RobotROS

Generated by Doxygen 1.9.8

Chapter 1

Robot ROS

1.1 Principal function

function main : init function and start kernel.
function microros_task : Create the publishers and the subscribers and exploit them.
function task_Motor_Left : Control the left motor.
function task_Motor_Right : Control the right motor.
function task_VL53 : Get the VL53 measure and put it in the queue.
function task_Grove_LCD : Get the information from rhe queue and print it on the LCD.
function task_Supervision : The brain's robot decide of the action depending of data receive from microROS.

1.2 Secondary function

function createPublisher : use to create a default publisher.
function createSubscriber : use to create a default subscriber.
function CHECKMRRET : Test if a microRos function success else print errror message.
function SubscriberCallbackFunction : callback call when message is receive.

1.3 Test function

function test_uart2 : Test printf and scanf.
function test_vl53 : Test VL53 sensors.
function test_motor : Test correcteur.

1.4 Config define

1.4.1 Config exo

1.4.2 Config param

Author

titouan melon

Attention

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

Data Structure Documentation

4.1 AMessage Struct Reference

Data Fields

- char command
- int data

4.1.1 Detailed Description

Use to send data to lcd's task

Definition at line 110 of file main.c.

4.1.2 Field Documentation

4.1.2.1 command

```
char command
```

Represent the direction of the robot

Definition at line 112 of file main.c.

4.1.2.2 data

```
int data
```

Represent the mode of the robot

Definition at line 113 of file main.c.

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

- C:/Users/Administrator/Documents/GitHub/ENIB_Robot_Mobile_Ros2/WORKSPACE_F411_uROS6/base↩_robot/Core/Src/main.c

4.2 MicroRosPubMsg Struct Reference

Data Fields

- char dir
- int mode
- int speed

4.2.1 Detailed Description

Use to send information from the task decision to microRos task

Definition at line 119 of file main.c.

4.2.2 Field Documentation

4.2.2.1 dir

```
char dir
```

Represent the direction of the robot

Definition at line 121 of file main.c.

4.2.2.2 mode

```
int mode
```

Represent the mode of the robot

Definition at line 122 of file main.c.

4.2.2.3 speed

```
int speed
```

Represent the speed of the robot

Definition at line 123 of file main.c.

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

- C:/Users/Administrator/Documents/GitHub/ENIB_Robot_Mobile_Ros2/WORKSPACE_F411_uROS6/base↩_robot/Core/Src/main.c

4.3 MicroRosSubMsg Struct Reference

Data Fields

- int dir
- int x
- int y
- int mode
- int speed

4.3.1 Detailed Description

Use to send information get by microRos to decision task

Definition at line 129 of file main.c.

4.3.2 Field Documentation

4.3.2.1 dir

```
int dir
```

Represent the direction send by the IHM

Definition at line 131 of file main.c.

4.3.2.2 mode

```
int mode
```

Represent the mode send by the IHM

Definition at line 134 of file main.c.

4.3.2.3 speed

```
int speed
```

Represent the speed send by the IHM

Definition at line 135 of file main.c.

4.3.2.4 x

```
int x
```

Represent the x position send by the camera

Definition at line 132 of file main.c.

4.3.2.5 y

```
int y
```

Represent the y position send by the camera

Definition at line 133 of file main.c.

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

- C:/Users/Administrator/Documents/GitHub/ENIB_Robot_Mobile_Ros2/WORKSPACE_F411_uROS6/base↩ _robot/Core/Src/main.c

4.4 SequenceStepEnables Struct Reference

Data Fields

- uint8_t tcc
- uint8_t msrc
- uint8_t dss
- uint8_t pre_range
- uint8_t final_range

4.4.1 Detailed Description

Definition at line 291 of file VL53L0X.h.

4.4.2 Field Documentation

4.4.2.1 dss

```
uint8_t dss
```

Definition at line 292 of file VL53L0X.h.

4.4.2.2 final_range

```
uint8_t final_range
```

Definition at line 292 of file VL53L0X.h.

4.4.2.3 msrc

```
uint8_t msrc
```

Definition at line 292 of file VL53L0X.h.

4.4.2.4 pre_range

```
uint8_t pre_range
```

Definition at line 292 of file VL53L0X.h.

4.4.2.5 tcc

```
uint8_t tcc
```

Definition at line 292 of file VL53L0X.h.

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

- C:/Users/Administrator/Documents/GitHub/ENIB_Robot_Mobile_Ros2/WORKSPACE_F411_uROS6/base↩_robot/Core/Inc/VL53L0X.h

4.5 SequenceStepTimeouts Struct Reference

Data Fields

- uint16_t pre_range_vcsel_period_pclks
- uint16_t final_range_vcsel_period_pclks
- uint16_t msrc_dss_tcc_mclks
- uint16_t pre_range_mclks
- uint16_t final_range_mclks
- uint32_t msrc_dss_tcc_us
- uint32_t pre_range_us
- uint32_t final_range_us

4.5.1 Detailed Description

Definition at line 295 of file VL53L0X.h.

