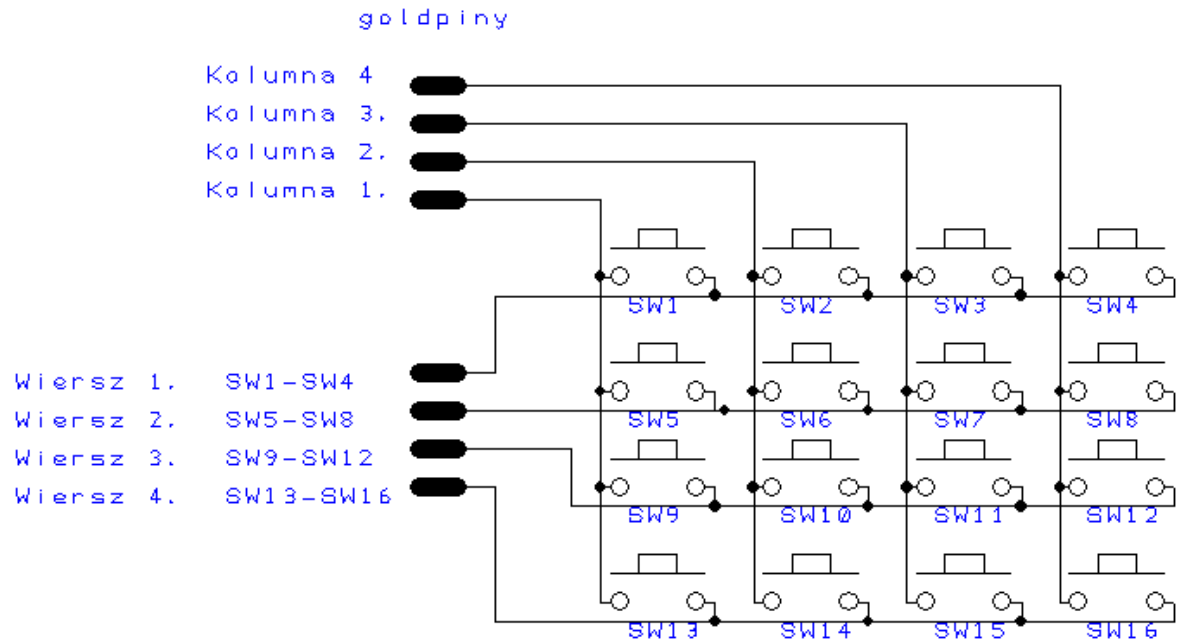
Klawiatura matrycowa

Na rys. 1. został przedstawiony schemat klawiatury matrycowej. W celu wykorzystania 4 przycisków znajdujących się w jednej kolumnie wystarczy podłączyć wybraną kolumnę do GND, a wiersze W1,W2,W3,W4 do czterech linii portu mikrokontrolera, które powinny być skonfigurowane jako wejścia z włączonymi rezystorami podciągającymi (ang. Pull up) do napięcia zasilania VCC. Jeżeli żaden przycisk nie będzie wciśnięty, to na wszystkich wierszach będą stany wysokie wynikające z włączonych wewnętrznych rezystorów pull up. Jeżeli podłączymy kolumnę K1 do GND, to po naciśnięciu przycisku SW5 zewrze on K1 do W2, w wyniku czego napięcie na W2 zmieni się z napięcia zasilania VCC na 0V. Stan wszystkich przycisków można łatwo określić, jeżeli kolumny K1,K2,K3,K4 połączy się do wyprowadzeń mikrokontrolera skonfigurowanych jako wyjścia. W celu sprawdzenia przycisków w dowolnej kolumnie ustawia się na niej stan niski, a na pozostałych kolumnach stan wysoki, następnie odczytuje się wszystkie wiersze. Jeżeli w którymś wierszu pojawi się stan niski, to znaczy, że naciśniety jest przycisk w sprawdzanej kolumnie. Po sprawdzeniu czterech kolumn znany jest stan wszystkich przycisków. Do obsługi 16 przycisków wystarczy 8 linii jednego port mikrokontrolera.

Wskazówka: Za pomocą operatora przesunięcia bitowego można w pętli generować odpowiednie stany kolumn.



