Projekt zdalnie sterowanego samochodzika

Generated by Doxygen 1.11.0

Chapter 1

Topic Index

1.1 Topics

Here is a list of all topics with brief descriptions:

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

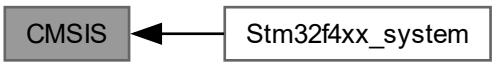
Here is a list of all documented files with brief descriptions:

Chapter 4

Topic Documentation

4.1 CMSIS

Collaboration diagram for CMSIS:



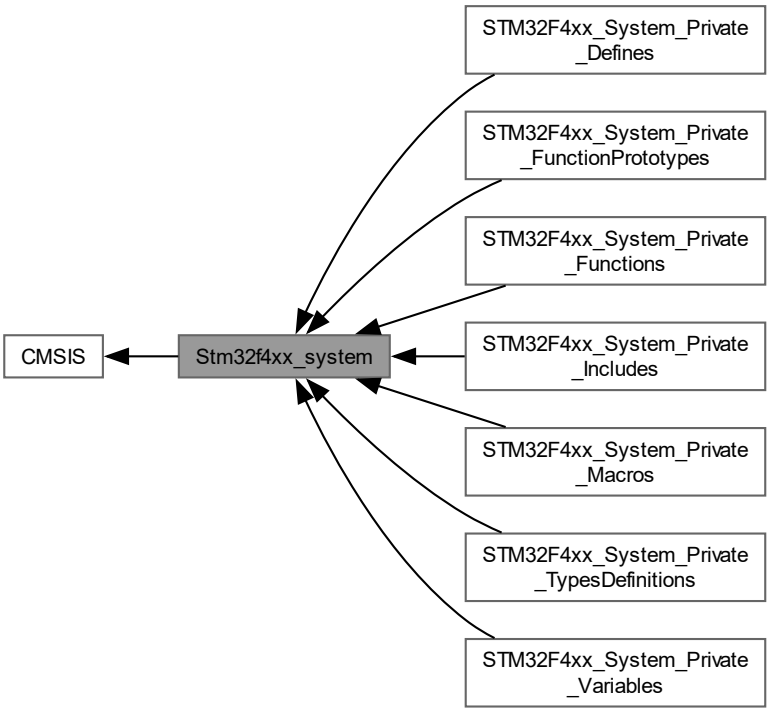
Topics

- Stm32f4xx_system

4.1.1 Detailed Description

4.1.2 Stm32f4xx_system

Collaboration diagram for Stm32f4xx_system:



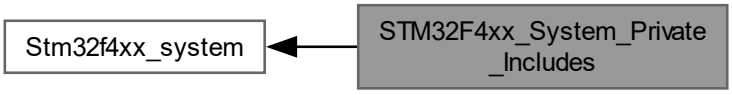
Topics

- STM32F4xx_System_Private_Includes
- STM32F4xx_System_Private_TypesDefinitions
- STM32F4xx_System_Private_Defines
- STM32F4xx_System_Private_Macros
- STM32F4xx_System_Private_Variables
- STM32F4xx_System_Private_FunctionPrototypes
- STM32F4xx_System_Private_Functions

4.1.2.1 Detailed Description

4.1.2.2 STM32F4xx_System_Private_Includes

Collaboration diagram for STM32F4xx_System_Private_Includes:



Macros

- #define HSE_VALUE ((uint32_t)25000000)
- #define HSI_VALUE ((uint32_t)16000000)

4.1.2.2.1 Detailed Description

4.1.2.2.2 Macro Definition Documentation

4.1.2.2.2.1 HSE_VALUE

```
#define HSE_VALUE ((uint32_t)25000000)
```

Default value of the External oscillator in Hz

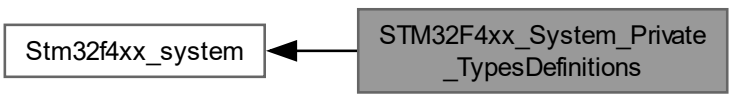
4.1.2.2.2.2 HSI_VALUE

```
#define HSI_VALUE ((uint32_t)16000000)
```

Value of the Internal oscillator in Hz

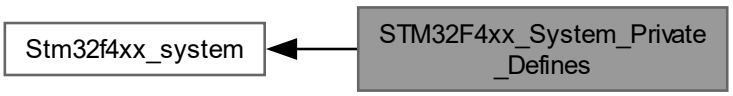
4.1.2.3 STM32F4xx_System_Private_TypesDefinitions

Collaboration diagram for STM32F4xx_System_Private_TypesDefinitions:



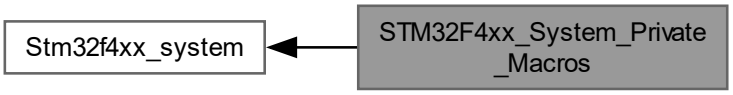
4.1.2.4 STM32F4xx_System_Private_Defines

Collaboration diagram for STM32F4xx_System_Private_Defines:



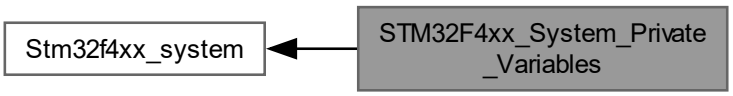
4.1.2.5 STM32F4xx_System_Private_Macros

Collaboration diagram for STM32F4xx_System_Private_Macros:



4.1.2.6 STM32F4xx_System_Private_Variables

Collaboration diagram for STM32F4xx_System_Private_Variables:



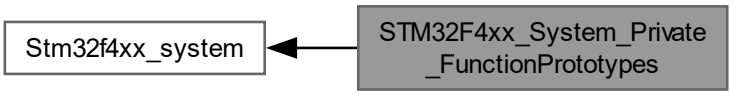
Variables

- uint32_t **SystemCoreClock** = 16000000
- const uint8_t **AHBPrescTable** [16] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6, 7, 8, 9}
- const uint8_t **APBPrescTable** [8] = {0, 0, 0, 0, 1, 2, 3, 4}

4.1.2.6.1 Detailed Description

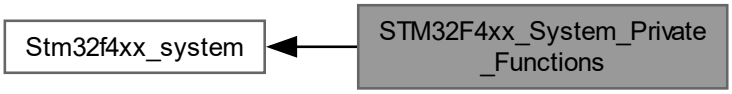
4.1.2.7 STM32F4xx_System_Private_FunctionPrototypes

Collaboration diagram for STM32F4xx_System_Private_FunctionPrototypes:



4.1.2.8 STM32F4xx_System_Private_Functions

Collaboration diagram for STM32F4xx_System_Private_Functions:



Functions

- void SystemInit (void)

  Setup the microcontroller system Initialize the FPU setting, vector table location and External memory configuration.
- void SystemCoreClockUpdate (void)

  Update SystemCoreClock variable according to Clock Register Values. The SystemCoreClock variable contains the core clock (HCLK), it can be used by the user application to setup the SysTick timer or configure other parameters.

4.1.2.8.1 Detailed Description

4.1.2.8.2 Function Documentation

4.1.2.8.2.1 SystemCoreClockUpdate()

```
void SystemCoreClockUpdate (
            void )
```

Update SystemCoreClock variable according to Clock Register Values. The SystemCoreClock variable contains the core clock (HCLK), it can be used by the user application to setup the SysTick timer or configure other parameters.

Note

Each time the core clock (HCLK) changes, this function must be called to update SystemCoreClock variable value. Otherwise, any configuration based on this variable will be incorrect.

- The system frequency computed by this function is not the real frequency in the chip. It is calculated based on the predefined constant and the selected clock source:

- If SYSCLK source is HSI, SystemCoreClock will contain the HSI_VALUE(∗)
- If SYSCLK source is HSE, SystemCoreClock will contain the HSE_VALUE(∗∗)
- If SYSCLK source is PLL, SystemCoreClock will contain the HSE_VALUE(∗∗) or HSI_VALUE(∗) multiplied/divided by the PLL factors.

(∗) HSI_VALUE is a constant defined in stm32f4xx_hal_conf.h file (default value 16 MHz) but the real value may vary depending on the variations in voltage and temperature.

(∗∗) HSE_VALUE is a constant defined in stm32f4xx_hal_conf.h file (its value depends on the application requirements), user has to ensure that HSE_VALUE is same as the real frequency of the crystal used. Otherwise, this function may have wrong result.

- The result of this function could be not correct when using fractional value for HSE crystal.

Parameters

None	

Return values

None	

4.1.2.8.2.2 SystemInit()

```
void SystemInit (
             void )
```

Setup the microcontroller system Initialize the FPU setting, vector table location and External memory configuration.

Parameters

None	

Return values

None	

Chapter 5

Data Structure Documentation

5.1 carStruct Struct Reference

Structure representing the state of the car.

`#include <car_control.h>`

Data Fields

- float obstacle_distance
- uint8_t ride
- uint8_t turn
- float turn_angle

5.1.1 Detailed Description

Structure representing the state of the car.

