

Mikroprocesory i mikrokontrolery Laboratorium nr 7	Temat: Zapoznanie się ze środowiskiem CUBE, obsługa portów I/O
Grupa: 21b	Michał Lechowicz

Cel ćwiczenia:

Celem ćwiczenia było zapoznanie się z działaniem i obsługą portów wejścia/wyjścia w mikrokontrolerze **STM32F0**. Mikrokontroler oparty jest o technologię ARM. Porty I/O służą do komunikacji z urządzeniami peryferyjnymi. Do konfiguracji kontrolera posłużył program CubeMX oraz IDE IAR.

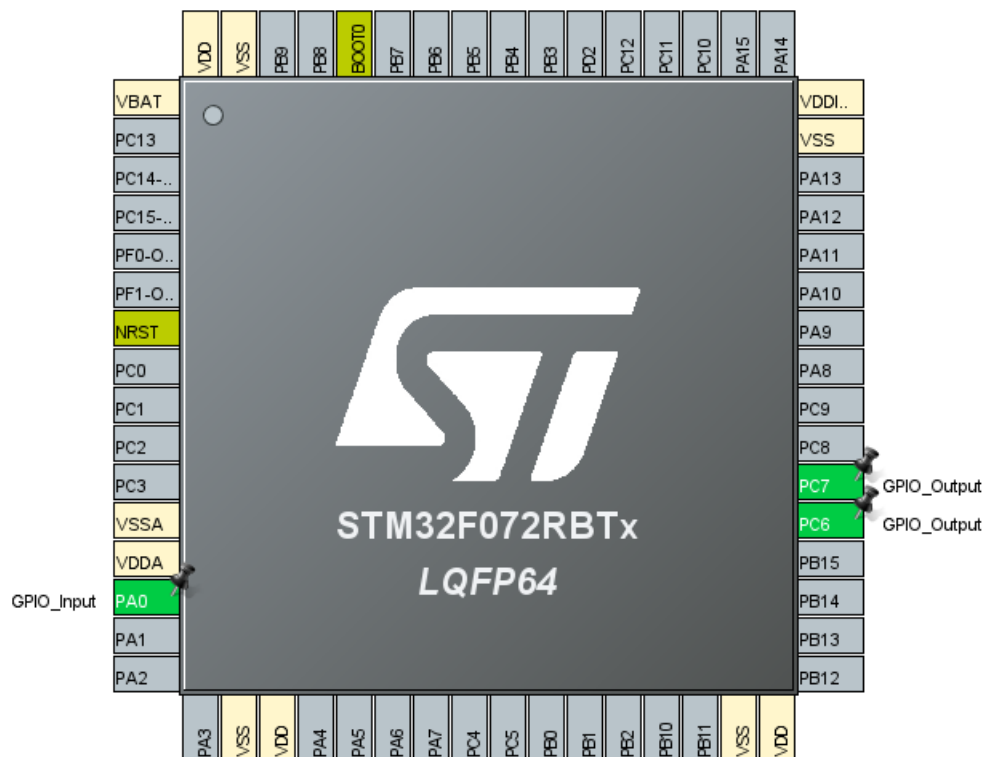
Zadanie na ocenę 5.0:

1. Opis zadania:

W zadaniu należało napisać program, w którym wykorzystamy dwie diody LED oraz wbudowany przycisk „USER”. Program przy starcie uruchamia jedną migającą diodę LED. Po wciśnięciu przycisku uruchamia się druga dioda (i miga), a pierwsza gaśnie. Kolejne wciśnięcie przycisku zamienia je miejscami.