4.5.2 Field Documentation

4.5.2.1 final_range_mclks

```
uint16_t final_range_mclks
```

Definition at line 298 of file VL53L0X.h.

4.5.2.2 final_range_us

```
uint32_t final_range_us
```

Definition at line 299 of file VL53L0X.h.

4.5.2.3 final_range_vcsel_period_pclks

```
uint16_t final_range_vcsel_period_pclks
```

Definition at line 296 of file VL53L0X.h.

4.5.2.4 msrc_dss_tcc_mclks

```
uint16_t msrc_dss_tcc_mclks
```

Definition at line 298 of file VL53L0X.h.

4.5.2.5 msrc_dss_tcc_us

```
uint32_t msrc_dss_tcc_us
```

Definition at line 299 of file VL53L0X.h.

4.5.2.6 pre_range_mclks

```
uint16_t pre_range_mclks
```

Definition at line 298 of file VL53L0X.h.

4.5.2.7 pre_range_us

```
uint32_t pre_range_us
```

Definition at line 299 of file VL53L0X.h.

4.5.2.8 pre_range_vcsel_period_pclks

```
uint16_t pre_range_vcsel_period_pclks
```

Definition at line 296 of file VL53L0X.h.

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

- C:/Users/Administrator/Documents/GitHub/ENIB_Robot_Mobile_Ros2/WORKSPACE_F411_uROS6/base↩_robot/Core/Inc/VL53L0X.h

4.6 statInfo_t Struct Reference

Data Fields

- uint16_t rawDistance
- uint16_t signalCnt
- uint16_t ambientCnt
- uint16_t spadCnt
- uint8_t rangeStatus

4.6.1 Detailed Description

Definition at line 193 of file VL53L0X.h.

4.6.2 Field Documentation

4.6.2.1 ambientCnt

```
uint16_t ambientCnt
```

Definition at line 196 of file VL53L0X.h.

4.6.2.2 rangeStatus

```
uint8_t rangeStatus
```

Definition at line 198 of file VL53L0X.h.

4.6.2.3 rawDistance

```
uint16_t rawDistance
```

Definition at line 194 of file VL53L0X.h.

4.6.2.4 signalCnt

```
uint16_t signalCnt
```

Definition at line 195 of file VL53L0X.h.

4.6.2.5 spadCnt

```
uint16_t spadCnt
```

Definition at line 197 of file VL53L0X.h.

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

- C:/Users/Administrator/Documents/GitHub/ENIB_Robot_Mobile_Ros2/WORKSPACE_F411_uROS6/base↩ _robot/Core/Inc/VL53L0X.h

Chapter 5

File Documentation

5.1 captDistIR.h

```
/*
 * IRMeasure.h
 */

#ifndef INC_CAPTDISTIR_H_
#define INC_CAPTDISTIR_H_

#include "main.h"


void captDistIR_Init(void);
int captDistIR_Get(int*);


#endif /* INC_CAPTDISTIR_H_ */
```

5.2 config.h

```
/*
 * config.h
 */

#ifndef INC_CONFIG_H_
#define INC_CONFIG_H_

//=========================================================
// USART : CHOIX DE LA LIAISON SERIE
// USART2 : USART_STLINK (cable)
// USART6 : USART_ZIGBEE (sans fil)
//=========================================================
#define USE_USART_STLINK 1          // A Commenter pour utiliser stlink dans term_printf !! faire un clean
//#define   USE_USART_ZIGBEE 1

#define NB_CAR_TO_RECEIVE       1       // nombre de caractères à recevoir pour déclencher une interruption

#define USART2_BAUDRATE         115200
#define USART6_BAUDRATE         9600
//=========================================================
//                  LIAISON I2C
//=========================================================
#define I2C1_CLOCKSPEED         100000
#define I2C2_CLOCKSPEED         100000

// CAPTEUR I2C DISTANCE ULTRASON SRF02
#define CAPT_US_LEFT_ADDRESS    0xE0
#define CAPT_US_RIGHT_ADDRESS   0xE2

// IMU MPU9250
#define    MPU9250_ADDRESS      0x68
#define    AK8963_ADDRESS       0x0C

// ECRAN LCD
```

```
#define SCREEN_LCD_ADDRESS      (0x30«1)

// SAMPLING PERIOD in ms
#define SAMPLING_PERIOD_ms      5

#endif /* INC_CONFIG_H_ */

/*^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^*/
```

5.3 drv_gpio.h

```
/*
 * drv_gpio.h
 *
 *  Created on: Mar 13, 2023
 *      Author: kerhoas
 */

#ifndef INC_DRV_GPIO_H_
#define INC_DRV_GPIO_H_

void MX_GPIO_Init(void);

#endif /* INC_DRV_GPIO_H_ */
```

