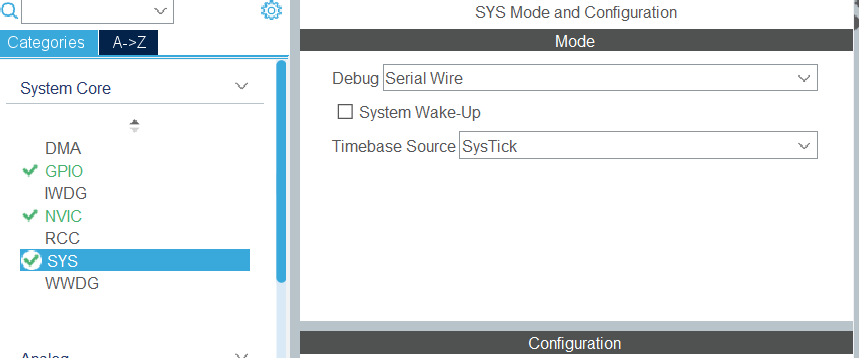
**STM32 CONFIGURACION GENERAL:**

**CONFIGURACION SYS LECTURA SERIAL**

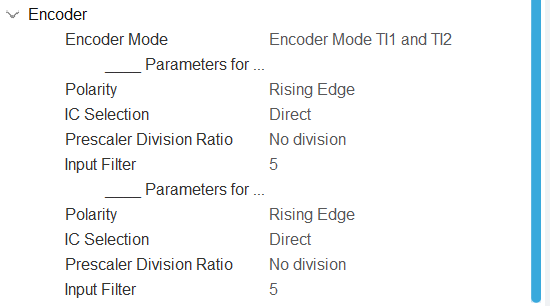
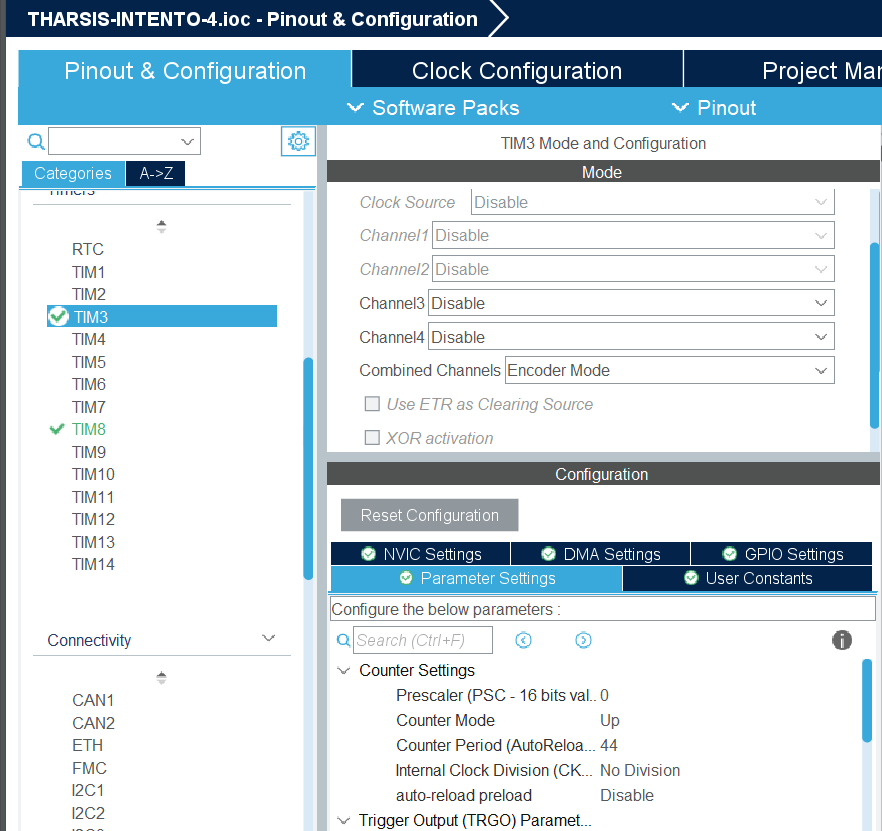