5.1.2 Field Documentation

5.1.2.1 obstacle_distance

`float carStruct::obstacle_distance`

Distance to the nearest obstacle

5.1.2.2 ride

`uint8_t carStruct::ride`

Direction of movement: forward 'f' or backward 'b'

5.1.2.3 turn

```
uint8_t carStruct::turn
```

Direction of turning: left 'l' or right 'r'

5.1.2.4 turn_angle

```
float carStruct::turn_angle
```

Angle of turning

The documentation for this struct was generated from the following file:

- car_control.h

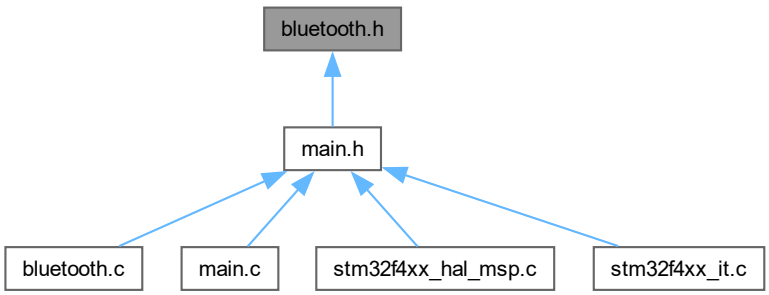
Chapter 6

File Documentation

6.1 bluetooth.h File Reference

Header file for Bluetooth communication and control functions.

This graph shows which files directly or indirectly include this file:



Functions

- void HAL_UART_RxCpltCallback (UART_HandleTypeDef *huart)

  Callback function for UART receive complete interrupt.
- void read_logic (void)

  Function to process the received line of data.
- void line_append (uint8_t value)

  Function to append a received byte to the line buffer.
- void bluetooth_send (void)

  Function to send a Bluetooth message.

6.1.1 Detailed Description

Header file for Bluetooth communication and control functions.

Author

Jakub Wysocki

Version

0.1

Date

2024-05-24

Copyright

6.1.2 Function Documentation

6.1.2.1 bluetooth_send()

```
void bluetooth_send (
            void )
```

Function to send a Bluetooth message.

This function formats and sends a message containing the car's obstacle distance over Bluetooth using the UART transmit interrupt.

6.1.2.2 HAL_UART_RxCpltCallback()

```
void HAL_UART_RxCpltCallback (
            UART_HandleTypeDef * huart)
```

Callback function for UART receive complete interrupt.

Parameters

huart	

Callback function for UART receive complete interrupt.

It checks if the received data is from USART2 and processes the received byte.

Parameters

huart	

6.1.2.3 line_append()

```
void line_append (
             uint8_t value)
```

Function to append a received byte to the line buffer.

Parameters

value	

This function adds a received byte to the line buffer. If the byte is a newline or carriage return, it processes the complete line.

Parameters

value	

6.1.2.4 read_logic()

```
void read_logic (
             void )
```

Function to process the received line of data.

This function processes the command received from the Bluetooth module. Depending on the command, it adjusts the car's:

- turn angle (left, right, idle),
- toggles lights (on, off),
- ride mode (foreward, backward, idle).

6.2 bluetooth.h

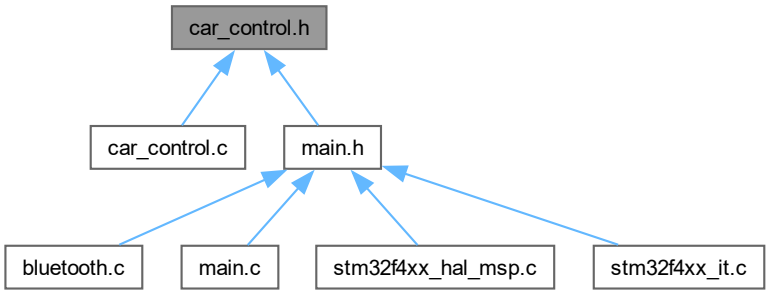
Go to the documentation of this file.

```
00001
00012 #ifndef INC_BLUETOOTH_H_
00013 #define INC_BLUETOOTH_H_
00014
00020 void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart);
00021
00025 void read_logic(void);
00026
00032 void line_append(uint8_t value);
00033
00037 void bluetooth_send(void);
00038
00039 #endif /* INC_BLUETOOTH_H_ */
```

6.3 car_control.h File Reference

Header file containing function prototypes and structure definition for car control.

This graph shows which files directly or indirectly include this file:



Data Structures

- struct carStruct

  Structure representing the state of the car.

Typedefs

- typedef struct carStruct **carStruct**

  Structure representing the state of the car.

Functions

- void **go_idle** (void)

  Set the car to idle state.
- void **go_foreward** (void)

  Set the car to move forward.
- void **go_backward** (void)

  Set the car to move backward.
- void **turn_idle** (void)

  Set the car to idle state for turning.
- void **turn_left** (void)

  Turn the car to the left.
- void **turn_right** (void)

  Turn the car to the right.
- void **car_init** (void)

  Initialize the car by setting it to idle state.
- void car_control (void)

  Control the car based on its current state.

Variables

- carStruct car

6.3.1 Detailed Description

Header file containing function prototypes and structure definition for car control.

Author

Jacek Bielski

Version

0.1

Date

2024-05-24

Copyright

6.3.2 Function Documentation

6.3.2.1 car_control()

```
void car_control (
            void )
```

Control the car based on its current state.

This function contains logic for steering the car, based on the current state of the car struct.

This function contains logic for steering the car, based on the current state of the car struct. It checks the `car.ride` state to determine the movement of the car (forward, backward, or idle). It also checks the `car.turn` state to determine the direction of the car (left, right, or idle).

6.3.3 Variable Documentation

6.3.3.1 car

```
carStruct car  [extern]
```

External reference to the car state

6.4 car_control.h

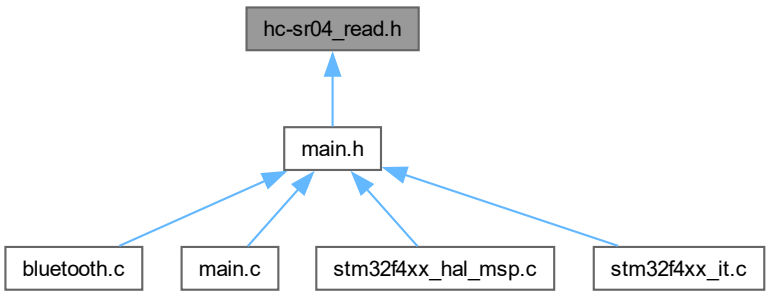
Go to the documentation of this file.

```
00001
00012 #ifndef __CAR_CONTROL_H
00013 #define __CAR_CONTROL_H
00014
00015 #ifdef __cplusplus
00016 extern "C"
00017 {
00018 #endif
00019
00023     typedef struct carStruct
00024     {
00025         float obstacle_distance;
00026         uint8_t ride;
00027         uint8_t turn;
00028         float turn_angle;
00029     } carStruct;
00030
00031     extern carStruct car;
00036     void go_idle(void);
00037
00041     void go_foreward(void);
00042
00046     void go_backward(void);
00047
00051     void turn_idle(void);
00052
00056     void turn_left(void);
00057
00061     void turn_right(void);
00062
00066     void car_init(void);
00067
00073     void car_control(void);
00074
00075 #ifdef __cplusplus
00076 }
00077 #endif
00078
00079 #endif /* __CAR_CONTROL_H */
```

6.5 hc-sr04_read.h File Reference

Header file containing function prototypes for HC-SR04 ultrasonic sensor reading.

This graph shows which files directly or indirectly include this file:



Functions

- void hcsr04_read_init (void)

  Initialize HC-SR04 sensor for distance measurement.
- float obstacle_detection (void)

  Perform obstacle detection using HC-SR04 sensor.

6.5.1 Detailed Description

Header file containing function prototypes for HC-SR04 ultrasonic sensor reading.

Author

https://forbot.pl/blog/kurs-stm32l4-czujnik-odleglosci-wyswietlacz-7-seg-id48628

Version

0.1

Date

2024-05-24

Copyright

6.5.2 Function Documentation

6.5.2.1 hcsr04_read_init()

```
void hcsr04_read_init (
            void )
```

Initialize HC-SR04 sensor for distance measurement.

Initialize HC-SR04 sensor for distance measurement.

The function activates three TIM2 channels: two for capturing the Echo signal (start and stop) and one to generate a trigger pulse for the HC-SR04 module.

6.5.2.2 obstacle_detection()

```
float obstacle_detection (
            void )
```

Perform obstacle detection using HC-SR04 sensor.

Returns

The distance to the nearest obstacle in centimeters.

Perform obstacle detection using HC-SR04 sensor.

The function calculates the distance based on the time difference between the moment the Echo signal appears and its disappearance. The result is converted to centimeters.

Returns

float Calculated distance in centimeters.

6.6 hc-sr04_read.h

Go to the documentation of this file.