2. Opis programu:

1. Konfiguracja programu STM32CubeMX;



Rys. 1 Pinout View

Pin Name	Signal on Pin	GPIO output level	GPIO mode	GPIO Pull-up/Pull-down	Maximum output s...	Fast Mode	User Label	Modified
PA0	n/a	n/a	Input mode	No pull-up and no pull-down	n/a	n/a		<input type="checkbox"/>
PC6	n/a	Low	Output Push Pull	No pull-up and no pull-down	High	n/a		<input checked="" type="checkbox"/>
PC7	n/a	Low	Output Push Pull	No pull-up and no pull-down	High	n/a		<input checked="" type="checkbox"/>

Rys. 2 Konfiguracja GPIO

2. Konfiguracja portów wykonana przez Cube;

```
main.c x
main()
162     GPIO_InitTypeDef GPIO_InitStruct = {0};
163
164     /* GPIO Ports Clock Enable */
165     __HAL_RCC_GPIOA_CLK_ENABLE();
166     __HAL_RCC_GPIOC_CLK_ENABLE();
167
168     /*Configure GPIO pin Output Level */
169     HAL_GPIO_WritePin(GPIOC, GPIO_PIN_6, GPIO_PIN_SET);
170
171     /*Configure GPIO pin : PA0 */ /* konfiguracja pinu odpowiedzialnego za przycisk */
172     GPIO_InitStruct.Pin = GPIO_PIN_0;
173     GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
174     GPIO_InitStruct.Pull = GPIO_NOPULL;
175     HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
176
177     /*Configure GPIO pins : PC6 PC7 */ /* Konfiguracja pinow odpowiedzialnych za diody */
178     GPIO_InitStruct.Pin = GPIO_PIN_6|GPIO_PIN_7;
179     GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
180     GPIO_InitStruct.Pull = GPIO_NOPULL;
181     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
182     HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
183
184 }
185
```