**CONFIGURACION CONNECTIVIDAD USART**

Interfaz de usuario gráfica

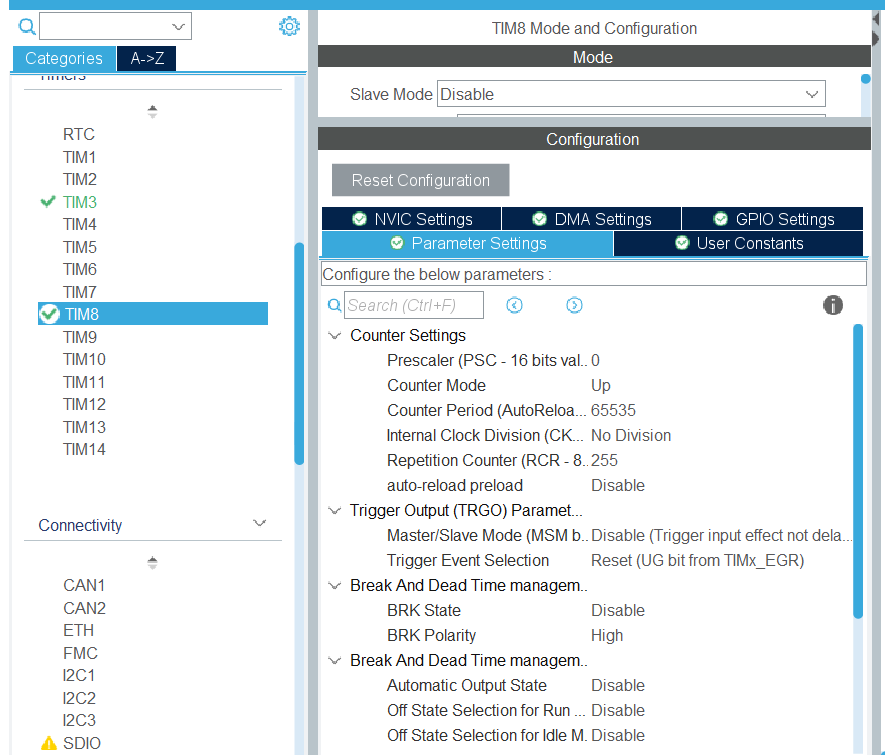
El contenido generado por IA puede ser incorrecto.

**CONFIGURACION TIM ENCODER**



INPUT\_FILTER: ES para encoder

**CONFIGURACION PINES PWM PARA MOTORES**



**CÓDIGO**

/\* USER CODE BEGIN Header \*/

/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @file : main.c

\* @brief : Main program body

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @attention

\*

\* Copyright (c) 2025 STMicroelectronics.

\* All rights reserved.

\*

\* This software is licensed under terms that can be found in the LICENSE file

\* in the root directory of this software component.

\* If no LICENSE file comes with this software, it is provided AS-IS.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

/\* USER CODE END Header \*/

/\* Includes ------------------------------------------------------------------\*/

**#include** "main.h"

/\* Private includes ----------------------------------------------------------\*/

/\* USER CODE BEGIN Includes \*/

**#include** <stdio.h> // Para usar sprintf

**#include** <string.h> // Para usar memset

/\* USER CODE END Includes \*/

/\* Private typedef -----------------------------------------------------------\*/

/\* USER CODE BEGIN PTD \*/

/\* USER CODE END PTD \*/

/\* Private define ------------------------------------------------------------\*/

/\* USER CODE BEGIN PD \*/

/\* USER CODE END PD \*/

/\* Private macro -------------------------------------------------------------\*/

/\* USER CODE BEGIN PM \*/

/\* USER CODE END PM \*/

/\* Private variables ---------------------------------------------------------\*/

**TIM\_HandleTypeDef** htim3;

**TIM\_HandleTypeDef** htim8;

**UART\_HandleTypeDef** huart1;

/\* USER CODE BEGIN PV \*/

**int32\_t** full\_rotations = 0; // Variable global para contar las vueltas completas

**uint32\_t** previous\_encoder\_value = 0; // Valor anterior del encoder

**uint32\_t** previous\_time = 0; // Tiempo anterior en milisegundos

**float** speed\_mps = 0; // Velocidad en m/s

**float** wheel\_radius = 0.09; // Radio de la rueda en metros

**char** buffer[50]; // Buffer para almacenar la cadena de texto

/\* USER CODE END PV \*/

/\* Private function prototypes -----------------------------------------------\*/