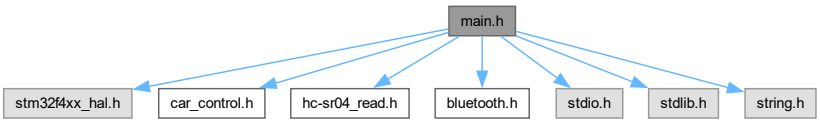
```
00001
00012 #ifndef INC_HC_SR04_READ_H_
00013 #define INC_HC_SR04_READ_H_
00014
00018 void hcsr04_read_init(void);
00019
00025 float obstacle_detection(void);
00026
00027 #endif /* INC_HC_SR04_READ_H_ */
```

6.7 main.h File Reference

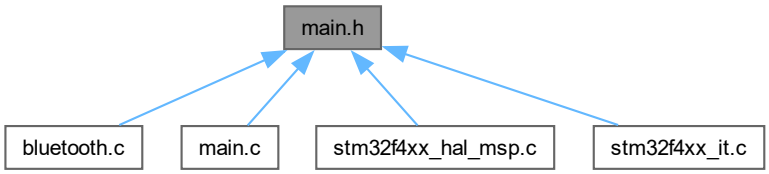
: Header for main.c file. This file contains the common defines of the application.

```
#include "stm32f4xx_hal.h"
#include "car_control.h"
#include "hc-sr04_read.h"
#include "bluetooth.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

Include dependency graph for main.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define **blue_led_Pin** GPIO_PIN_13
- #define **blue_led_GPIO_Port** GPIOC
- #define **foreward_Pin** GPIO_PIN_5
- #define **foreward_GPIO_Port** GPIOA

- #define **backward_Pin** GPIO_PIN_6
- #define **backward_GPIO_Port** GPIOA
- #define **left_Pin** GPIO_PIN_9
- #define **left_GPIO_Port** GPIOA
- #define **right_Pin** GPIO_PIN_10
- #define **right_GPIO_Port** GPIOA

Functions

- void HAL_TIM_MspPostInit (TIM_HandleTypeDef ∗htim)
- void Error_Handler (void)

  This function is executed in case of error occurrence.

6.7.1 Detailed Description

: Header for main.c file. This file contains the common defines of the application.

Attention

6.7.2 Function Documentation

6.7.2.1 Error_Handler()

```
void Error_Handler (
             void )
```

This function is executed in case of error occurrence.

Return values

None	

6.7.2.2 HAL_TIM_MspPostInit()

```
void HAL_TIM_MspPostInit (
             TIM_HandleTypeDef * htim)
```

TIM2 GPIO Configuration PA1 ------> TIM2_CH2

TIM2 GPIO Configuration PA1 ------> TIM2_CH2

6.8 main.h

Go to the documentation of this file.

```
00001 /* USER CODE BEGIN Header */
00019 /* USER CODE END Header */
00020
00021 /* Define to prevent recursive inclusion -------------------------------------*/
00022 #ifndef __MAIN_H
00023 #define __MAIN_H
00024
00025 #ifdef __cplusplus
00026 extern "C"
00027 {
00028 #endif
00029
00030 /* Includes ------------------------------------------------------------------*/
00031 #include "stm32f4xx_hal.h"
00032
00033 /* Private includes ----------------------------------------------------------*/
00034 /* USER CODE BEGIN Includes */
00035 #include "car_control.h"
00036 #include "hc-sr04_read.h"
00037 #include "bluetooth.h"
00038 #include <stdio.h>
00039 #include <stdlib.h>
00040 #include <string.h>
00041   /* USER CODE END Includes */
00042
00043   /* Exported types ------------------------------------------------------------*/
00044   /* USER CODE BEGIN ET */
00045
00046   /* USER CODE END ET */
00047
00048   /* Exported constants --------------------------------------------------------*/
00049   /* USER CODE BEGIN EC */
00050
00051   /* USER CODE END EC */
00052
00053   /* Exported macro ------------------------------------------------------------*/
00054   /* USER CODE BEGIN EM */
00055
00056   /* USER CODE END EM */
00057
00058   void HAL_TIM_MspPostInit(TIM_HandleTypeDef *htim);
00059
00060   /* Exported functions prototypes ---------------------------------------------*/
00061   void Error_Handler(void);
00062
00063 /* USER CODE BEGIN EFP */
00064
00065 /* USER CODE END EFP */
00066
00067 /* Private defines -----------------------------------------------------------*/
00068 #define blue_led_Pin GPIO_PIN_13
00069 #define blue_led_GPIO_Port GPIOC
00070 #define foreward_Pin GPIO_PIN_5
00071 #define foreward_GPIO_Port GPIOA
00072 #define backward_Pin GPIO_PIN_6
00073 #define backward_GPIO_Port GPIOA
00074 #define left_Pin GPIO_PIN_9
00075 #define left_GPIO_Port GPIOA
00076 #define right_Pin GPIO_PIN_10
00077 #define right_GPIO_Port GPIOA
00078
00079   /* USER CODE BEGIN Private defines */
00080
00081   /* USER CODE END Private defines */
00082
00083 #ifdef __cplusplus
00084 }
00085 #endif
00086
00087 #endif /* __MAIN_H */
```

6.9 stm32f4xx_hal_conf.h

```
00001 /* USER CODE BEGIN Header */
00021 /* USER CODE END Header */
00022
00023 /* Define to prevent recursive inclusion -------------------------------------*/
00024 #ifndef __STM32F4xx_HAL_CONF_H
```

```
00025 #define __STM32F4xx_HAL_CONF_H
00026
00027 #ifdef __cplusplus
00028 extern "C"
00029 {
00030 #endif
00031
00032 /* Exported types ------------------------------------------------------------*/
00033 /* Exported constants --------------------------------------------------------*/
00034
00035 /* ########################## Module Selection ############################## */
00039 #define HAL_MODULE_ENABLED
00040
00041   /* #define HAL_CRYP_MODULE_ENABLED */
00042 /* #define HAL_ADC_MODULE_ENABLED */
00043 /* #define HAL_CAN_MODULE_ENABLED */
00044 /* #define HAL_CRC_MODULE_ENABLED */
00045 /* #define HAL_CAN_LEGACY_MODULE_ENABLED */
00046 /* #define HAL_DAC_MODULE_ENABLED */
00047 /* #define HAL_DCMI_MODULE_ENABLED */
00048 /* #define HAL_DMA2D_MODULE_ENABLED */
00049 /* #define HAL_ETH_MODULE_ENABLED */
00050 /* #define HAL_ETH_LEGACY_MODULE_ENABLED */
00051 /* #define HAL_NAND_MODULE_ENABLED */
00052 /* #define HAL_NOR_MODULE_ENABLED */
00053 /* #define HAL_PCCARD_MODULE_ENABLED */
00054 /* #define HAL_SRAM_MODULE_ENABLED */
00055 /* #define HAL_SDRAM_MODULE_ENABLED */
00056 /* #define HAL_HASH_MODULE_ENABLED */
00057 /* #define HAL_I2C_MODULE_ENABLED */
00058 /* #define HAL_I2S_MODULE_ENABLED */
00059 /* #define HAL_IWDG_MODULE_ENABLED */
00060 /* #define HAL_LTDC_MODULE_ENABLED */
00061 /* #define HAL_RNG_MODULE_ENABLED */
00062 /* #define HAL_RTC_MODULE_ENABLED */
00063 /* #define HAL_SAI_MODULE_ENABLED */
00064 /* #define HAL_SD_MODULE_ENABLED */
00065 /* #define HAL_MMC_MODULE_ENABLED */
00066 /* #define HAL_SPI_MODULE_ENABLED */
00067 #define HAL_TIM_MODULE_ENABLED
00068 #define HAL_UART_MODULE_ENABLED
00069 /* #define HAL_USART_MODULE_ENABLED */
00070 /* #define HAL_IRDA_MODULE_ENABLED */
00071 /* #define HAL_SMARTCARD_MODULE_ENABLED */
00072 /* #define HAL_SMBUS_MODULE_ENABLED */
00073 /* #define HAL_WWDG_MODULE_ENABLED */
00074 /* #define HAL_PCD_MODULE_ENABLED */
00075 /* #define HAL_HCD_MODULE_ENABLED */
00076 /* #define HAL_DSI_MODULE_ENABLED */
00077 /* #define HAL_QSPI_MODULE_ENABLED */
00078 /* #define HAL_QSPI_MODULE_ENABLED */
00079 /* #define HAL_CEC_MODULE_ENABLED */
00080 /* #define HAL_FMPI2C_MODULE_ENABLED */
00081 /* #define HAL_FMPSMBUS_MODULE_ENABLED */
00082 /* #define HAL_SPDIFRX_MODULE_ENABLED */
00083 /* #define HAL_DFSDM_MODULE_ENABLED */
00084 /* #define HAL_LPTIM_MODULE_ENABLED */
00085 #define HAL_GPIO_MODULE_ENABLED
00086 #define HAL_EXTI_MODULE_ENABLED
00087 #define HAL_DMA_MODULE_ENABLED
00088 #define HAL_RCC_MODULE_ENABLED
00089 #define HAL_FLASH_MODULE_ENABLED
00090 #define HAL_PWR_MODULE_ENABLED
00091 #define HAL_CORTEX_MODULE_ENABLED
00092
00093 /* ########################## HSE/HSI Values adaptation ##################### */
00099 #if !defined(HSE_VALUE)
00100 #define HSE_VALUE 25000000U
00101 #endif                        /* HSE_VALUE */
00102
00103 #if !defined(HSE_STARTUP_TIMEOUT)
00104 #define HSE_STARTUP_TIMEOUT 100U
00105 #endif                            /* HSE_STARTUP_TIMEOUT */
00106
00112 #if !defined(HSI_VALUE)
00113 #define HSI_VALUE ((uint32_t)16000000U)
00114 #endif                                  /* HSI_VALUE */
00115
00119 #if !defined(LSI_VALUE)
00120 #define LSI_VALUE 32000U
00121 #endif /* LSI_VALUE */
00127 #if !defined(LSE_VALUE)
00128 #define LSE_VALUE 32768U
00129 #endif                    /* LSE_VALUE */
00130
00131 #if !defined(LSE_STARTUP_TIMEOUT)
00132 #define LSE_STARTUP_TIMEOUT 5000U
```

