Zadanie 1. Omówienie

A) Uruchomienie Timera, w konfiguracji wybrałem TIM2

```
/* USER CODE BEGIN 2 */
HAL_TIM_Base_Start_IT(&htim2);
/* USER CODE END 2 */
```

B) Wygenerowany kod po skonfigurowaniu Timera dla diody działającej z częstotliwością 5Hz. Celem jest uzyskanie 10Hz ponieważ przełączając diodę używamy zmiany jej stanu zatem jeżeli chcemy 5 zmian (włącz i wyłącz) na sekundę to musi być 10 sygnałów zmiany stanu wysłanych w jednej sekundzie.

Funkcja HAL_GPIO_TogglePin() odpowiada za zmiane stanu

```
/* USER CODE BEGIN TIM2_Init 1 */

/* USER CODE END TIM2_Init 1 */
htim2.Instance = TIM2;
htim2.Init.Prescaler = 16000;
htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
htim2.Init.Period = 4294967295;
htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
htim2.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
if (HAL_TIM_Base_Init(&htim2) != HAL_OK)
{
  Error_Handler();
}
sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
if (HAL_TIM_ConfigClockSource(&htim2, &sClockSourceConfig) != HAL_OK)
{
  Error_Handler();
}
sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
if (HAL_TIMEx_MasterConfigSynchronization(&htim2, &sMasterConfig) != HAL_OK)
{
  Error_Handler();
}
/* USER CODE BEGIN TIM2_Init 2 */

/* USER CODE END TIM2_Init 2 */



/**
 * @brief GPIO Initialization Function
```

C) Kod wygenerowany po ustawieniu Pinu PA5 jako wyjścia dla diody

```c
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
  /* USER CODE BEGIN MX_GPIO_Init_1 */

  /* USER CODE END MX_GPIO_Init_1 */

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOH_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(Diodia_GPIO_Port, Diodia_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin : Diodia_Pin */
  GPIO_InitStruct.Pin = Diodia_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(Diodia_GPIO_Port, &GPIO_InitStruct);

  /* USER CODE BEGIN MX_GPIO_Init_2 */
```

D) Kod wygenerowany po skonfigurowaniu watchdoga (dzielnik 32 dla czestotliwosci LSI 32kHz aby po 2 sekundach zawieszenia resetował

```c
/* USER CODE END IWDG_Init 1 */
hiwdg.Instance = IWDG;
hiwdg.Init.Prescaler = IWDG_PRESCALER_16;
hiwdg.Init.Reload = 4095;
if (HAL_IWDG_Init(&hiwdg) != HAL_OK)
{
  Error_Handler();
}
/* USER CODE BEGIN IWDG_Init 2 */
```

E) Ustawienie watchdoga w głównej pętli aby sprawdzał czy program działa

```c
while (1)
{
    HAL_IWDG_Refresh(&hiwdg);
  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */


}
```

F) Ustawienie zewnętrznego źródła taktowania o częstotliwości 16MHz w ustawieniach Reset and Clock Controler i końcowa konfiguracja

```
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  /** Initializes the RCC Oscillators according to the specified parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_LSI|RCC_OSCILLATORTYPE_HSE;
  RCC_OscInitStruct.HSEState = RCC_HSE_ON;
  RCC_OscInitStruct.LSIState = RCC_LSI_ON;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_NONE;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }

  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_HSE;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_0) != HAL_OK)
  {
    Error_Handler();
  }
```