**void** **SystemClock\_Config**(**void**);

**static** **void** **MX\_GPIO\_Init**(**void**);

**static** **void** **MX\_TIM8\_Init**(**void**);

**static** **void** **MX\_TIM3\_Init**(**void**);

**static** **void** **MX\_USART1\_UART\_Init**(**void**);

/\* USER CODE BEGIN PFP \*/

**int** **\_write**(**int** file, **char** \*ptr, **int** len) {

**HAL\_UART\_Transmit**(&huart1, (**uint8\_t**\*)ptr, len, HAL\_MAX\_DELAY);

**return** len;

}

/\* USER CODE END PFP \*/

/\* Private user code ---------------------------------------------------------\*/

/\* USER CODE BEGIN 0 \*/

/\* USER CODE END 0 \*/

/\*\*

\* @brief The application entry point.

\* @retval int

\*/

**int** **main**(**void**)

{

/\* USER CODE BEGIN 1 \*/

/\* USER CODE END 1 \*/

/\* MCU Configuration--------------------------------------------------------\*/

/\* Reset of all peripherals, Initializes the Flash interface and the Systick. \*/

**HAL\_Init**();

/\* USER CODE BEGIN Init \*/

/\* USER CODE END Init \*/

/\* Configure the system clock \*/

**SystemClock\_Config**();

/\* USER CODE BEGIN SysInit \*/

/\* USER CODE END SysInit \*/

/\* Initialize all configured peripherals \*/

**MX\_GPIO\_Init**();

**MX\_TIM8\_Init**();

**MX\_TIM3\_Init**();

**MX\_USART1\_UART\_Init**();

/\* USER CODE BEGIN 2 \*/

**HAL\_TIM\_Encoder\_Start**(&htim3, TIM\_CHANNEL\_ALL); // Inicia el encoder

**HAL\_TIM\_PWM\_Start**(&htim8, TIM\_CHANNEL\_1);

TIM8->CCR1 =0;

/\* USER CODE END 2 \*/

/\* Infinite loop \*/

/\* USER CODE BEGIN WHILE \*/

**HAL\_TIM\_PWM\_Start**(&htim8,TIM\_CHANNEL\_1);

**while** (1)

{

**HAL\_GPIO\_WritePin**(GPIOD, GPIO\_PIN\_10, *GPIO\_PIN\_SET*);

// Variable para almacenar el valor de PWM ingresado

**uint16\_t** pwm\_value = 32767;

// Solicitar al usuario que ingrese el valor de PWM (0 a 65535)

// Aquí puedes usar una interfaz como UART, un teclado, o cualquier otro método de entrada.

// Por simplicidad, asumimos que el valor se ingresa manualmente en el código.

// Ejemplo: Cambia este valor manualmente para ajustar la velocidad del motor.

// pwm\_value = 20000; // Por ejemplo, 50% del ciclo de trabajo

// Aplicar el valor de PWM al registro CCR1

TIM8->CCR1 = pwm\_value;

// Mantener la velocidad del motor con el valor ingresado

**HAL\_Delay**(1000); // Pequeño retardo para evitar cambios bruscos

// Lee el valor actual del contador del encoder

**uint32\_t** encoder\_value = TIM3->CNT;

// Calcula el cambio en la posición del encoder, teniendo en cuenta el desbordamiento

**int32\_t** delta\_pulses = (**int32\_t**)(encoder\_value - previous\_encoder\_value);

// Actualiza el valor anterior del encoder

previous\_encoder\_value = encoder\_value;