```
00133 #endif                            /* LSE_STARTUP_TIMEOUT */
00134
00140 #if !defined(EXTERNAL_CLOCK_VALUE)
00141 #define EXTERNAL_CLOCK_VALUE 12288000U
00142 #endif                                /* EXTERNAL_CLOCK_VALUE */
00143
00144 /* Tip: To avoid modifying this file each time you need to use different HSE,
00145    ===  you can define the HSE value in your toolchain compiler preprocessor. */
00146
00147 /* ########################## System Configuration ######################### */
00151 #define VDD_VALUE 3300U
00152 #define TICK_INT_PRIORITY 15U
00153 #define USE_RTOS 0U
00154 #define PREFETCH_ENABLE 1U
00155 #define INSTRUCTION_CACHE_ENABLE 1U
00156 #define DATA_CACHE_ENABLE 1U
00157
00158 #define USE_HAL_ADC_REGISTER_CALLBACKS 0U       /* ADC register callback disabled       */
00159 #define USE_HAL_CAN_REGISTER_CALLBACKS 0U       /* CAN register callback disabled       */
00160 #define USE_HAL_CEC_REGISTER_CALLBACKS 0U       /* CEC register callback disabled       */
00161 #define USE_HAL_CRYP_REGISTER_CALLBACKS 0U      /* CRYP register callback disabled      */
00162 #define USE_HAL_DAC_REGISTER_CALLBACKS 0U       /* DAC register callback disabled       */
00163 #define USE_HAL_DCMI_REGISTER_CALLBACKS 0U      /* DCMI register callback disabled      */
00164 #define USE_HAL_DFSDM_REGISTER_CALLBACKS 0U     /* DFSDM register callback disabled     */
00165 #define USE_HAL_DMA2D_REGISTER_CALLBACKS 0U     /* DMA2D register callback disabled     */
00166 #define USE_HAL_DSI_REGISTER_CALLBACKS 0U       /* DSI register callback disabled       */
00167 #define USE_HAL_ETH_REGISTER_CALLBACKS 0U       /* ETH register callback disabled       */
00168 #define USE_HAL_HASH_REGISTER_CALLBACKS 0U      /* HASH register callback disabled      */
00169 #define USE_HAL_HCD_REGISTER_CALLBACKS 0U       /* HCD register callback disabled       */
00170 #define USE_HAL_I2C_REGISTER_CALLBACKS 0U       /* I2C register callback disabled       */
00171 #define USE_HAL_FMPI2C_REGISTER_CALLBACKS 0U    /* FMPI2C register callback disabled    */
00172 #define USE_HAL_FMPSMBUS_REGISTER_CALLBACKS 0U  /* FMPSMBUS register callback disabled  */
00173 #define USE_HAL_I2S_REGISTER_CALLBACKS 0U       /* I2S register callback disabled       */
00174 #define USE_HAL_IRDA_REGISTER_CALLBACKS 0U      /* IRDA register callback disabled      */
00175 #define USE_HAL_LPTIM_REGISTER_CALLBACKS 0U     /* LPTIM register callback disabled     */
00176 #define USE_HAL_LTDC_REGISTER_CALLBACKS 0U      /* LTDC register callback disabled      */
00177 #define USE_HAL_MMC_REGISTER_CALLBACKS 0U       /* MMC register callback disabled       */
00178 #define USE_HAL_NAND_REGISTER_CALLBACKS 0U      /* NAND register callback disabled      */
00179 #define USE_HAL_NOR_REGISTER_CALLBACKS 0U       /* NOR register callback disabled       */
00180 #define USE_HAL_PCCARD_REGISTER_CALLBACKS 0U    /* PCCARD register callback disabled    */
00181 #define USE_HAL_PCD_REGISTER_CALLBACKS 0U       /* PCD register callback disabled       */
00182 #define USE_HAL_QSPI_REGISTER_CALLBACKS 0U      /* QSPI register callback disabled      */
00183 #define USE_HAL_RNG_REGISTER_CALLBACKS 0U       /* RNG register callback disabled       */
00184 #define USE_HAL_RTC_REGISTER_CALLBACKS 0U       /* RTC register callback disabled       */
00185 #define USE_HAL_SAI_REGISTER_CALLBACKS 0U       /* SAI register callback disabled       */
00186 #define USE_HAL_SD_REGISTER_CALLBACKS 0U        /* SD register callback disabled        */
00187 #define USE_HAL_SMARTCARD_REGISTER_CALLBACKS 0U /* SMARTCARD register callback disabled */
00188 #define USE_HAL_SDRAM_REGISTER_CALLBACKS 0U     /* SDRAM register callback disabled     */
00189 #define USE_HAL_SRAM_REGISTER_CALLBACKS 0U      /* SRAM register callback disabled      */
00190 #define USE_HAL_SPDIFRX_REGISTER_CALLBACKS 0U   /* SPDIFRX register callback disabled   */
00191 #define USE_HAL_SMBUS_REGISTER_CALLBACKS 0U     /* SMBUS register callback disabled     */
00192 #define USE_HAL_SPI_REGISTER_CALLBACKS 0U       /* SPI register callback disabled       */
00193 #define USE_HAL_TIM_REGISTER_CALLBACKS 0U       /* TIM register callback disabled       */
00194 #define USE_HAL_UART_REGISTER_CALLBACKS 0U      /* UART register callback disabled      */
00195 #define USE_HAL_USART_REGISTER_CALLBACKS 0U     /* USART register callback disabled     */
00196 #define USE_HAL_WWDG_REGISTER_CALLBACKS 0U      /* WWDG register callback disabled      */
00197
00198 /* ########################## Assert Selection ############################## */
00203 /* #define USE_FULL_ASSERT    1U */
00204
00205 /* ################### Ethernet peripheral configuration ##################### */
00206
00207 /* Section 1 : Ethernet peripheral configuration */
00208
00209 /* MAC ADDRESS: MAC_ADDR0:MAC_ADDR1:MAC_ADDR2:MAC_ADDR3:MAC_ADDR4:MAC_ADDR5 */
00210 #define MAC_ADDR0 2U
00211 #define MAC_ADDR1 0U
00212 #define MAC_ADDR2 0U
00213 #define MAC_ADDR3 0U
00214 #define MAC_ADDR4 0U
00215 #define MAC_ADDR5 0U
00216
00217 /* Definition of the Ethernet driver buffers size and count */
00218 #define ETH_RX_BUF_SIZE                     /* buffer size for receive               */
00219 #define ETH_TX_BUF_SIZE ETH_MAX_PACKET_SIZE /* buffer size for transmit              */
00220 #define ETH_RXBUFNB 4U                      /* 4 Rx buffers of size ETH_RX_BUF_SIZE  */
00221 #define ETH_TXBUFNB 4U                      /* 4 Tx buffers of size ETH_TX_BUF_SIZE  */
00222
00223 /* Section 2: PHY configuration section */
00224
00225 /* DP83848_PHY_ADDRESS Address*/
00226 #define DP83848_PHY_ADDRESS
00227 /* PHY Reset delay these values are based on a 1 ms Systick interrupt*/
00228 #define PHY_RESET_DELAY 0x000000FFU
00229 /* PHY Configuration delay */
00230 #define PHY_CONFIG_DELAY 0x00000FFFU
00231
```