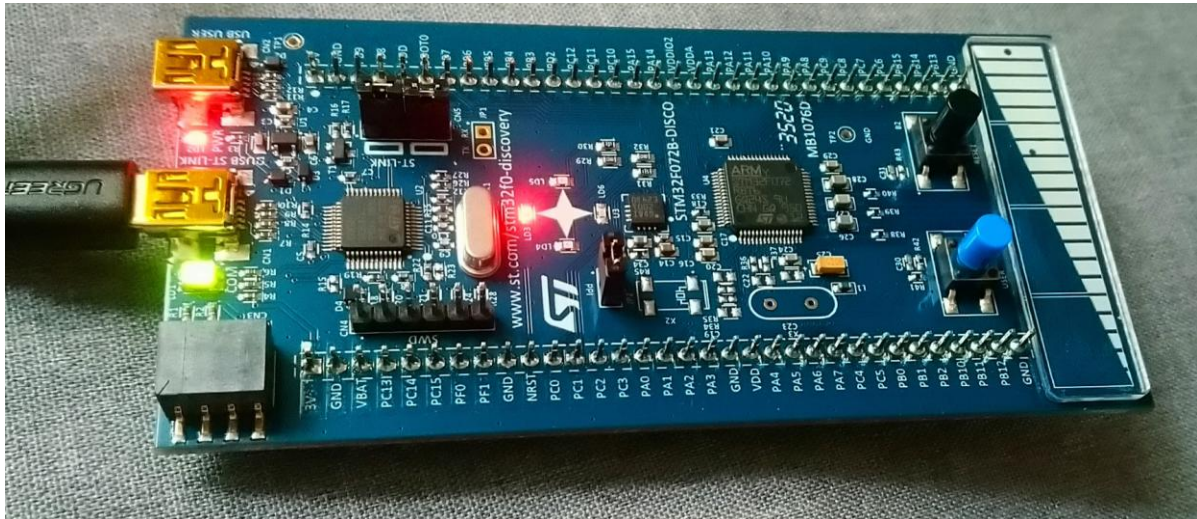
Rys. 3. Konfiguracja portów

3. Ustawienie zmiennych odpowiedzialnych za przechowywanie wyłączonej diody oraz aktualnie zapalanej diody. Zaświecenie aktywnej diody;

```
main.c x
main()
81
82     /* USER CODE BEGIN SysInit */
83
84     /* USER CODE END SysInit */
85
86     /* Initialize all configured peripherals */
87     MX_GPIO_Init();
88     /* USER CODE BEGIN 2 */
89     uint32_t launch_diode = GPIO_PIN_6; /* Przypisanie startowych wartosci dla obu diod */
90     uint32_t off_diode = GPIO_PIN_7;
91
92     HAL_GPIO_WritePin(GPIOC, launch_diode, GPIO_PIN_SET); /* zaswiecenie diody */
93     /* USER CODE END 2 */
94
95     /* Infinite loop */
96     /* USER CODE BEGIN WHILE */
97     while (1)
98     {
99         /* USER CODE END WHILE */
100
101         /* USER CODE BEGIN 3 */
102         if (HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0) == GPIO_PIN_SET)
103         {
104             if (launch_diode == GPIO_PIN_7) /* sprawdzenie ktorej diody jest zapalona */
```

Watch 1	
Expression	Value
launch_diode	64
<click to add>	

Rys. 4. Inicjalizacja zmiennych



Zdjęcie. 1. Zapalenie diody czerwonej

4. Sprawdzenie, czy przycisk jest wciśnięty;

```
main.c x
main()
100
101
102 if (HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0) == GPIO_PIN_SET) /* Sprawdzenie czy przycisk jest wciśnięty */
103 {
104     if (launch_diode == GPIO_PIN_7) /* sprawdzenie ktora dioda jest aktywna */
105     {
106         launch_diode = GPIO_PIN_6; /* podmiana zmiennych odpowiedzialnych za diode swiecaca */
107         off_diode = GPIO_PIN_7; /* i zgaszona */
108     } else {
109         launch_diode = GPIO_PIN_7; /* analogicznie */
110         off_diode = GPIO_PIN_6;
111     }
112     HAL_GPIO_WritePin(GPIOC, launch_diode, GPIO_PIN_SET); /* zaswiecenie diody */
113     HAL_GPIO_WritePin(GPIOC, off_diode, GPIO_PIN_RESET); /* zgaszenie diody */
114 }
115 HAL_GPIO_TogglePin(GPIOC, launch_diode); /* włącz/wyłącz diode aktywna */
116 HAL_Delay(500); /* ustaw opoznienie na 0,5 sekundy */
117 }
118 /* USER CODE END 3 */
```

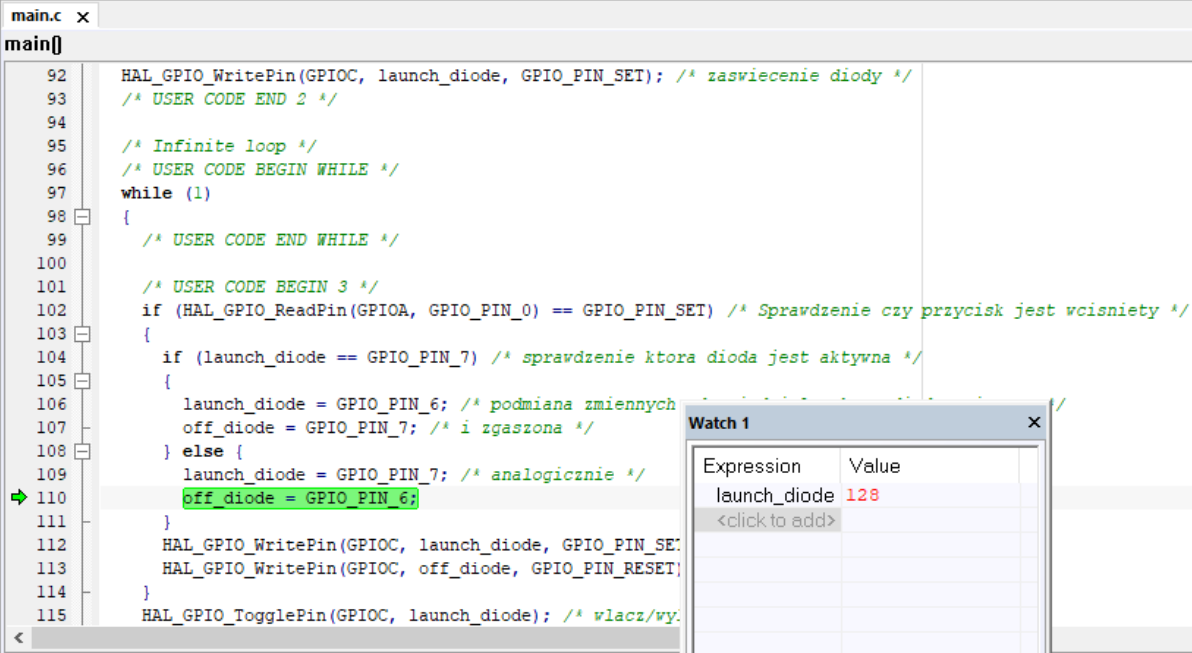
Rys. 5. Test wciśnięcia przycisku.