// Ajusta delta\_pulses si hay un desbordamiento

**if** (delta\_pulses > 22) { // Si el cambio es mayor que la mitad del ARR (44/2 = 22)

delta\_pulses -= 44; // Ajusta para manejar el desbordamiento negativo

} **else** **if** (delta\_pulses < -22) { // Si el cambio es menor que la mitad negativa del ARR

delta\_pulses += 44; // Ajusta para manejar el desbordamiento positivo

}

// Obtén el tiempo actual en milisegundos

**uint32\_t** current\_time = **HAL\_GetTick**();

// Calcula el intervalo de tiempo (en segundos)

**float** delta\_time = (current\_time - previous\_time) / 1000.0;

// Actualiza el tiempo anterior

previous\_time = current\_time;

// Calcula la velocidad en m/s

**float** delta\_theta = delta\_pulses \* (2 \* 3.14159265358979323846 / 44);

speed\_mps = (delta\_theta \* wheel\_radius) / delta\_time;

// Imprime la velocidad en m/s

**printf**("Velocidad: %.2f m/s\r\n", speed\_mps);

// Pequeño retardo para evitar saturar el terminal

**HAL\_Delay**(100);

/\* USER CODE END WHILE \*/

/\* USER CODE BEGIN 3 \*/

}

/\* USER CODE END 3 \*/

}

/\*\*

\* @brief System Clock Configuration

\* @retval None

\*/

**void** **SystemClock\_Config**(**void**)

{

**RCC\_OscInitTypeDef** RCC\_OscInitStruct = {0};

**RCC\_ClkInitTypeDef** RCC\_ClkInitStruct = {0};

/\*\* Configure the main internal regulator output voltage

\*/

\_\_HAL\_RCC\_PWR\_CLK\_ENABLE();

\_\_HAL\_PWR\_VOLTAGESCALING\_CONFIG(PWR\_REGULATOR\_VOLTAGE\_SCALE1);

/\*\* Initializes the RCC Oscillators according to the specified parameters

\* in the RCC\_OscInitTypeDef structure.

\*/

RCC\_OscInitStruct.OscillatorType = RCC\_OSCILLATORTYPE\_HSI;

RCC\_OscInitStruct.HSIState = RCC\_HSI\_ON;

RCC\_OscInitStruct.HSICalibrationValue = RCC\_HSICALIBRATION\_DEFAULT;

RCC\_OscInitStruct.PLL.PLLState = RCC\_PLL\_ON;

RCC\_OscInitStruct.PLL.PLLSource = RCC\_PLLSOURCE\_HSI;

RCC\_OscInitStruct.PLL.PLLM = 8;

RCC\_OscInitStruct.PLL.PLLN = 180;

RCC\_OscInitStruct.PLL.PLLP = RCC\_PLLP\_DIV2;

RCC\_OscInitStruct.PLL.PLLQ = 4;

**if** (**HAL\_RCC\_OscConfig**(&RCC\_OscInitStruct) != *HAL\_OK*)

{

**Error\_Handler**();

}

/\*\* Activate the Over-Drive mode

\*/

**if** (**HAL\_PWREx\_EnableOverDrive**() != *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\_PLLCLK;

RCC\_ClkInitStruct.AHBCLKDivider = RCC\_SYSCLK\_DIV1;

RCC\_ClkInitStruct.APB1CLKDivider = RCC\_HCLK\_DIV4;

RCC\_ClkInitStruct.APB2CLKDivider = RCC\_HCLK\_DIV2;

**if** (**HAL\_RCC\_ClockConfig**(&RCC\_ClkInitStruct, FLASH\_LATENCY\_5) != *HAL\_OK*)

{

**Error\_Handler**();

}

}

/\*\*

\* @brief TIM3 Initialization Function

\* @param None

\* @retval None

\*/

**static** **void** **MX\_TIM3\_Init**(**void**)

{

/\* USER CODE BEGIN TIM3\_Init 0 \*/

/\* USER CODE END TIM3\_Init 0 \*/

**TIM\_Encoder\_InitTypeDef** sConfig = {0};

**TIM\_MasterConfigTypeDef** sMasterConfig = {0};

/\* USER CODE BEGIN TIM3\_Init 1 \*/

/\* USER CODE END TIM3\_Init 1 \*/

htim3.Instance = TIM3;

htim3.Init.Prescaler = 0;