```
00232 #define PHY_READ_TO 0x0000FFFFU
00233 #define PHY_WRITE_TO 0x0000FFFFU
00234
00235   /* Section 3: Common PHY Registers */
00236
00237 #define PHY_BCR ((uint16_t)0x0000U)
00238 #define PHY_BSR ((uint16_t)0x0001U)
00240 #define PHY_RESET ((uint16_t)0x8000U)
00241 #define PHY_LOOPBACK ((uint16_t)0x4000U)
00242 #define PHY_FULLDUPLEX_100M ((uint16_t)0x2100U)
00243 #define PHY_HALFDUPLEX_100M ((uint16_t)0x2000U)
00244 #define PHY_FULLDUPLEX_10M ((uint16_t)0x0100U)
00245 #define PHY_HALFDUPLEX_10M ((uint16_t)0x0000U)
00246 #define PHY_AUTONEGOTIATION ((uint16_t)0x1000U)
00247 #define PHY_RESTART_AUTONEGOTIATION ((uint16_t)0x0200U)
00248 #define PHY_POWERDOWN ((uint16_t)0x0800U)
00249 #define PHY_ISOLATE ((uint16_t)0x0400U)
00251 #define PHY_AUTONEGO_COMPLETE ((uint16_t)0x0020U)
00252 #define PHY_LINKED_STATUS ((uint16_t)0x0004U)
00253 #define PHY_JABBER_DETECTION ((uint16_t)0x0002U)
00255 /* Section 4: Extended PHY Registers */
00256 #define PHY_SR ((uint16_t))
00258 #define PHY_SPEED_STATUS ((uint16_t))
00259 #define PHY_DUPLEX_STATUS ((uint16_t))
00261   /* ################## SPI peripheral configuration ########################## */
00262
00263   /* CRC FEATURE: Use to activate CRC feature inside HAL SPI Driver
00264    * Activated: CRC code is present inside driver
00265    * Deactivated: CRC code cleaned from driver
00266    */
00267
00268 #define USE_SPI_CRC 0U
00269
00270   /* Includes ------------------------------------------------------------------*/
00275 #ifdef HAL_RCC_MODULE_ENABLED
00276 #include "stm32f4xx_hal_rcc.h"
00277 #endif /* HAL_RCC_MODULE_ENABLED */
00278
00279 #ifdef HAL_GPIO_MODULE_ENABLED
00280 #include "stm32f4xx_hal_gpio.h"
00281 #endif /* HAL_GPIO_MODULE_ENABLED */
00282
00283 #ifdef HAL_EXTI_MODULE_ENABLED
00284 #include "stm32f4xx_hal_exti.h"
00285 #endif /* HAL_EXTI_MODULE_ENABLED */
00286
00287 #ifdef HAL_DMA_MODULE_ENABLED
00288 #include "stm32f4xx_hal_dma.h"
00289 #endif /* HAL_DMA_MODULE_ENABLED */
00290
00291 #ifdef HAL_CORTEX_MODULE_ENABLED
00292 #include "stm32f4xx_hal_cortex.h"
00293 #endif /* HAL_CORTEX_MODULE_ENABLED */
00294
00295 #ifdef HAL_ADC_MODULE_ENABLED
00296 #include "stm32f4xx_hal_adc.h"
00297 #endif /* HAL_ADC_MODULE_ENABLED */
00298
00299 #ifdef HAL_CAN_MODULE_ENABLED
00300 #include "stm32f4xx_hal_can.h"
00301 #endif /* HAL_CAN_MODULE_ENABLED */
00302
00303 #ifdef HAL_CAN_LEGACY_MODULE_ENABLED
00304 #include "stm32f4xx_hal_can_legacy.h"
00305 #endif /* HAL_CAN_LEGACY_MODULE_ENABLED */
00306
00307 #ifdef HAL_CRC_MODULE_ENABLED
00308 #include "stm32f4xx_hal_crc.h"
00309 #endif /* HAL_CRC_MODULE_ENABLED */
00310
00311 #ifdef HAL_CRYP_MODULE_ENABLED
00312 #include "stm32f4xx_hal_cryp.h"
00313 #endif /* HAL_CRYP_MODULE_ENABLED */
00314
00315 #ifdef HAL_DMA2D_MODULE_ENABLED
00316 #include "stm32f4xx_hal_dma2d.h"
00317 #endif /* HAL_DMA2D_MODULE_ENABLED */
00318
00319 #ifdef HAL_DAC_MODULE_ENABLED
00320 #include "stm32f4xx_hal_dac.h"
00321 #endif /* HAL_DAC_MODULE_ENABLED */
00322
00323 #ifdef HAL_DCMI_MODULE_ENABLED
00324 #include "stm32f4xx_hal_dcmi.h"
00325 #endif /* HAL_DCMI_MODULE_ENABLED */
00326
00327 #ifdef HAL_ETH_MODULE_ENABLED
```

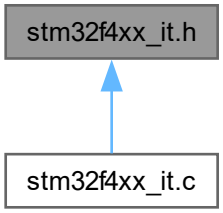
```
00328 #include "stm32f4xx_hal_eth.h"
00329 #endif /* HAL_ETH_MODULE_ENABLED */
00330
00331 #ifdef HAL_ETH_LEGACY_MODULE_ENABLED
00332 #include "stm32f4xx_hal_eth_legacy.h"
00333 #endif /* HAL_ETH_LEGACY_MODULE_ENABLED */
00334
00335 #ifdef HAL_FLASH_MODULE_ENABLED
00336 #include "stm32f4xx_hal_flash.h"
00337 #endif /* HAL_FLASH_MODULE_ENABLED */
00338
00339 #ifdef HAL_SRAM_MODULE_ENABLED
00340 #include "stm32f4xx_hal_sram.h"
00341 #endif /* HAL_SRAM_MODULE_ENABLED */
00342
00343 #ifdef HAL_NOR_MODULE_ENABLED
00344 #include "stm32f4xx_hal_nor.h"
00345 #endif /* HAL_NOR_MODULE_ENABLED */
00346
00347 #ifdef HAL_NAND_MODULE_ENABLED
00348 #include "stm32f4xx_hal_nand.h"
00349 #endif /* HAL_NAND_MODULE_ENABLED */
00350
00351 #ifdef HAL_PCCARD_MODULE_ENABLED
00352 #include "stm32f4xx_hal_pccard.h"
00353 #endif /* HAL_PCCARD_MODULE_ENABLED */
00354
00355 #ifdef HAL_SDRAM_MODULE_ENABLED
00356 #include "stm32f4xx_hal_sdram.h"
00357 #endif /* HAL_SDRAM_MODULE_ENABLED */
00358
00359 #ifdef HAL_HASH_MODULE_ENABLED
00360 #include "stm32f4xx_hal_hash.h"
00361 #endif /* HAL_HASH_MODULE_ENABLED */
00362
00363 #ifdef HAL_I2C_MODULE_ENABLED
00364 #include "stm32f4xx_hal_i2c.h"
00365 #endif /* HAL_I2C_MODULE_ENABLED */
00366
00367 #ifdef HAL_SMBUS_MODULE_ENABLED
00368 #include "stm32f4xx_hal_smbus.h"
00369 #endif /* HAL_SMBUS_MODULE_ENABLED */
00370
00371 #ifdef HAL_I2S_MODULE_ENABLED
00372 #include "stm32f4xx_hal_i2s.h"
00373 #endif /* HAL_I2S_MODULE_ENABLED */
00374
00375 #ifdef HAL_IWDG_MODULE_ENABLED
00376 #include "stm32f4xx_hal_iwdg.h"
00377 #endif /* HAL_IWDG_MODULE_ENABLED */
00378
00379 #ifdef HAL_LTDC_MODULE_ENABLED
00380 #include "stm32f4xx_hal_ltdc.h"
00381 #endif /* HAL_LTDC_MODULE_ENABLED */
00382
00383 #ifdef HAL_PWR_MODULE_ENABLED
00384 #include "stm32f4xx_hal_pwr.h"
00385 #endif /* HAL_PWR_MODULE_ENABLED */
00386
00387 #ifdef HAL_RNG_MODULE_ENABLED
00388 #include "stm32f4xx_hal_rng.h"
00389 #endif /* HAL_RNG_MODULE_ENABLED */
00390
00391 #ifdef HAL_RTC_MODULE_ENABLED
00392 #include "stm32f4xx_hal_rtc.h"
00393 #endif /* HAL_RTC_MODULE_ENABLED */
00394
00395 #ifdef HAL_SAI_MODULE_ENABLED
00396 #include "stm32f4xx_hal_sai.h"
00397 #endif /* HAL_SAI_MODULE_ENABLED */
00398
00399 #ifdef HAL_SD_MODULE_ENABLED
00400 #include "stm32f4xx_hal_sd.h"
00401 #endif /* HAL_SD_MODULE_ENABLED */
00402
00403 #ifdef HAL_SPI_MODULE_ENABLED
00404 #include "stm32f4xx_hal_spi.h"
00405 #endif /* HAL_SPI_MODULE_ENABLED */
00406
00407 #ifdef HAL_TIM_MODULE_ENABLED
00408 #include "stm32f4xx_hal_tim.h"
00409 #endif /* HAL_TIM_MODULE_ENABLED */
00410
00411 #ifdef HAL_UART_MODULE_ENABLED
00412 #include "stm32f4xx_hal_uart.h"
00413 #endif /* HAL_UART_MODULE_ENABLED */
00414
```