Rys. 1 Schemat klawiatury matrycowej

W pliku main.h (katalog Core/Inc) jest m.in. Poniższy wiersz:

```
#define B1_Pin GPIO_PIN_13
```

Po naciśnięciu przycisku Ctrl I kliknięciu na GPIO_PIN_13 w main.h nastąpi otwarcie pliku stm32fxx_hal_gpio.h, w którym znajdują się definicje wszystkich pinów:

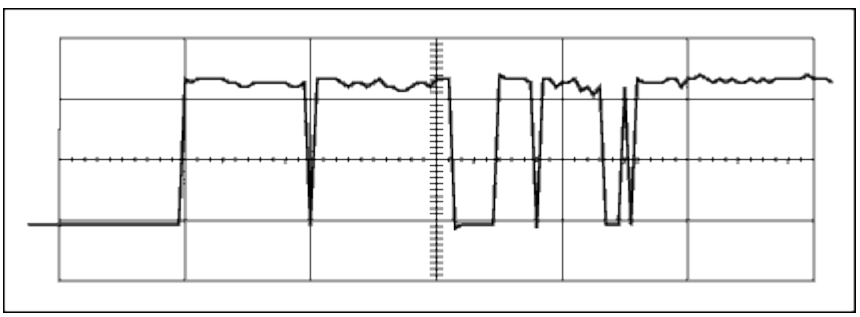
```
#define GPIO_PIN_0                 ((uint16_t)0x0001U)  /* Pin 0 selected    */
#define GPIO_PIN_1                 ((uint16_t)0x0002U)  /* Pin 1 selected    */
#define GPIO_PIN_2                 ((uint16_t)0x0004U)  /* Pin 2 selected    */
#define GPIO_PIN_3                 ((uint16_t)0x0008U)  /* Pin 3 selected    */
#define GPIO_PIN_4                 ((uint16_t)0x0010U)  /* Pin 4 selected    */
#define GPIO_PIN_5                 ((uint16_t)0x0020U)  /* Pin 5 selected    */
#define GPIO_PIN_6                 ((uint16_t)0x0040U)  /* Pin 6 selected    */
#define GPIO_PIN_7                 ((uint16_t)0x0080U)  /* Pin 7 selected    */
#define GPIO_PIN_8                 ((uint16_t)0x0100U)  /* Pin 8 selected    */
#define GPIO_PIN_9                 ((uint16_t)0x0200U)  /* Pin 9 selected    */
#define GPIO_PIN_10                ((uint16_t)0x0400U)  /* Pin 10 selected   */
#define GPIO_PIN_11                ((uint16_t)0x0800U)  /* Pin 11 selected   */
#define GPIO_PIN_12                ((uint16_t)0x1000U)  /* Pin 12 selected   */
#define GPIO_PIN_13                ((uint16_t)0x2000U)  /* Pin 13 selected   */
#define GPIO_PIN_14                ((uint16_t)0x4000U)  /* Pin 14 selected   */
#define GPIO_PIN_15                ((uint16_t)0x8000U)  /* Pin 15 selected   */
#define GPIO_PIN_All               ((uint16_t)0xFFFFU)  /* All pins selected */
```

Znalezienie ich definicji jest również możliwe po nacisnąć przycisk Ctrl i wybraniu odpowiedniej opcji z menu kontekstowego, które pojawi się po kliknięciu prawym przyciskiem myszy na GPIO_PIN_13
Proszę zwrócic uwagę, jakie wartości mają kolejne piny w zapisie binarnym:

GPIO_PIN_0 00000000 00000001
GPIO_PIN_1 00000000 00000010
GPIO_PIN_2 00000000 00000100
GPIO_PIN_3 00000000 00001000
…

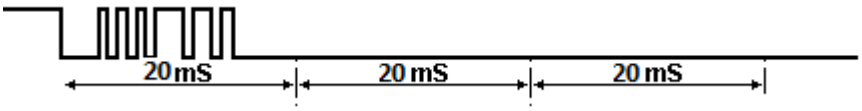
Funkcja HAL_GPIO_WritePin (PORTx, GPIO_PIN_y, state) umożliwia równocześnie ustawienie lub wyzerowanie kilku linii portu PORTx, (gdzie x oznacza literę portu: A,B,C lub D), np.
poniższ sekwencja instrukcji spowoduje ustawienie stanu wysokiego na trzech liniach portu B mikrokontrolera:
HAL_GPIO_WritePin (PORTB, GPIO_PIN_1 | PORTB, GPIO_PIN_2 | PORTB, GPIO_PIN_3, SET)
Operator przesunięcia bitowego `o x pozycji (1<<x) można wykorzystać w pętli w zadaniu 4. do ustawienia stanu niskiego na odpowiedniej linii mikrokontrolera podłączonej do klawiatury matrycowej.

Istotny problem stanowią drgania styków. Na rysunku 2. przedstawiono przykładowy oscylogram.