5.4 drv_i2c.h

```
/*
 * drv_i2c.h
 *
 *  Created on: Mar 13, 2023
 *      Author: kerhoas
 */

#ifndef INC_DRV_I2C_H_
#define INC_DRV_I2C_H_

void MX_I2C1_Init(void);

// Transmit n_data bytes to i2c slave
int i2c1_WriteBuffer(uint16_t addrSlave, uint8_t *data, int n_data);

// Receive n_data bytes from i2c slave
int i2c1_ReadBuffer(uint16_t addrSlave, uint8_t *data, int n_data);

// Receive n_data bytes - located at regAddr - from i2c slave
int i2c1_ReadRegBuffer(uint16_t addrSlave, uint8_t  regAddr,  uint8_t *data, int n_data);

// Write n_data bytes - have to be written at regAddr - to i2c slave
int i2c1_WriteRegBuffer(uint16_t addrSlave, uint8_t  regAddr,  uint8_t *data, int n_data);

// Write 1 byte at regAddr Slave - Interrupt Method
void i2c1_WriteRegByte_IT(uint16_t addrSlave, uint8_t  regAddr,  uint8_t data);

// Read 1 byte from regAddr Slave - Interrupt Method
void i2c1_ReadRegBuffer_IT(uint16_t addrSlave, uint8_t  regAddr,  uint8_t* datas, int len);

// Write 1 byte to regAddr (16 bits) Slave
int i2c1_WriteReg16Byte(uint16_t addrSlave, uint16_t  regAddr,  uint8_t data);

// Write 16 bits word to regAddr (16 bits) Slave
int i2c1_WriteReg16Word16(uint16_t addrSlave, uint16_t  regAddr,  uint16_t data);

// Write 32 bits word to regAddr (16 bits) Slave
int i2c1_WriteReg16Word32(uint16_t addrSlave, uint16_t  regAddr,  uint32_t data);

// Read 1 byte from regAddr (16 bits) Slave
int i2c1_ReadReg16Byte(uint16_t addrSlave, uint16_t  regAddr, uint8_t *data);

// Read 16 bits word from regAddr (16 bits) Slave
int i2c1_ReadReg16Word16(uint16_t addrSlave, uint16_t  regAddr, uint16_t *data);

// Read 32 bits word from regAddr (16 bits) Slave
int i2c1_ReadReg16Word32(uint16_t addrSlave, uint16_t  regAddr, uint32_t *data);

// Read n_data bytes from regAddr (16 bits) Slave
int i2c1_ReadReg16Buffer(uint16_t addrSlave, uint16_t  regAddr,  uint8_t *data, int n_data);

#endif /* INC_DRV_I2C_H_ */
```

5.5 drv_uart.h

```
/*
 * drv_uart.h
 *
 *  Created on: Mar 13, 2023
 *      Author: kerhoas
 */

#ifndef INC_DRV_UART_H_
#define INC_DRV_UART_H_

void MX_USART1_UART_Init(void);
void MX_USART2_UART_Init(void);
void MX_DMA_Init(void);




#endif /* INC_DRV_UART_H_ */
```