htim3.Init.CounterMode = TIM\_COUNTERMODE\_UP;

htim3.Init.Period = 44;

htim3.Init.ClockDivision = TIM\_CLOCKDIVISION\_DIV1;

htim3.Init.AutoReloadPreload = TIM\_AUTORELOAD\_PRELOAD\_DISABLE;

sConfig.EncoderMode = TIM\_ENCODERMODE\_TI12;

sConfig.IC1Polarity = TIM\_ICPOLARITY\_RISING;

sConfig.IC1Selection = TIM\_ICSELECTION\_DIRECTTI;

sConfig.IC1Prescaler = TIM\_ICPSC\_DIV1;

sConfig.IC1Filter = 5;

sConfig.IC2Polarity = TIM\_ICPOLARITY\_RISING;

sConfig.IC2Selection = TIM\_ICSELECTION\_DIRECTTI;

sConfig.IC2Prescaler = TIM\_ICPSC\_DIV1;

sConfig.IC2Filter = 5;

**if** (**HAL\_TIM\_Encoder\_Init**(&htim3, &sConfig) != *HAL\_OK*)

{

**Error\_Handler**();

}

sMasterConfig.MasterOutputTrigger = TIM\_TRGO\_RESET;

sMasterConfig.MasterSlaveMode = TIM\_MASTERSLAVEMODE\_DISABLE;

**if** (**HAL\_TIMEx\_MasterConfigSynchronization**(&htim3, &sMasterConfig) != *HAL\_OK*)

{

**Error\_Handler**();

}

/\* USER CODE BEGIN TIM3\_Init 2 \*/

/\* USER CODE END TIM3\_Init 2 \*/

}

/\*\*

\* @brief TIM8 Initialization Function

\* @param None

\* @retval None

\*/

**static** **void** **MX\_TIM8\_Init**(**void**)

{

/\* USER CODE BEGIN TIM8\_Init 0 \*/

/\* USER CODE END TIM8\_Init 0 \*/

**TIM\_MasterConfigTypeDef** sMasterConfig = {0};

**TIM\_OC\_InitTypeDef** sConfigOC = {0};

**TIM\_BreakDeadTimeConfigTypeDef** sBreakDeadTimeConfig = {0};

/\* USER CODE BEGIN TIM8\_Init 1 \*/

/\* USER CODE END TIM8\_Init 1 \*/

htim8.Instance = TIM8;

htim8.Init.Prescaler = 0;

htim8.Init.CounterMode = TIM\_COUNTERMODE\_UP;

htim8.Init.Period = 65535;

htim8.Init.ClockDivision = TIM\_CLOCKDIVISION\_DIV1;

htim8.Init.RepetitionCounter = 255;

htim8.Init.AutoReloadPreload = TIM\_AUTORELOAD\_PRELOAD\_DISABLE;

**if** (**HAL\_TIM\_PWM\_Init**(&htim8) != *HAL\_OK*)

{

**Error\_Handler**();

}

sMasterConfig.MasterOutputTrigger = TIM\_TRGO\_RESET;

sMasterConfig.MasterSlaveMode = TIM\_MASTERSLAVEMODE\_DISABLE;

**if** (**HAL\_TIMEx\_MasterConfigSynchronization**(&htim8, &sMasterConfig) != *HAL\_OK*)

{

**Error\_Handler**();

}

sConfigOC.OCMode = TIM\_OCMODE\_PWM1;

sConfigOC.Pulse = 0;

sConfigOC.OCPolarity = TIM\_OCPOLARITY\_HIGH;

sConfigOC.OCNPolarity = TIM\_OCNPOLARITY\_HIGH;

sConfigOC.OCFastMode = TIM\_OCFAST\_DISABLE;

sConfigOC.OCIdleState = TIM\_OCIDLESTATE\_RESET;

sConfigOC.OCNIdleState = TIM\_OCNIDLESTATE\_RESET;