5. Jeśli tak, program sprawdza, która dioda jest aktywna;

```
97     while (1)
98     {
99         /* USER CODE END WHILE */
100
101         /* USER CODE BEGIN 3 */
102         if (HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0) == GPIO_PIN_SET) /* Sprawdzenie czy przycisk jest wcisnięty */
103         {
104             if (launch_diode == GPIO_PIN_7) /* sprawdzenie która dioda jest aktywna */
105             {
106                 launch_diode = GPIO_PIN_6; /* podmiana zmiennych odpowiedzialnych za diode swiecaca */
107                 off_diode = GPIO_PIN_7; /* i zgaszona */
108             } else {
109                 launch_diode = GPIO_PIN_7; /* analogicznie */
110                 off_diode = GPIO_PIN_6;
111             }
112             HAL_GPIO_WritePin(GPIOC, launch_diode, GPIO_PIN_SET); /* zaswiecenie diody */
113             HAL_GPIO_WritePin(GPIOC, off_diode, GPIO_PIN_RESET); /* zgaszenie diody */
114         }
115         HAL_GPIO_TogglePin(GPIOC, launch_diode); /* włącz/wyłącz diode aktywna */
116         HAL_Delay(500); /* ustaw opóźnienie na 0,5 sekundy */
117     }
```

Rys. 6. Oczekiwanie na znak od użytkownika

6. Program zamienia wartości pod zmiennymi, wskazującymi, która dioda ma się teraz zaświecić, a która zgasnąć;



The screenshot shows a code editor with a C program. The code is a while loop that checks a button (GPIO_PIN_0) and toggles LEDs. The Watch window shows the value of launch_diode as 128.

```
main.c x
main()
92     HAL_GPIO_WritePin(GPIOC, launch_diode, GPIO_PIN_SET); /* zaswiecenie diody */
93     /* USER CODE END 2 */
94
95     /* Infinite loop */
96     /* USER CODE BEGIN WHILE */
97     while (1)
98     {
99         /* USER CODE END WHILE */
100
101         /* USER CODE BEGIN 3 */
102         if (HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0) == GPIO_PIN_SET) /* Sprawdzenie czy przycisk jest wcisnięty */
103         {
104             if (launch_diode == GPIO_PIN_7) /* sprawdzenie która dioda jest aktywna */
105             {
106                 launch_diode = GPIO_PIN_6; /* podmiana zmiennych
107                 off_diode = GPIO_PIN_7; /* i zgaszona */
108             } else {
109                 launch_diode = GPIO_PIN_7; /* analogicznie */
110                 off_diode = GPIO_PIN_6;
111             }
112             HAL_GPIO_WritePin(GPIOC, launch_diode, GPIO_PIN_SE
113             HAL_GPIO_WritePin(GPIOC, off_diode, GPIO_PIN_RESET)
114         }
115         HAL_GPIO_TogglePin(GPIOC, launch_diode); /* włącz/wy.
```