5.6 FreeRTOSConfig.h

```
/* USER CODE BEGIN Header */
/*
 * FreeRTOS Kernel V10.3.1
 * Portion Copyright (C) 2017 Amazon.com, Inc. or its affiliates.  All Rights Reserved.
 * Portion Copyright (C) 2019 StMicroelectronics, Inc.  All Rights Reserved.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 * this software and associated documentation files (the "Software"), to deal in
 * the Software without restriction, including without limitation the rights to
 * use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
 * the Software, and to permit persons to whom the Software is furnished to do so,
 * subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included in all
 * copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
 * FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 * COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
 * IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
 * CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 *
 * http://www.FreeRTOS.org
 * http://aws.amazon.com/freertos
 *
 * 1 tab == 4 spaces!
 */
/* USER CODE END Header */

#ifndef FREERTOS_CONFIG_H
#define FREERTOS_CONFIG_H

/*-----------------------------------------------------------
 * Application specific definitions.
 *
 * These definitions should be adjusted for your particular hardware and
 * application requirements.
 *
 * These parameters and more are described within the ’configuration’ section of the
 * FreeRTOS API documentation available on the FreeRTOS.org web site.
 *
 * See http://www.freertos.org/a00110.html
 *----------------------------------------------------------*/

/* USER CODE BEGIN Includes */
/* Section where include file can be added */
/* USER CODE END Includes */

/* Ensure definitions are only used by the compiler, and not by the assembler. */
#if defined(__ICCARM__) || defined(__CC_ARM) || defined(__GNUC__)
  #include <stdint.h>
  extern uint32_t SystemCoreClock;
#endif
#ifndef CMSIS_device_header
#define CMSIS_device_header "stm32f4xx.h"
#endif /* CMSIS_device_header */

#define configENABLE_FPU                         0
#define configENABLE_MPU                         0
```

```

#define configUSE_PREEMPTION                     1
#define configSUPPORT_STATIC_ALLOCATION          1
#define configSUPPORT_DYNAMIC_ALLOCATION         1
#define configUSE_IDLE_HOOK                      0
#define configUSE_TICK_HOOK                      0
#define configCPU_CLOCK_HZ                       ( SystemCoreClock )
#define configTICK_RATE_HZ                       ((TickType_t)1000)
#define configMAX_PRIORITIES                     ( 56 )
#define configMINIMAL_STACK_SIZE                 ((uint16_t)128)
#define configTOTAL_HEAP_SIZE                    ((size_t)16384)//((size_t)15360)
#define configMAX_TASK_NAME_LEN                  ( 16 )
#define configUSE_TRACE_FACILITY                 1
#define configUSE_16_BIT_TICKS                   0
#define configUSE_MUTEXES                        1
#define configQUEUE_REGISTRY_SIZE                8
#define configUSE_RECURSIVE_MUTEXES              1
#define configUSE_COUNTING_SEMAPHORES            1
#define configUSE_PORT_OPTIMISED_TASK_SELECTION  0
/* USER CODE BEGIN MESSAGE_BUFFER_LENGTH_TYPE */
/* Defaults to size_t for backward compatibility, but can be changed
   if lengths will always be less than the number of bytes in a size_t. */
#define configMESSAGE_BUFFER_LENGTH_TYPE         size_t
/* USER CODE END MESSAGE_BUFFER_LENGTH_TYPE */

/* Co-routine definitions. */
#define configUSE_CO_ROUTINES                    0
#define configMAX_CO_ROUTINE_PRIORITIES          ( 2 )

/* Software timer definitions. */
#define configUSE_TIMERS                         1
#define configTIMER_TASK_PRIORITY                ( 2 )
#define configTIMER_QUEUE_LENGTH                 10
#define configTIMER_TASK_STACK_DEPTH             256

/* The following flag must be enabled only when using newlib */
#define configUSE_NEWLIB_REENTRANT          1

/* CMSIS-RTOS V2 flags */
#define configUSE_OS2_THREAD_SUSPEND_RESUME  1
#define configUSE_OS2_THREAD_ENUMERATE       1
#define configUSE_OS2_EVENTFLAGS_FROM_ISR    1
#define configUSE_OS2_THREAD_FLAGS           1
#define configUSE_OS2_TIMER                  1
#define configUSE_OS2_MUTEX                  1

/* Set the following definitions to 1 to include the API function, or zero
to exclude the API function. */
#define INCLUDE_vTaskPrioritySet             1
#define INCLUDE_uxTaskPriorityGet            1
#define INCLUDE_vTaskDelete                  1
#define INCLUDE_vTaskCleanUpResources        0
#define INCLUDE_vTaskSuspend                 1
#define INCLUDE_vTaskDelayUntil              1
#define INCLUDE_vTaskDelay                   1
#define INCLUDE_xTaskGetSchedulerState       1
#define INCLUDE_xTimerPendFunctionCall       1
#define INCLUDE_xQueueGetMutexHolder         1
#define INCLUDE_uxTaskGetStackHighWaterMark  1
#define INCLUDE_xTaskGetCurrentTaskHandle    1
#define INCLUDE_eTaskGetState                1

/*
 * The CMSIS-RTOS V2 FreeRTOS wrapper is dependent on the heap implementation used
 * by the application thus the correct define need to be enabled below
 */
#define USE_FreeRTOS_HEAP_4

/* Cortex-M specific definitions. */
#ifdef __NVIC_PRIO_BITS
 /* __BVIC_PRIO_BITS will be specified when CMSIS is being used. */
 #define configPRIO_BITS         __NVIC_PRIO_BITS
#else
 #define configPRIO_BITS         4
#endif

/* The lowest interrupt priority that can be used in a call to a "set priority"
function. */
#define configLIBRARY_LOWEST_INTERRUPT_PRIORITY   15

/* The highest interrupt priority that can be used by any interrupt service
routine that makes calls to interrupt safe FreeRTOS API functions.  DO NOT CALL
INTERRUPT SAFE FREERTOS API FUNCTIONS FROM ANY INTERRUPT THAT HAS A HIGHER
PRIORITY THAN THIS! (higher priorities are lower numeric values. */
#define configLIBRARY_MAX_SYSCALL_INTERRUPT_PRIORITY 5

/* Interrupt priorities used by the kernel port layer itself.  These are generic
```

```
to all Cortex-M ports, and do not rely on any particular library functions. */
#define configKERNEL_INTERRUPT_PRIORITY         ( configLIBRARY_LOWEST_INTERRUPT_PRIORITY « (8 - configPRIO_BITS) )
/* !!!! configMAX_SYSCALL_INTERRUPT_PRIORITY must not be set to zero !!!!
See http://www.FreeRTOS.org/RTOS-Cortex-M3-M4.html. */
#define configMAX_SYSCALL_INTERRUPT_PRIORITY    ( configLIBRARY_MAX_SYSCALL_INTERRUPT_PRIORITY « (8 - configPRIO_BITS) )

/* Normal assert() semantics without relying on the provision of an assert.h
header file. */
/* USER CODE BEGIN 1 */
#define configASSERT( x ) if ((x) == 0) {taskDISABLE_INTERRUPTS(); for( ;; );}
/* USER CODE END 1 */

/* Definitions that map the FreeRTOS port interrupt handlers to their CMSIS
standard names. */
#define vPortSVCHandler    SVC_Handler
#define xPortPendSVHandler PendSV_Handler

/* IMPORTANT: After 10.3.1 update, Systick_Handler comes from NVIC (if SYS timebase = systick), otherwise from cmsis_os2.c */

#define USE_CUSTOM_SYSTICK_HANDLER_IMPLEMENTATION 0

/* USER CODE BEGIN Defines */
/* Section where parameter definitions can be added (for instance, to override default ones in FreeRTOS.h) */
/* USER CODE END Defines */

#endif /* FREERTOS_CONFIG_H */
```