```
00415 #ifdef HAL_USART_MODULE_ENABLED
00416 #include "stm32f4xx_hal_usart.h"
00417 #endif /* HAL_USART_MODULE_ENABLED */
00418
00419 #ifdef HAL_IRDA_MODULE_ENABLED
00420 #include "stm32f4xx_hal_irda.h"
00421 #endif /* HAL_IRDA_MODULE_ENABLED */
00422
00423 #ifdef HAL_SMARTCARD_MODULE_ENABLED
00424 #include "stm32f4xx_hal_smartcard.h"
00425 #endif /* HAL_SMARTCARD_MODULE_ENABLED */
00426
00427 #ifdef HAL_WWDG_MODULE_ENABLED
00428 #include "stm32f4xx_hal_wwdg.h"
00429 #endif /* HAL_WWDG_MODULE_ENABLED */
00430
00431 #ifdef HAL_PCD_MODULE_ENABLED
00432 #include "stm32f4xx_hal_pcd.h"
00433 #endif /* HAL_PCD_MODULE_ENABLED */
00434
00435 #ifdef HAL_HCD_MODULE_ENABLED
00436 #include "stm32f4xx_hal_hcd.h"
00437 #endif /* HAL_HCD_MODULE_ENABLED */
00438
00439 #ifdef HAL_DSI_MODULE_ENABLED
00440 #include "stm32f4xx_hal_dsi.h"
00441 #endif /* HAL_DSI_MODULE_ENABLED */
00442
00443 #ifdef HAL_QSPI_MODULE_ENABLED
00444 #include "stm32f4xx_hal_qspi.h"
00445 #endif /* HAL_QSPI_MODULE_ENABLED */
00446
00447 #ifdef HAL_CEC_MODULE_ENABLED
00448 #include "stm32f4xx_hal_cec.h"
00449 #endif /* HAL_CEC_MODULE_ENABLED */
00450
00451 #ifdef HAL_FMPI2C_MODULE_ENABLED
00452 #include "stm32f4xx_hal_fmpi2c.h"
00453 #endif /* HAL_FMPI2C_MODULE_ENABLED */
00454
00455 #ifdef HAL_FMPSMBUS_MODULE_ENABLED
00456 #include "stm32f4xx_hal_fmpsmbus.h"
00457 #endif /* HAL_FMPSMBUS_MODULE_ENABLED */
00458
00459 #ifdef HAL_SPDIFRX_MODULE_ENABLED
00460 #include "stm32f4xx_hal_spdifrx.h"
00461 #endif /* HAL_SPDIFRX_MODULE_ENABLED */
00462
00463 #ifdef HAL_DFSDM_MODULE_ENABLED
00464 #include "stm32f4xx_hal_dfsdm.h"
00465 #endif /* HAL_DFSDM_MODULE_ENABLED */
00466
00467 #ifdef HAL_LPTIM_MODULE_ENABLED
00468 #include "stm32f4xx_hal_lptim.h"
00469 #endif /* HAL_LPTIM_MODULE_ENABLED */
00470
00471 #ifdef HAL_MMC_MODULE_ENABLED
00472 #include "stm32f4xx_hal_mmc.h"
00473 #endif /* HAL_MMC_MODULE_ENABLED */
00474
00475 /* Exported macro ------------------------------------------------------------*/
00476 #ifdef USE_FULL_ASSERT
00485 #define assert_param(expr) ((expr) ? (void)0U : assert_failed((uint8_t *)__FILE__, __LINE__))
00486   /* Exported functions ------------------------------------------------------- */
00487   void assert_failed(uint8_t *file, uint32_t line);
00488 #else
00489 #define assert_param(expr) ((void)0U)
00490 #endif /* USE_FULL_ASSERT */
00491
00492 #ifdef __cplusplus
00493 }
00494 #endif
00495
00496 #endif /* __STM32F4xx_HAL_CONF_H */
```

6.10 stm32f4xx_it.h File Reference

This file contains the headers of the interrupt handlers.

This graph shows which files directly or indirectly include this file:



Functions

- void **NMI_Handler** (void)

  This function handles Non maskable interrupt.
- void **HardFault_Handler** (void)

  This function handles Hard fault interrupt.
- void **MemManage_Handler** (void)

  This function handles Memory management fault.
- void **BusFault_Handler** (void)

  This function handles Pre-fetch fault, memory access fault.
- void **UsageFault_Handler** (void)

  This function handles Undefined instruction or illegal state.
- void **SVC_Handler** (void)

  This function handles System service call via SWI instruction.
- void **DebugMon_Handler** (void)

  This function handles Debug monitor.
- void **PendSV_Handler** (void)

  This function handles Pendable request for system service.
- void **SysTick_Handler** (void)

  This function handles System tick timer.
- void TIM2_IRQHandler (void)

  This function handles TIM2 global interrupt, which is the main loop of the program.
- void **USART2_IRQHandler** (void)

  This function handles USART2 global interrupt.

6.10.1 Detailed Description

This file contains the headers of the interrupt handlers.

Attention

6.10.2 Function Documentation

6.10.2.1 TIM2_IRQHandler()

```
void TIM2_IRQHandler (
            void )
```

This function handles TIM2 global interrupt, which is the main loop of the program.

Note

In the event of interrupt, obstacle_detection() pass measured distance to car struct. Then its send by bluetooth_send(). After that based on the received data from bluetooth, car_control() function is steering car accordingly

Parameters

None	

Return values

None	

< Measured distance is passed to car struct

6.11 stm32f4xx_it.h

Go to the documentation of this file.

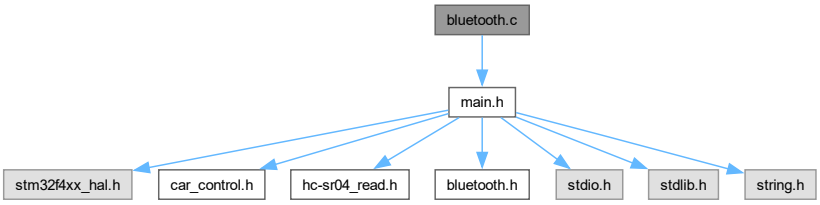
```
00001 /* USER CODE BEGIN Header */
00018 /* USER CODE END Header */
00019
00020 /* Define to prevent recursive inclusion -------------------------------------*/
00021 #ifndef __STM32F4xx_IT_H
00022 #define __STM32F4xx_IT_H
00023
00024 #ifdef __cplusplus
00025 extern "C"
00026 {
00027 #endif
00028
00029   /* Private includes ----------------------------------------------------------*/
00030   /* USER CODE BEGIN Includes */
00031
00032   /* USER CODE END Includes */
00033
00034   /* Exported types ------------------------------------------------------------*/
00035   /* USER CODE BEGIN ET */
00036
00037   /* USER CODE END ET */
00038
00039   /* Exported constants --------------------------------------------------------*/
00040   /* USER CODE BEGIN EC */
00041
00042   /* USER CODE END EC */
00043
00044   /* Exported macro ------------------------------------------------------------*/
00045   /* USER CODE BEGIN EM */
00046
00047   /* USER CODE END EM */
00048
00049   /* Exported functions prototypes ---------------------------------------------*/
00050   void NMI_Handler(void);
00051   void HardFault_Handler(void);
00052   void MemManage_Handler(void);
```

```
00053   void BusFault_Handler(void);
00054   void UsageFault_Handler(void);
00055   void SVC_Handler(void);
00056   void DebugMon_Handler(void);
00057   void PendSV_Handler(void);
00058   void SysTick_Handler(void);
00059   void TIM2_IRQHandler(void);
00060   void USART2_IRQHandler(void);
00061   /* USER CODE BEGIN EFP */
00062
00063   /* USER CODE END EFP */
00064
00065 #ifdef __cplusplus
00066 }
00067 #endif
00068
00069 #endif /* __STM32F4xx_IT_H */
```

6.12 bluetooth.c File Reference

This file contains all functions used to connect to the Bluetooth.

`#include "main.h"`
Include dependency graph for bluetooth.c:



Macros

- #define **LINE_MAX_LENGTH** 64

Functions

- void HAL_UART_RxCpltCallback (UART_HandleTypeDef ∗huart)

  This function is called as interrupt when receiving data.
- void read_logic (void)

  Function to process the received line of data.
- void line_append (uint8_t value)

  Function to append a received byte to the line buffer.
- void bluetooth_send (void)

  Function to send a Bluetooth message.

Variables

- UART_HandleTypeDef **huart2**
- uint8_t received
- uint8_t message [64]
- uint16_t message_size = 0

6.12.1 Detailed Description

This file contains all functions used to connect to the Bluetooth.

Author

Jacek Bielski && Jakub Wysocki

Version

0.1

Date

2024-05-09

Copyright

Copyright (c) 2024

6.12.2 Function Documentation

6.12.2.1 bluetooth_send()

```
void bluetooth_send (
            void )
```

Function to send a Bluetooth message.

This function formats and sends a message containing the car's obstacle distance over Bluetooth using the UART transmit interrupt.

6.12.2.2 HAL_UART_RxCpltCallback()

```
void HAL_UART_RxCpltCallback (
            UART_HandleTypeDef * huart)
```

This function is called as interrupt when receiving data.

Callback function for UART receive complete interrupt.

It checks if the received data is from USART2 and processes the received byte.

Parameters

huart	

6.12.2.3 line_append()

```
void line_append (
            uint8_t value)
```

Function to append a received byte to the line buffer.

This function adds a received byte to the line buffer. If the byte is a newline or carriage return, it processes the complete line.

Parameters

value	

6.12.2.4 read_logic()

```
void read_logic (
            void )
```

Function to process the received line of data.

This function processes the command received from the Bluetooth module. Depending on the command, it adjusts the car's:

- turn angle (left, right, idle),
- toggles lights (on, off),
- ride mode (foreward, backward, idle).

6.12.3 Variable Documentation

6.12.3.1 message

```
uint8_t message[64]
```

Buffer for storing the message to be sent over Bluetooth.

6.12.3.2 message_size

```
uint16_t message_size = 0
```

Size of the message stored in the buffer.

6.12.3.3 received

```
uint8_t received  [extern]
```

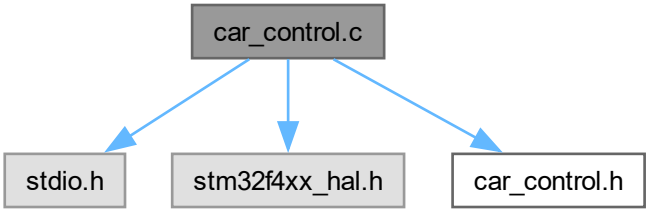
Variable to store the received data byte.

6.13 car_control.c File Reference

This file contains all functions to control the car.