Expression	Value
launch_diode	128
<click to add>	

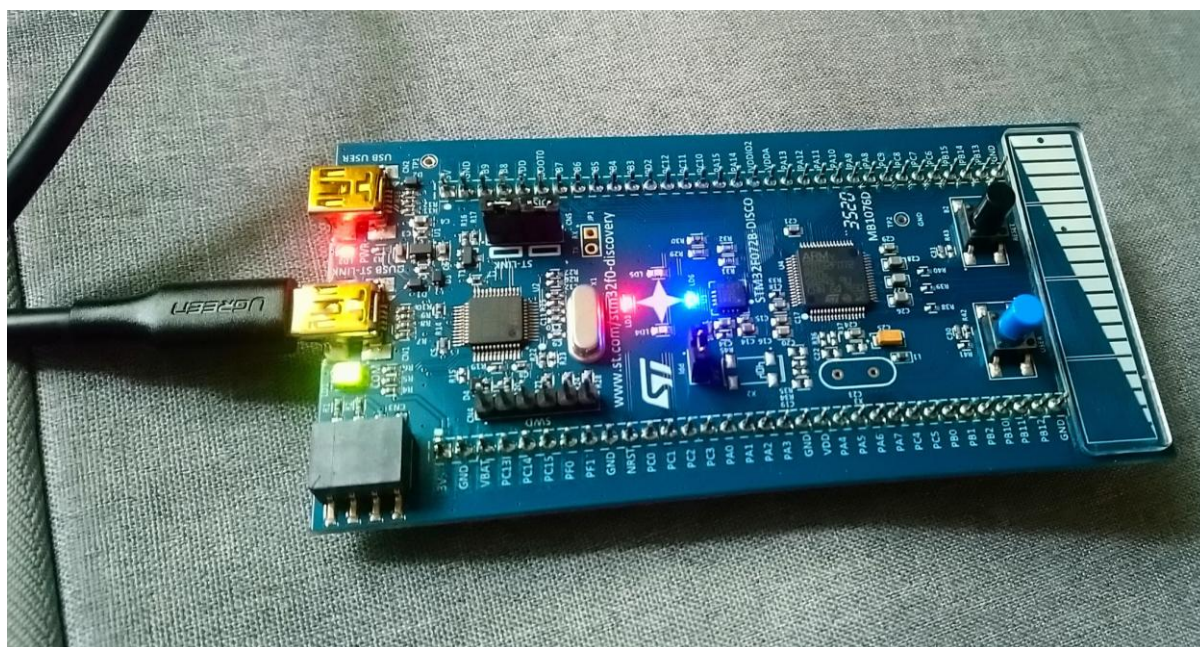
Rys. 7. Zamiana wartości dla zmiennych launch_diode i off_diode