5.7 groveLCD.h

```
/*
 * groveLCD.h
 *
 *  Created on: Oct 16, 2019
 *      Author: kerhoas
 */
#ifndef INC_GROVELCD_H_
#define INC_GROVELCD_H_

#include "main.h"

// Device I2C Arress
#define LCD_ADDRESS     (0x7c)
#define RGB_ADDRESS     (0xc4)


// color define
#define WHITE           0
#define RED             1
#define GREEN           2
#define BLUE            3

#define REG_RED         0x04        // pwm2
#define REG_GREEN       0x03        // pwm1
#define REG_BLUE        0x02        // pwm0

#define REG_MODE1       0x00
#define REG_MODE2       0x01
#define REG_OUTPUT      0x08

// commands
#define LCD_CLEARDISPLAY 0x01
#define LCD_RETURNHOME 0x02
#define LCD_ENTRYMODESET 0x04
#define LCD_DISPLAYCONTROL 0x08
#define LCD_CURSORSHIFT 0x10
#define LCD_FUNCTIONSET 0x20
#define LCD_SETCGRAMADDR 0x40
#define LCD_SETDDRAMADDR 0x80

// flags for display entry mode
#define LCD_ENTRYRIGHT 0x00
#define LCD_ENTRYLEFT 0x02
#define LCD_ENTRYSHIFTINCREMENT 0x01
#define LCD_ENTRYSHIFTDECREMENT 0x00

// flags for display on/off control
#define LCD_DISPLAYON 0x04
#define LCD_DISPLAYOFF 0x00
#define LCD_CURSORON 0x02
```

```
#define LCD_CURSOROFF 0x00
#define LCD_BLINKON 0x01
#define LCD_BLINKOFF 0x00

// flags for display/cursor shift
#define LCD_DISPLAYMOVE 0x08
#define LCD_CURSORMOVE 0x00
#define LCD_MOVERIGHT 0x04
#define LCD_MOVELEFT 0x00

// flags for function set
#define LCD_8BITMODE 0x10
#define LCD_4BITMODE 0x00
#define LCD_2LINE 0x08
#define LCD_1LINE 0x00
#define LCD_5x10DOTS 0x04
#define LCD_5x8DOTS 0x00

void groveLCD_test();
void groveLCD_begin(uint8_t cols, uint8_t lines, uint8_t dotsize);
void groveLCD_setColorAll();
void groveLCD_setColorWhite();
void groveLCD_clear();
void groveLCD_home();
void groveLCD_setCursor(uint8_t col, uint8_t row);
void groveLCD_noDisplay();
void groveLCD_display() ;
void groveLCD_noCursor();
void groveLCD_cursor() ;
void groveLCD_noBlink();
void groveLCD_blink();
void groveLCD_scrollDisplayLeft(void);
void groveLCD_scrollDisplayRight(void);
void groveLCD_leftToRight(void);
void groveLCD_rightToLeft(void);
void groveLCD_autoscroll(void);
void groveLCD_noAutoscroll(void);
void groveLCD_createChar(uint8_t location, uint8_t charmap[]);
void groveLCD_blinkLED(void);
void groveLCD_noBlinkLED(void);
void groveLCD_command(uint8_t value);
int groveLCD_write(uint8_t value);
void groveLCD_setReg(unsigned char addr, unsigned char dta);
void groveLCD_setRGB(unsigned char r, unsigned char g, unsigned char b);
void groveLCD_setColor(unsigned char color);
void groveLCD_putString(char* s);
void groveLCD_term_printf(const char* fmt, ...);



#endif /* INC_GROVELCD_H_ */
```