```
#include <stdio.h>
#include "stm32f4xx_hal.h"
#include "car_control.h"
```

Include dependency graph for car_control.c:



Macros

- #define **blue_led_Pin** GPIO_PIN_13
- #define **blue_led_GPIO_Port** GPIOC
- #define **foreward_Pin** GPIO_PIN_5
- #define **foreward_GPIO_Port** GPIOA
- #define **backward_Pin** GPIO_PIN_6
- #define **backward_GPIO_Port** GPIOA
- #define **left_Pin** GPIO_PIN_9
- #define **left_GPIO_Port** GPIOA
- #define **right_Pin** GPIO_PIN_10
- #define **right_GPIO_Port** GPIOA

Functions

- void **go_idle** (void)

  Set the car to idle state.
- void **go_foreward** (void)

  Set the car to move forward.
- void **go_backward** (void)

  Set the car to move backward.
- void **turn_idle** (void)

  Set the car to idle state for turning.
- void **turn_left** (void)

  Turn the car to the left.
- void **turn_right** (void)

  Turn the car to the right.
- void **car_init** (void)

  Initialize the car by setting it to idle state.
- void car_control (void)

  Control the car based on its current state.

6.13.1 Detailed Description

This file contains all functions to control the car.

Author

Jacek Bielski && Jakub Wysocki

Version

0.1

Date

2024-05-09

Copyright

Copyright (c) 2024

6.13.2 Function Documentation

6.13.2.1 car_control()

```
void car_control (
            void )
```

Control the car based on its current state.

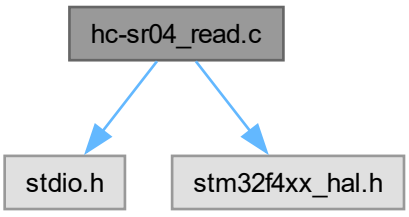
This function contains logic for steering the car, based on the current state of the car struct. It checks the `car.ride` state to determine the movement of the car (forward, backward, or idle). It also checks the `car.turn` state to determine the direction of the car (left, right, or idle).

6.14 hc-sr04_read.c File Reference

Contains all functions for reading distances from the HC-SR04 module.

```
#include <stdio.h>
#include "stm32f4xx_hal.h"
```

Include dependency graph for hc-sr04_read.c:



Functions

- void hcsr04_read_init (void)

  Initializes the three TIM2 channels used to correctly read distances from the HC-SR04 module.
- float obstacle_detection (void)

  Calculates distance based on the width of the Echo signal.

Variables

- TIM_HandleTypeDef **htim2**

6.14.1 Detailed Description

Contains all functions for reading distances from the HC-SR04 module.

Version

0.1

Date

2024-05-09

The file contains functions necessary for initialization and reading distances from the HC-SR04 module using STM32 microcontroller and TIM2 timer.

Authors

https://forbot.pl/blog/kurs-stm32l4-czujnik-odleglosci-wyswietlacz-7-seg-id48628

Copyright

6.14.2 Function Documentation

6.14.2.1 hcsr04_read_init()

```
void hcsr04_read_init (
            void )
```

Initializes the three TIM2 channels used to correctly read distances from the HC-SR04 module.

Initialize HC-SR04 sensor for distance measurement.

The function activates three TIM2 channels: two for capturing the Echo signal (start and stop) and one to generate a trigger pulse for the HC-SR04 module.

6.14.2.2 obstacle_detection()

```
float obstacle_detection (
            void )
```

Calculates distance based on the width of the Echo signal.

Perform obstacle detection using HC-SR04 sensor.

The function calculates the distance based on the time difference between the moment the Echo signal appears and its disappearance. The result is converted to centimeters.

Returns

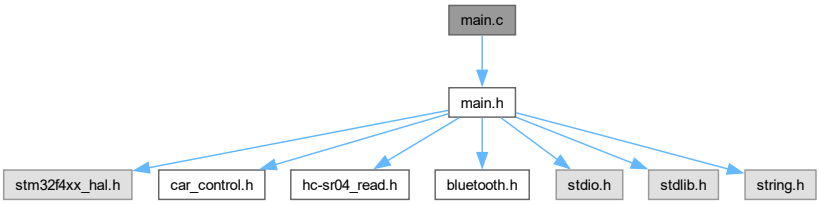
float Calculated distance in centimeters.

6.15 main.c File Reference

: Main program body of the application for PMIK project

```
#include "main.h"
```

Include dependency graph for main.c:



Functions

- void SystemClock_Config (void)

  System Clock Configuration.
- int main (void)

  The application entry point.
- void Error_Handler (void)

  This function is executed in case of error occurrence.

Variables

- TIM_HandleTypeDef **htim2**
- UART_HandleTypeDef **huart2**
- carStruct car = {200, 0, 0, 0}
- uint8_t received = 0

6.15.1 Detailed Description

: Main program body of the application for PMIK project

Author

: Jacek Bielski

: Jakub Wysocki

Note

The aim of the project is to remotely control a toy car using bluetooth, This program was made as student project of the Warsaw University of Technology as part of the PMIK subject (Programming microcontrollers)

In main function all peripherals are initialized, then program works on interupts from tim2 and usart2

Attention

This program is intendet to run on blackpill stm32f411

In order to properly run the program make sure that all crutial periphentials are enabled such as tim2, usart2, Make sure that bluetooth.h, car_control.h, hc-sr04_read.h are included to main.h

6.15.2 Function Documentation

6.15.2.1 Error_Handler()

```
void Error_Handler (
             void )
```

This function is executed in case of error occurrence.

Return values

None	

6.15.2.2 main()

```
int main (
             void )
```

The application entry point.

Return values

int	

Note

This function initializes all used peripherals

Then it contains empty infinite while loop

6.15.2.3 SystemClock_Config()

```
void SystemClock_Config (
             void )
```

System Clock Configuration.

Return values

None	

Configure the main internal regulator output voltage

Initializes the RCC Oscillators according to the specified parameters in the RCC_OscInitTypeDef structure.

Initializes the CPU, AHB and APB buses clocks

6.15.3 Variable Documentation

6.15.3.1 car

```
carStruct car = {200, 0, 0, 0}
```

External reference to the car state

6.15.3.2 received

```
uint8_t received = 0
```

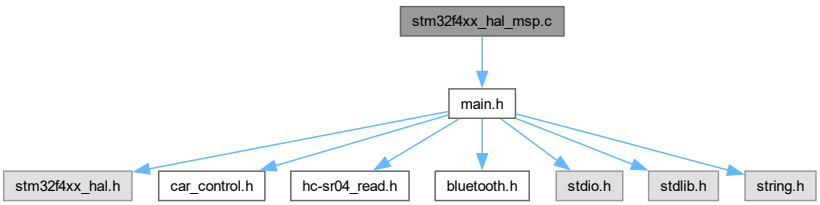
Variable to store the received data byte.

6.16 stm32f4xx_hal_msp.c File Reference

This file provides code for the MSP Initialization and de-Initialization codes.