7. Zamiana diod (świecącej i zgaszonej);

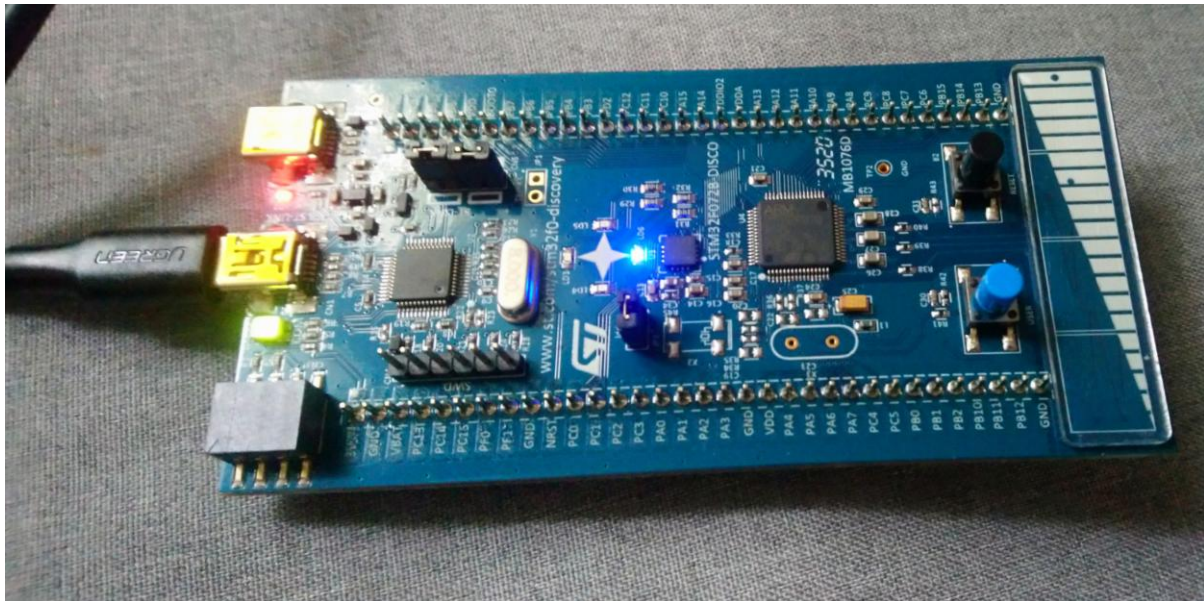
```
main.c x
main()

95  /* Infinite loop */
96  /* USER CODE BEGIN WHILE */
97  while (1)
98  {
99      /* USER CODE END WHILE */
100
101      /* USER CODE BEGIN 3 */
102      if (HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0) == GPIO_PIN_SET) /* Sprawdzenie czy przy
103      {
104          if (launch_diode == GPIO_PIN_7) /* sprawdzenie ktora dioda jest aktywna */
105          {
106              launch_diode = GPIO_PIN_6; /* podmiana zmiennych odpowiedzialnych za diode s
107              off_diode = GPIO_PIN_7; /* i zgaszona */
108          } else {
109              launch_diode = GPIO_PIN_7; /* analogicznie */
110              off_diode = GPIO_PIN_6;
111          }
112          HAL_GPIO_WritePin(GPIOC, launch_diode, GPIO_PIN_SET); /* zaswiecenie diody */
113          HAL_GPIO_WritePin(GPIOC, off_diode, GPIO_PIN_RESET); /* zgaszenie diody */
114      }
115      HAL_GPIO_TogglePin(GPIOC, launch_diode); /* włącz/wyłącz diode aktywna */
116      HAL_Delay(500); /* ustaw opoznienie na 0,5 sekundy */
117  }
118  /* USER CODE END 3 */
```