Rys. 2 Oscylogram drgań styków w przeciągu 6ms (źródło https://www.maximintegrated.com/en/app-notes/index.mvp/id/287)

Na różnorodne sposoby można starać się pozbyć negatywnych skutków drgań (ang. debouncing). Najprostszy (nie znaczy, że najlepszy) polega na kilkukrotnym sprawdzeniu stanu w ustalonych odstępach czasu po wykryciu zmiany z 1 na 0 (rys. 3).



Rys. 3 Kilkukrotne sprawdzenie stanu

Drgania można odfiltrować np. za pomocą prostego filtru dolnoprzepustowego RC. Filtry RC zastosowano do przycisków zamontowanych w nucleo. Można też użyć dedykowanych układów scalonych. Stosowanie dodatkowych komponentów może być niemożliwe w układach, w których kluczowy jest mały rozmiar. Czas drgań zależy od modelu przycisku.

Wysyłanie informacji przez UART

W celu wysyłania informacji przez UART do komputera PC, należy wykorzystać konwerter USB/UART. W zakładce Connectivity (w Device configuration Tool) należy włączyć USART1 I skonfigurować go następująco:
Baud Rate 38400 bit/s
Word Length 8 bits
Parity None
Stop Bits 1

Identyczne ustawienia należy wprowadzić na komputerze w programie
RealTerm https://sourceforge.net/projects/realterm/
lub
TeraTerm https://ttssh2.osdn.jp/index.html.en

Proszę znaleźć wyprowadzenie mikrokontrolera USART1_TX (PC4) i podłaczyć je do wyprowadzenia RX konwertera USB/USART. Dodatkowo należy połączyć masę z nucleo i konwertera.

Do wysyłania napisów można wykorzystać funckcję HAL_UART_Transmit wraz z instrukcję sprintf
W celu użycia sprintf należy dołączyć plik nagłowkowy stdio.h (#include <stdio.h>)
Dla funkcji strlen potrzebny jest #include <string.h>.
Przykład

```
sprintf (buf,"i=%d \r\n",i);
HAL_UART_Transmit(&huart1, (uint8_t*)buf, strlen(buf), HAL_MAX_DELAY);
gdzie tablicę buf mozna utworzyć np. następująco:
char buf[30];
```

Zadania do wykonania:

1) zapalenie diody LED po naciśnięciu SW1; zgaszenie jej po naciśnięciu SW5

2) zmiana stanu LED po naciśnięciu SW3 (należy uwzględnić drgania styków)

3) zliczanie naciśnięć SW2 (należy uwzględnić drgania styków). Liczbę nacisnięć należy przedstawić w postaci binarnej za pomocą diód LED wlutowanych w płytkę sterownika silnika krokowego (silnik powinien być odłączony) . GND I 5Vz modułu silnika należy podłączyć z GND I 5V na nucleo. Podanie stanu niskiego na IN1, IN2, IN, IN4 powoduje zapalenie LED. Jeżeli numer przycisku wynosi 1 (001 binarnie) to powinna się świecić tylko dioda IN1. Dla przycisku 3 (0011bin) powinny się świecić diody IN2, IN1. Dla przycisku numer 15 (0011bin) powinny się świecić wszystkie diody LED. Przedstawianie liczb w postaci binarnej będzie wykorzystywane w innch zadań, więc zalecane jest utworzenie funkcji mającej argument typu uint8_t, którego 4 mniej znaczące bity będą pokazywane na diodach LED.

3) obsługa wszystkich przycisków – numer przycisku należy binarnie pokazać na linijce diodowej z modułu silników krokowych (można zignorować drgania styków). Silnik należy odłaczyć.

Uwaga – w zadanich 1-4 liczba wykorzystywanych przycisków jest mniejsza niż 4, więc można odpowiednią kolumnę macierzy klawiszy podłączyć przewodem do masy.

Przed przystąpieniem do ćwiczeń należy:

a) znać sposób konfiguracji portów w STM32CubeIDE oraz wiedzieć jak się odczytuje I zmienia stan portów za pomocą funkcji HAL_GPIO_READPin HAL_GPIO_WritePin. Informacje te można znaleźć np. na https://forbot.pl/blog/kurs-stm32l4-wejscia-wyjscia-czyli-gpio-stm32-id46571
b) zapoznać się ze schematem klawiatury matrycowej
c) zapoznać się ze zjawiskiem drgania styków

Warto również: przeczytać w Internecie o drganiach styków - przeanalizować rozwiązania sprzętowe i programistyczne (ang. debouncing). Można wyprowadzić wzór na częstotliwość graniczną filtru dolnoprzepustowego utworzonego z pojedynczego kondensatora i rezystora.