```
#include "main.h"
```

Include dependency graph for stm32f4xx_hal_msp.c:



Functions

- void HAL_TIM_MspPostInit (TIM_HandleTypeDef ∗htim)
- void HAL_MspInit (void)
- void HAL_TIM_Base_MspInit (TIM_HandleTypeDef ∗htim_base)

  TIM_Base MSP Initialization This function configures the hardware resources used in this example.
- void HAL_TIM_Base_MspDeInit (TIM_HandleTypeDef ∗htim_base)

  TIM_Base MSP De-Initialization This function freeze the hardware resources used in this example.
- void HAL_UART_MspInit (UART_HandleTypeDef ∗huart)

  UART MSP Initialization This function configures the hardware resources used in this example.
- void HAL_UART_MspDeInit (UART_HandleTypeDef ∗huart)

  UART MSP De-Initialization This function freeze the hardware resources used in this example.

6.16.1 Detailed Description

This file provides code for the MSP Initialization and de-Initialization codes.

Attention

6.16.2 Function Documentation

6.16.2.1 HAL_MspInit()

```
void HAL_MspInit (
             void )
```

Initializes the Global MSP.

6.16.2.2 HAL_TIM_Base_MspDeInit()

```
void HAL_TIM_Base_MspDeInit (
             TIM_HandleTypeDef * htim_base)
```

TIM_Base MSP De-Initialization This function freeze the hardware resources used in this example.

Parameters

htim_base	TIM_Base handle pointer

Return values

None	

TIM2 GPIO Configuration PA1 ------> TIM2_CH2 PB10 ------> TIM2_CH3

6.16.2.3 HAL_TIM_Base_MspInit()

```
void HAL_TIM_Base_MspInit (
            TIM_HandleTypeDef * htim_base)
```

TIM_Base MSP Initialization This function configures the hardware resources used in this example.

Parameters

htim_base	TIM_Base handle pointer

Return values

None	

TIM2 GPIO Configuration PB10 ------> TIM2_CH3

6.16.2.4 HAL_TIM_MspPostInit()

```
void HAL_TIM_MspPostInit (
            TIM_HandleTypeDef * htim)
```

TIM2 GPIO Configuration PA1 ------> TIM2_CH2

6.16.2.5 HAL_UART_MspDeInit()

```
void HAL_UART_MspDeInit (
            UART_HandleTypeDef * huart)
```

UART MSP De-Initialization This function freeze the hardware resources used in this example.

Parameters

huart	UART handle pointer

Return values

None	

USART2 GPIO Configuration PA2 ------> USART2_TX PA3 ------> USART2_RX

6.16.2.6 HAL_UART_MspInit()

```
void HAL_UART_MspInit (
            UART_HandleTypeDef * huart)
```

UART MSP Initialization This function configures the hardware resources used in this example.

Parameters

huart	UART handle pointer

Return values

None	

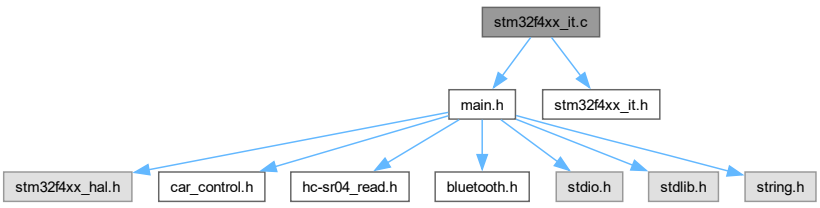
USART2 GPIO Configuration PA2 ------> USART2_TX PA3 ------> USART2_RX

6.17 stm32f4xx_it.c File Reference

Interrupt Service Routines.

```
#include "main.h"
#include "stm32f4xx_it.h"
```

Include dependency graph for stm32f4xx_it.c:



Functions

- void **NMI_Handler** (void)

  This function handles Non maskable interrupt.
- void **HardFault_Handler** (void)

  This function handles Hard fault interrupt.
- void **MemManage_Handler** (void)

  This function handles Memory management fault.
- void **BusFault_Handler** (void)

  This function handles Pre-fetch fault, memory access fault.
- void **UsageFault_Handler** (void)

  This function handles Undefined instruction or illegal state.
- void **SVC_Handler** (void)

  This function handles System service call via SWI instruction.
- void **DebugMon_Handler** (void)

  This function handles Debug monitor.
- void **PendSV_Handler** (void)

  This function handles Pendable request for system service.
- void **SysTick_Handler** (void)

  This function handles System tick timer.
- void TIM2_IRQHandler (void)

  This function handles TIM2 global interrupt, which is the main loop of the program.
- void **USART2_IRQHandler** (void)

  This function handles USART2 global interrupt.

Variables

- TIM_HandleTypeDef **htim2**
- UART_HandleTypeDef **huart2**

6.17.1 Detailed Description

Interrupt Service Routines.

Attention

6.17.2 Function Documentation

6.17.2.1 TIM2_IRQHandler()

```
void TIM2_IRQHandler (
            void )
```

This function handles TIM2 global interrupt, which is the main loop of the program.

Note

In the event of interrupt, obstacle_detection() pass measured distance to car struct. Then its send by bluetooth_send(). After that based on the received data from bluetooth, car_control() function is steering car accordingly

Parameters

None	

Return values

None	

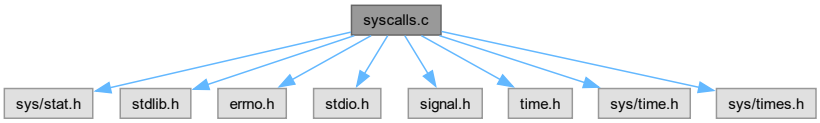
< Measured distance is passed to car struct

6.18 syscalls.c File Reference

STM32CubeIDE Minimal System calls file.

```
#include <sys/stat.h>
#include <stdlib.h>
#include <errno.h>
#include <stdio.h>
#include <signal.h>
#include <time.h>
#include <sys/time.h>
#include <sys/times.h>
```

Include dependency graph for syscalls.c:



Functions

- int **__io_putchar** (int ch) __attribute__((weak))
- int **__io_getchar** (void)
- void **initialise_monitor_handles** ()
- int **_getpid** (void)
- int **_kill** (int pid, int sig)
- void **_exit** (int status)
- **__attribute__** ((weak))
- int **_close** (int file)
- int **_fstat** (int file, struct stat *st)
- int **_isatty** (int file)
- int **_lseek** (int file, int ptr, int dir)
- int **_open** (char *path, int flags,...)
- int **_wait** (int *status)
- int **_unlink** (char *name)
- int **_times** (struct tms *buf)
- int **_stat** (char *file, struct stat *st)
- int **_link** (char *old, char *new)
- int **_fork** (void)
- int **_execve** (char *name, char **argv, char **env)

Variables

- char ** **environ** = __env

6.18.1 Detailed Description

STM32CubeIDE Minimal System calls file.

Author

Auto-generated by STM32CubeIDE

```
For more information about which c-functions
need which of these lowlevel functions
please consult the Newlib libc-manual
```

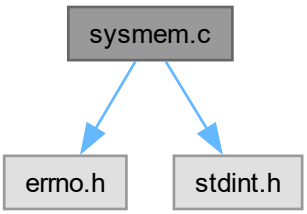
Attention

6.19 sysmem.c File Reference

STM32CubeIDE System Memory calls file.

```
#include <errno.h>
#include <stdint.h>
```

Include dependency graph for sysmem.c:



Functions

- void * _sbrk (ptrdiff_t incr)

  _sbrk() allocates memory to the newlib heap and is used by malloc and others from the C library

6.19.1 Detailed Description

STM32CubeIDE System Memory calls file.

Author

Generated by STM32CubeIDE

```
For more information about which C functions
need which of these lowlevel functions
please consult the newlib libc manual
```

Attention

6.19.2 Function Documentation

6.19.2.1 _sbrk()

```
void * _sbrk (
             ptrdiff_t incr)
```

_sbrk() allocates memory to the newlib heap and is used by malloc and others from the C library

```
* ##########################################################################
* #  .data  #  .bss  #       newlib heap       #          MSP stack          #
* #         #        #                         # Reserved by _Min_Stack_Size #
* ##########################################################################
* ^-- RAM start      ^-- _end                             _estack, RAM end --^
*
```

This implementation starts allocating at the '_end' linker symbol The '_Min_Stack_Size' linker symbol reserves a memory for the MSP stack The implementation considers '_estack' linker symbol to be RAM end NOTE: If the MSP stack, at any point during execution, grows larger than the reserved size, please increase the '_Min_Stack_Size'.

Parameters

incr	Memory size

Returns

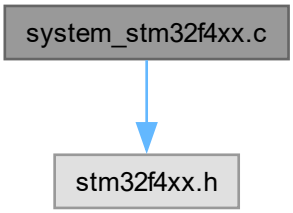
Pointer to allocated memory

6.20 system_stm32f4xx.c File Reference

CMSIS Cortex-M4 Device Peripheral Access Layer System Source File.

```
#include "stm32f4xx.h"
```

Include dependency graph for system_stm32f4xx.c:



Macros

- #define HSE_VALUE ((uint32_t)25000000)
- #define HSI_VALUE ((uint32_t)16000000)

Functions

- void SystemInit (void)

  Setup the microcontroller system Initialize the FPU setting, vector table location and External memory configuration.
- void SystemCoreClockUpdate (void)

  Update SystemCoreClock variable according to Clock Register Values. The SystemCoreClock variable contains the core clock (HCLK), it can be used by the user application to setup the SysTick timer or configure other parameters.

Variables

- uint32_t **SystemCoreClock** = 16000000
- const uint8_t **AHBPrescTable** [16] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6, 7, 8, 9}
- const uint8_t **APBPrescTable** [8] = {0, 0, 0, 0, 1, 2, 3, 4}

6.20.1 Detailed Description

CMSIS Cortex-M4 Device Peripheral Access Layer System Source File.

Author

MCD Application Team

This file provides two functions and one global variable to be called from user application:

- SystemInit(): This function is called at startup just after reset and before branch to main program. This call is made inside the "startup_stm32f4xx.s" file.
- SystemCoreClock variable: Contains the core clock (HCLK), it can be used by the user application to setup the SysTick timer or configure other parameters.
- SystemCoreClockUpdate(): Updates the variable SystemCoreClock and must be called whenever the core clock is changed during program execution.

Attention

Index