5.8 C:/Users/Administrator/Documents/GitHub/ENIB_Robot_Mobile_Ros2/WORKSPACE_F411_uROS6/base_robot/Core/Inc/main.h File Reference

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

```
#include "stm32f4xx_hal.h"
#include <stdio.h>
#include <rcl/rcl.h>
#include <rcl/error_handling.h>
#include <rclc/rclc.h>
#include <rclc/executor.h>
#include <uxr/client/transport.h>
#include <rmw_microxrcedds_c/config.h>
#include <rmw_microros/rmw_microros.h>
#include <std_msgs/msg/int32.h>
#include <std_msgs/msg/string.h>
#include <std_msgs/msg/header.h>
#include "FreeRTOS.h"
```

```
#include "task.h"
#include "queue.h"
#include "semphr.h"
#include "systemclock.h"
#include "drv_uart.h"
#include "drv_gpio.h"
#include "drv_i2c.h"
#include "cmsis_os.h"
#include "microROS.h"
#include "retarget.h"
```

Macros

- #define B1_Pin GPIO_PIN_13
- #define B1_GPIO_Port GPIOC
- #define USART_TX_Pin GPIO_PIN_2
- #define USART_TX_GPIO_Port GPIOA
- #define USART_RX_Pin GPIO_PIN_3
- #define USART_RX_GPIO_Port GPIOA
- #define LD2_Pin GPIO_PIN_5
- #define LD2_GPIO_Port GPIOA
- #define TMS_Pin GPIO_PIN_13
- #define TMS_GPIO_Port GPIOA
- #define TCK_Pin GPIO_PIN_14
- #define TCK_GPIO_Port GPIOA
- #define SWO_Pin GPIO_PIN_3
- #define SWO_GPIO_Port GPIOB

Functions

- void Error_Handler (void)
- void CHECKMRRET (rcl_ret_t ret, char ∗msg)
- void SubscriberCallbackFunction (const void ∗msgin)
- void microros_task (void ∗argument)
- void task_Motor_Left (void ∗pvParameters)
- void task_Motor_Right (void ∗pvParameters)
- void task_VL53 (void ∗pvParameters)
- void task_Grove_LCD (void ∗pvParameters)
- void task_Supervision (void ∗pvParameters)
- int main (void)

Test function

- void test_uart2 (void ∗pvParameters)
- void test_vl53 (void ∗pvParameters)
- void test_motor (void ∗pvParameters)

5.8.1 Detailed Description

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

Definition in file main.h.

5.8.2 Macro Definition Documentation

5.8.2.1 B1_GPIO_Port

```
#define B1_GPIO_Port GPIOC
```

Definition at line 217 of file main.h.

5.8.2.2 B1_Pin

```
#define B1_Pin GPIO_PIN_13
```

Definition at line 216 of file main.h.

5.8.2.3 LD2_GPIO_Port

```
#define LD2_GPIO_Port GPIOA
```

Definition at line 223 of file main.h.

5.8.2.4 LD2_Pin

```
#define LD2_Pin GPIO_PIN_5
```

Definition at line 222 of file main.h.

5.8.2.5 SWO_GPIO_Port

```
#define SWO_GPIO_Port GPIOB
```

Definition at line 229 of file main.h.

5.8.2.6 SWO_Pin

```
#define SWO_Pin GPIO_PIN_3
```

Definition at line 228 of file main.h.

5.8.2.7 TCK_GPIO_Port

```
#define TCK_GPIO_Port GPIOA
```

Definition at line 227 of file main.h.

5.8.2.8 TCK_Pin

```
#define TCK_Pin GPIO_PIN_14
```

Definition at line 226 of file main.h.

5.8.2.9 TMS_GPIO_Port

```
#define TMS_GPIO_Port GPIOA
```

Definition at line 225 of file main.h.

5.8.2.10 TMS_Pin

```
#define TMS_Pin GPIO_PIN_13
```

Definition at line 224 of file main.h.

5.8.2.11 USART_RX_GPIO_Port

```
#define USART_RX_GPIO_Port GPIOA
```

Definition at line 221 of file main.h.

5.8.2.12 USART_RX_Pin

```
#define USART_RX_Pin GPIO_PIN_3
```

Definition at line 220 of file main.h.

5.8.2.13 USART_TX_GPIO_Port

```
#define USART_TX_GPIO_Port GPIOA
```

Definition at line 219 of file main.h.

5.8.2.14 USART_TX_Pin

```
#define USART_TX_Pin GPIO_PIN_2
```

Definition at line 218 of file main.h.