**if** (**HAL\_TIM\_PWM\_ConfigChannel**(&htim8, &sConfigOC, TIM\_CHANNEL\_1) != *HAL\_OK*)

{

**Error\_Handler**();

}

sBreakDeadTimeConfig.OffStateRunMode = TIM\_OSSR\_DISABLE;

sBreakDeadTimeConfig.OffStateIDLEMode = TIM\_OSSI\_DISABLE;

sBreakDeadTimeConfig.LockLevel = TIM\_LOCKLEVEL\_OFF;

sBreakDeadTimeConfig.DeadTime = 0;

sBreakDeadTimeConfig.BreakState = TIM\_BREAK\_DISABLE;

sBreakDeadTimeConfig.BreakPolarity = TIM\_BREAKPOLARITY\_HIGH;

sBreakDeadTimeConfig.AutomaticOutput = TIM\_AUTOMATICOUTPUT\_DISABLE;

**if** (**HAL\_TIMEx\_ConfigBreakDeadTime**(&htim8, &sBreakDeadTimeConfig) != *HAL\_OK*)

{

**Error\_Handler**();

}

/\* USER CODE BEGIN TIM8\_Init 2 \*/

/\* USER CODE END TIM8\_Init 2 \*/

**HAL\_TIM\_MspPostInit**(&htim8);

}

/\*\*

\* @brief USART1 Initialization Function

\* @param None

\* @retval None

\*/

**static** **void** **MX\_USART1\_UART\_Init**(**void**)

{

/\* USER CODE BEGIN USART1\_Init 0 \*/

/\* USER CODE END USART1\_Init 0 \*/

/\* USER CODE BEGIN USART1\_Init 1 \*/

/\* USER CODE END USART1\_Init 1 \*/

huart1.Instance = USART1;

huart1.Init.BaudRate = 115200;

huart1.Init.WordLength = UART\_WORDLENGTH\_8B;

huart1.Init.StopBits = UART\_STOPBITS\_1;

huart1.Init.Parity = UART\_PARITY\_NONE;

huart1.Init.Mode = UART\_MODE\_TX\_RX;

huart1.Init.HwFlowCtl = UART\_HWCONTROL\_NONE;

huart1.Init.OverSampling = UART\_OVERSAMPLING\_16;

**if** (**HAL\_UART\_Init**(&huart1) != *HAL\_OK*)

{

**Error\_Handler**();

}

/\* USER CODE BEGIN USART1\_Init 2 \*/

/\* USER CODE END USART1\_Init 2 \*/

}

/\*\*

\* @brief GPIO Initialization Function

\* @param None

\* @retval None

\*/

**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\_GPIOA\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOD\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOC\_CLK\_ENABLE();

/\*Configure GPIO pin Output Level \*/

**HAL\_GPIO\_WritePin**(GPIOD, GPIO\_PIN\_10, *GPIO\_PIN\_RESET*);

/\*Configure GPIO pin : PD10 \*/

GPIO\_InitStruct.Pin = GPIO\_PIN\_10;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

**HAL\_GPIO\_Init**(GPIOD, &GPIO\_InitStruct);

/\* USER CODE BEGIN MX\_GPIO\_Init\_2 \*/

/\* USER CODE END MX\_GPIO\_Init\_2 \*/

}

/\* USER CODE BEGIN 4 \*/

/\* USER CODE END 4 \*/

/\*\*

\* @brief This function is executed in case of error occurrence.

\* @retval None

\*/

**void** **Error\_Handler**(**void**)

{

/\* USER CODE BEGIN Error\_Handler\_Debug \*/

/\* User can add his own implementation to report the HAL error return state \*/

**\_\_disable\_irq**();

**while** (1)

{

}

/\* USER CODE END Error\_Handler\_Debug \*/

}

**#ifdef** USE\_FULL\_ASSERT

/\*\*

\* @brief Reports the name of the source file and the source line number

\* where the assert\_param error has occurred.

\* @param file: pointer to the source file name

\* @param line: assert\_param error line source number

\* @retval None

\*/

**void** assert\_failed(uint8\_t \*file, uint32\_t line)

{

/\* USER CODE BEGIN 6 \*/

/\* User can add his own implementation to report the file name and line number,

ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) \*/

/\* USER CODE END 6 \*/

}

**#endif** /\* USE\_FULL\_ASSERT \*/