Rys. 8. Aktywacja nowej diody i zgaszenie poprzedniej



Zdjęcie 2. Zapalenie niebieskiej diody

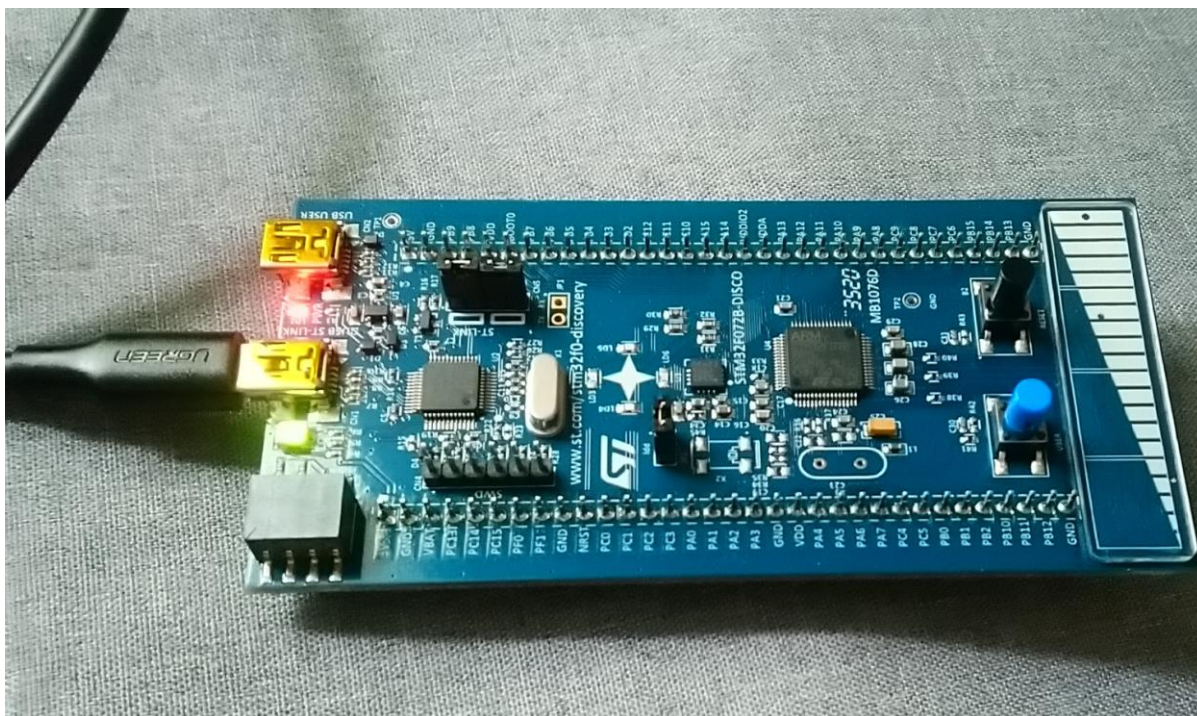


Zdjęcie 3. Zgaszenie czerwonej diody

8. Zgaś lub zapal aktywną diodę. Następnie odczekaj 0,5 sekundy;

```
main.c x
main()
95  /* Infinite loop */
96  /* USER CODE BEGIN WHILE */
97  while (1)
98  {
99      /* USER CODE END WHILE */
100
101      /* USER CODE BEGIN 3 */
102      if (HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0) == GPIO_PIN_SET) /* Sprawdzenie czy przycis
103      {
104          if (launch_diode == GPIO_PIN_7) /* sprawdzenie ktora dioda jest aktywna */
105          {
106              launch_diode = GPIO_PIN_6; /* podmiana zmiennych odpowiedzialnych za diode swie
107              off_diode = GPIO_PIN_7; /* i zgaszona */
108          } else {
109              launch_diode = GPIO_PIN_7; /* analogicznie */
110              off_diode = GPIO_PIN_6;
111          }
112          HAL_GPIO_WritePin(GPIOC, launch_diode, GPIO_PIN_SET); /* zaswiecenie diody */
113          HAL_GPIO_WritePin(GPIOC, off_diode, GPIO_PIN_RESET); /* zgaszenie diody */
114      }
115      HAL_GPIO_TogglePin(GPIOC, launch_diode); /* włącz/wyłącz diode aktywna */
116      HAL_Delay(500); /* ustaw opoznienie na 0,5 sekundy */
117  }
118  /* USER CODE END 3 */
```

Rys. 9. Odwróć stan diody



Zdjęcie 4. Zgaszenie diody aktywnej

9. W następnej iteracji, jeśli klawisz nie zostanie wciśnięty, program zapali diodę i odczeka 500 milisekund. Później cykl się powtórzy;

4. Podsumowanie i wnioski:

Obsługa Portów wejścia/wyjścia wymaga wygenerowania dużej ilości plików konfiguracyjnych. Na szczęście większość tej pracy wykonuje za nas program CubeMX, więc w trakcie przygotowania programu do płytki możemy skupić się na samym meritum programu, bez dogłębnego analizowania plików konfiguracyjnych.