5.8.3 Function Documentation

5.8.3.1 CHECKMRRET()

```
void CHECKMRRET (
            rcl_ret_t ret,
            char * msg )
```

check if microRos function success else print msg in console

Parameters

ret	return value of microRos function
msg	message to print if fail

Definition at line 153 of file main.c.

5.8.3.2 Error_Handler()

```
void Error_Handler (
             void  )
```

Definition at line 941 of file main.c.

5.8.3.3 main()

```
int main (
             void  )
```

Init all GPIO and drivers, start the task, init semaphore and queue and launch the kernel

- Config EXSTARTUP
  - Launch microRos, supervision, left motor, right motor and lcd task
- Config EXTEST_UART2
  - Launch test_uart2 task
- Config EXCORRECTOR
  - Launch test_motor task
- Config EXTESTCORRECTOR
  - Launch supervision, left motor and right motor task
- Config EXTEST_VL53
  - Launch test_vl53 task
- Config EXTEST_MICROROS
  - Launch microRos task
- Config EXFINAL
  - Launch microRos, supervision, left motor, right motor, vl53 and lcd task

Definition at line 775 of file main.c.

5.8.3.4 microros_task()

```
void microros_task (
             void * argument )
```

- All config
  - Create the node *STM32_node*
  - Set the Domain id of microRos
- Config EXSTARTUP :
  - Create a publisher and send a message on it
- Config EXTEST_MICROROS :
  - Create a publisher, a subscriber and an executor
  - Init the executor and add the subscriber to it
  - Run the executor and send the receive message on the publisher
- Config EXFINAL :
  - Create 3 publishers, 5 subscriber and an executor
  - Init the executor and add the 5 subscribers to it
  - run the executor and if they are no elements waiting to be read by the task decision put the receive information in the queue If decison task send data then publish data to microRos

Parameters

argument	

Definition at line 166 of file main.c.

5.8.3.5 SubscriberCallbackFunction()

```
void SubscriberCallbackFunction (
             const void * msgin )
```

callback call by microros when a message is receive here use as debug and just print the receive msg

Parameters

message	receive

Definition at line 155 of file main.c.

5.8.3.6 task_Grove_LCD()

```
void task_Grove_LCD (
             void * pvParameters )
```

Task use to write information on LCD depending of the data in the LCD queue

- Config EXSTARTUP :
  - Print 'TEST' LCD on screen
- Config EXFINAL :
  - Print different messages depending of the actual mode

Parameters

argument	

5.8.3.7 task_Motor_Left()

```
void task_Motor_Left (
            void * pvParameters )
```

Task use to control the left motor of the robot

Parameters

argument	

Definition at line 365 of file main.c.

5.8.3.8 task_Motor_Right()

```
void task_Motor_Right (
            void * pvParameters )
```

Task use to control the right motor of the robot

Parameters

argument	

Definition at line 391 of file main.c.

5.8.3.9 task_Supervision()

```
void task_Supervision (
            void * pvParameters )
```

Brain of the robot. get information for MicroRos and VL53 task, then send speed to left and right motor, lcd and microRos task

- Config EXSTARTUP :
  - Make the robot drive forward until an obstacle are found

- Config EXTESTCORRECTOR :
    - Make the robot drive forward at speed set by config
- Config EXFINAL :
    - Make robot switch beetween 3 behaviour depending of the mode
    - Obstacle : drive and avoid obstacles
    - Manual : drive in direction set in ihm
    - Camera : follow an object

Parameters

argument	

Definition at line 476 of file main.c.

5.8.3.10 task_VL53()

```
void task_VL53 (
            void * pvParameters )
```

task that get the value of the VL53 sensor and put it on the VL53 queue

Parameters

argument	

5.8.3.11 test_motor()

```
void test_motor (
            void * pvParameters )
```

Use to set the duty cycle and register the motor speed at each Te

Definition at line 895 of file main.c.

5.8.3.12 test_uart2()

```
void test_uart2 (
            void * pvParameters )
```

Use to test printf and scanf function

Definition at line 872 of file main.c.

5.8.3.13 test_vl53()

```
void test_vl53 (
              void * pvParameters )
```

Use to test the VL53 sensor

Definition at line 884 of file main.c.

