Autor: Michał Tokarczyk

Sprawozdanie Systemy Wbudowanie

Opis projektu:

Projekt to gra zręcznościowa, która polega na jak najszybszym klikaniu wbudowanego przycisku w momencie pojawienia się zielonego kółka, po 10 kliknięciach gra się kończy i jest podany wynik czasowy z dokładniością do milisekund.

Jak zagrać:

W momencie uruchomienia gry na płytce uruchamia się komunikat “Start gry!” i po chwili gra się uruchamia i przy użyciu wbudowanego przycisku klikamy jak najszybciej w momencie pojawienia się zielonego kółka, po 10 kółkach następuje koniec gry i wyświetla się komunikat “Koniec gry! Czas:xx.xx s”, gdzie xx to nasz czas np: “Koniec gry! Czas:7.43 s”. Aby zresetować grę i zagrać ponownie należy kliknąć wbudowany przycisk reset na płytce.

Najważniejsze fragmenty kodu:

1.Ekran Startowy:

- BSP_LCD_DisplayStringAt(0, 10, (uint8_t*)"Start gry!", CENTER_MODE); - Wyświetla komunikat powitalny “Start gry!” na LCD.
- HAL_Delay(1000); - Czeka 1 sekundę.
- BSP_LCD_Clear(LCD_COLOR_BLACK); - Czyści ekran

2.Logika gry:

- start_time = HAL_GetTick(); - Zapisuje czas rozpoczęcia gry.
- while (hits < MAX_HITS) { - Gra trwa, dopóki gracz nie kliknie 10 razy.
- if (HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0) == GPIO_PIN_SET) { - Sprawdza, czy przycisk został naciśnięty.
- HAL_GPIO_WritePin(GPIOG, GPIO_PIN_14, GPIO_PIN_SET); - Włącza diodę LED.
- uint16_t x = rand() % (240 - 2 * CIRCLE_RADIUS) + CIRCLE_RADIUS; - Losowa pozycja X okręgu.
- uint16_t y = rand() % (240 - 2 * CIRCLE_RADIUS) + CIRCLE_RADIUS; - Losowa pozycja Y okręgu.

- DrawCircle(x, y, LCD_COLOR_GREEN); - Rysuje okrąg w losowym miejscu.
- hits++; - Zwiększa liczbę trafień.
- HAL_Delay(700); - Opóźnienie, aby okrąg nie był zbyt szybki.
- HAL_GPIO_WritePin(GPIOG, GPIO_PIN_14, GPIO_PIN_RESET); - Wyłącza diodę LED.
- BSP_LCD_Clear(LCD_COLOR_BLACK); - Czyści ekran, przygotowując go na nowy okrąg.

3.Obliczanie i wyświetlenie wyniku:

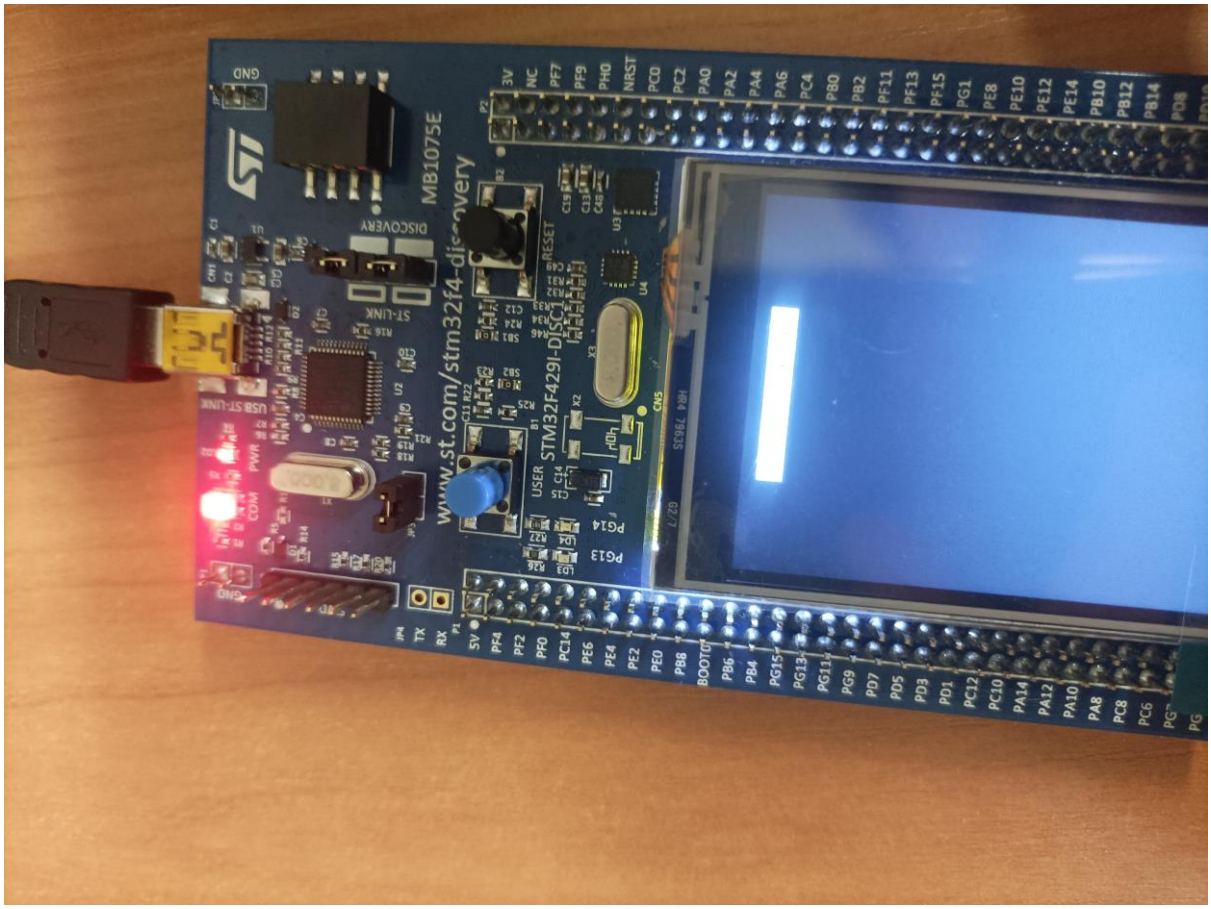
- end_time = HAL_GetTick(); - Zapisuje czas zakończenia gry.
- uint32_t total_time_ms = end_time - start_time; - Oblicza całkowity czas gry w milisekundach.
- uint32_t seconds = total_time_ms / 1000; - Przekształca czas na sekundy.
- uint32_t miliseconds = (total_time_ms % 1000) * 1000; - Oblicza pozostałe milisekundy.
- sprintf(buffer, "Koniec gry! Czas:%lu.%03lu s", seconds, miliseconds / 1000); - Wyświetla końcowy wynik.
- BSP_LCD_Clear(LCD_COLOR_BLACK); - Czyści ekran.
- BSP_LCD_SetTextColor(LCD_COLOR_RED); - Ustawia kolor tekstu na czerwony.
- BSP_LCD_DisplayStringAt(0, 10, (uint8_t*)buffer, LEFT_MODE); - Wyświetla wynik na ekranie.

4.Funkcja pomocnicza:

- void DrawCircle(uint16_t x, uint16_t y, uint32_t color) { BSP_LCD_SetTextColor(color); - Ustawia kolor tekstu.
- BSP_LCD_FillCircle(x, y, CIRCLE_RADIUS); - Rysuje wypełniony okrąg.
- }

