja całkowitych liczb bez wiodącego zera***”.

3.1.2. Dodanie możliwości prezentacji liczb całkowitych ujemnych (2 pkt.)

Należy rozszerzyć działanie funkcji *SEG_DisplayDec* tak aby umożliwiła prezentowanie liczby ujemnych w zakresie od -1 do -999. W tym celu należy rozbudować istniejące wyrażenie warunkowe. Należy korzystając z przygotowanych funkcji uzupełnić funkcję główną *main* tak aby zaprezentować funkcjonalność wprowadzonych zmian. Przygotowany program należy skompilować, uruchomić i przedstawić prowadzącemu do ocenienie. Po zatwierdzeniu wykonać migawkę z komentarzem „***Zadanie 3.1.2 – Dodanie możliwości prezentacji liczb całkowitych ujemnych***”.

3.1.3. Dodanie możliwości prezentacji liczb rzeczywistych (2 pkt.)

Należy zdefiniować funkcję *SEG_DisplayFixedPoint* tak aby umożliwiła prezentowanie liczby rzeczywistych z określoną za pomocą parametru dokładnością (tzn. liczbą miejsc po przecinku. Należy korzystając z przygotowanych funkcji uzupełnić funkcję główną *main* tak aby zaprezentować funkcjonalność wprowadzonych zmian. Przygotowany program należy skompilować, uruchomić i przedstawić prowadzącemu do ocenienie. Po zatwierdzeniu wykonać migawkę z komentarzem „***Zadanie 3.1.3 – Dodanie możliwości prezentacji liczb rzeczywistych***”.

3.1.4. Dodanie możliwości prezentacji liczb w formacie szesnastkowym (2 pkt.)

Należy dodać funkcję *SEG_DisplayHex* tak aby umożliwiła prezentowanie liczby w formacie szesnastkowym. W tym celu należy rozbudować tablicę *SEG_Bin2Seg* o dodatkowe wpisy, które będą prezentowały cyfry zapisu szesnastkowego: *A*, *b*, *c*, *d*, *E*, *F*. Należy korzystając z przygotowanych funkcji uzupełnić funkcję główną *main* tak aby zaprezentować funkcjonalność wprowadzonych zmian. Przygotowany program należy skompilować, uruchomić i przedstawić prowadzącemu do ocenienie. Po zatwierdzeniu wykonać migawkę z komentarzem „***Zadanie 3.1.4 – Dodanie możliwości prezentacji liczb w formacie szesnastkowym***”.

3.2. Sterowanie wyświetlaczem tekstowym LCD (7 pkt.)

Zadania uzupełniające będą polegać na rozszerzeniu dotychczasowej funkcjonalności przygotowanych wcześniej funkcji.

3.2.1. Częsta zmiana zawartości prezentowanej zawartości na wyświetlaczu - stoper (dodatkowe – 7 pkt.)

Zadanie to polega na utworzeniu stopera oraz licznika upływu czasu od uruchomienia systemu. Stoper załączany oraz wyłączany jest przy pomocy przycisku *SW_OK*. Jednokrotne naciśniecie przycisku rozpoczyna pomiar czasu i jego upływ jest wyświetlany w pierwszej linii wyświetlacza. Kolejne naciśnięcie – kończy i zatrzymuje wynik pomiaru. Pomiar upływu czasu od uruchomienia układu prezentowany jest w drugiej linii wyświetlacza. Prezentowany czas należy podać zgodnie z poprawnym formatem, tj. *hh*:*mm*:*ss*.

Należy korzystając z przygotowanych funkcji uzupełnić funkcję główną *main* tak aby zaprezentować funkcjonalność wprowadzonych zmian. Przygotowany program należy skompilować, uruchomić i przedstawić prowadzącemu do ocenienie. Po zatwierdzeniu wykonać migawkę z komentarzem „***Zadanie 3.2.1 – Stoper***”.