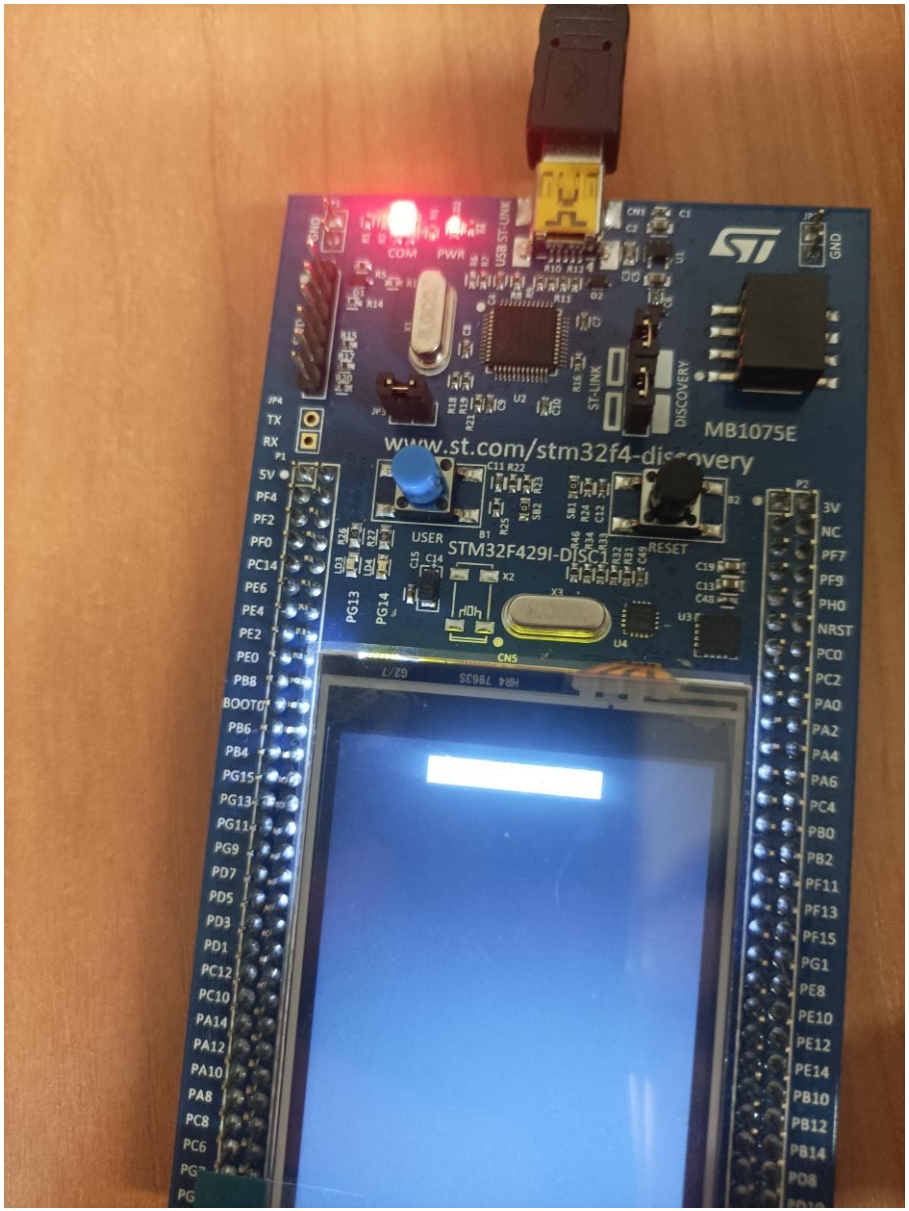
Zdjęcia działania gry(niestety przez małą czcionkę nie widać dobrze Startu i końca gry):

Start gry:



Środek gry:

Koniec gry:


COM
PWR
USB ST-LINK
ST-LINK
DISCOVERY
MB1075E
www.st.com/stm32f4-discovery
USER
STM32F429I-DISC1
RESET
CN5
GND
3V
NC
PF7
PF9
PH0
NRST
PC0
PC2
PA0
PA2
PA4
PA6
PC4
PB0
PB2
PF11
PF13
PF15
PG1
PE8
PE10
PE12
PE14
PB10
PB12
PB14
PD8
5V
PF4
PF2
PF0
PC14
PE6
PE4
PE2
PE0
PB8
BOOT0
PB6
PB4
PG15
PG13
PG11
PG9
PD7
PD5
PD3
PD1
PC12
PC10
PA14
PA12
PA10
PA8
PC8
PC6
TX
RX
PG13
PG14