5.9 main.h

Go to the documentation of this file.

```
/* USER CODE BEGIN Header */
/* USER CODE END Header */

/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __MAIN_H
#define __MAIN_H

#ifdef __cplusplus
extern "C" {
#endif

/* Includes ------------------------------------------------------------------*/
#include "stm32f4xx_hal.h"

#include <stdio.h>
#include <rcl/rcl.h>
#include <rcl/error_handling.h>
#include <rclc/rclc.h>
#include <rclc/executor.h>
#include <uxr/client/transport.h>
#include <rmw_microxrcedds_c/config.h>
#include <rmw_microros/rmw_microros.h>

#include <std_msgs/msg/int32.h>
#include <std_msgs/msg/string.h>
#include <std_msgs/msg/header.h>

#include "FreeRTOS.h"
#include "task.h"
#include "queue.h"
#include "semphr.h"

#include "systemclock.h"
#include "drv_uart.h"
#include "drv_gpio.h"
#include "drv_i2c.h"
#include "cmsis_os.h"

#include "microROS.h" //Custom microRos utils
#include "retarget.h" //To redirect printf and scanf on UART2

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* Exported types ------------------------------------------------------------*/
/* USER CODE BEGIN ET */

/* USER CODE END ET */

/* Exported constants --------------------------------------------------------*/
/* USER CODE BEGIN EC */

/* USER CODE END EC */

/* Exported macro ------------------------------------------------------------*/
/* USER CODE BEGIN EM */

/* USER CODE END EM */

/* Exported functions prototypes ---------------------------------------------*/
void Error_Handler(void);

/* USER CODE BEGIN EFP */
void CHECKMRRET(rcl_ret_t ret, char* msg);

void SubscriberCallbackFunction(const void *msgin);

void microros_task(void *argument);

void task_Motor_Left(void *pvParameters);

void task_Motor_Right(void *pvParameters);

void task_VL53(void *pvParameters);

void task_Grove_LCD(void *pvParameters);

void task_Supervision(void *pvParameters);

void test_uart2(void *pvParameters);
void test_vl53(void *pvParameters);
void test_motor(void *pvParameters);
int main(void);

/* USER CODE END EFP */

/* Private defines -----------------------------------------------------------*/
#define B1_Pin GPIO_PIN_13
#define B1_GPIO_Port GPIOC
#define USART_TX_Pin GPIO_PIN_2
#define USART_TX_GPIO_Port GPIOA
#define USART_RX_Pin GPIO_PIN_3
#define USART_RX_GPIO_Port GPIOA
#define LD2_Pin GPIO_PIN_5
#define LD2_GPIO_Port GPIOA
#define TMS_Pin GPIO_PIN_13
#define TMS_GPIO_Port GPIOA
#define TCK_Pin GPIO_PIN_14
#define TCK_GPIO_Port GPIOA
#define SWO_Pin GPIO_PIN_3
#define SWO_GPIO_Port GPIOB
/* USER CODE BEGIN Private defines */

/* USER CODE END Private defines */

#ifdef __cplusplus
}
#endif

#endif /* __MAIN_H */
```

5.10 C:/Users/Administrator/Documents/GitHub/ENIB_Robot_Mobile_↩ Ros2/WORKSPACE_F411_uROS6/base_robot/Core/Inc/microROS.h File Reference

: Contain microROS topic and default creator for subscriber and publisher

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

Macros

- #define ARRAY_LEN 100
- #define CAPTEUR_DIR_TOPIC "capteur/dir"
- #define ETAT_MODE_TOPIC "etat/mode"
- #define ETAT_SPEED_TOPIC "etat/speed"
- #define CAMERA_X_TOPIC "camera/X"
- #define CAMERA_Y_TOPIC "camera/Y"
- #define TELECOMMANDE_DIR_TOPIC "direction"
- #define CONFIG_MODE_TOPIC "mode"
- #define CONFIG_SPEED_TOPIC "speed"

Functions

- void createPublisher (rcl_publisher_t ∗publisher, const rcl_node_t ∗node, const rosidl_message_type_↩ support_t ∗type_support, const char ∗topic_name, std_msgs__msg__Int32 ∗msg)
- void createSubscriber (rcl_subscription_t ∗subscription, rcl_node_t ∗node, const rosidl_message_type_↩ support_t ∗type_support, const char ∗topic_name, std_msgs__msg__Int32 ∗msg)

5.10.1 Detailed Description

: Contain microROS topic and default creator for subscriber and publisher

Definition in file microROS.h.

5.10.2 Macro Definition Documentation

5.10.2.1 ARRAY_LEN

```
#define ARRAY_LEN 100
```

Length of string messages

Definition at line 10 of file microROS.h.

5.10.2.2 CAMERA_X_TOPIC

```
#define CAMERA_X_TOPIC "camera/X"
```

Topic name of x camera subscriber

Definition at line 14 of file microROS.h.

5.10.2.3 CAMERA_Y_TOPIC

```
#define CAMERA_Y_TOPIC "camera/Y"
```

Topic name of y camera subscriber

Definition at line 15 of file microROS.h.

5.10.2.4 CAPTEUR_DIR_TOPIC

```
#define CAPTEUR_DIR_TOPIC "capteur/dir"
```

Topic name of direction publisher

Definition at line 11 of file microROS.h.

5.10.2.5 CONFIG_MODE_TOPIC

```
#define CONFIG_MODE_TOPIC "mode"
```

Definition at line 17 of file microROS.h.

5.10.2.6 CONFIG_SPEED_TOPIC

```
#define CONFIG_SPEED_TOPIC "speed"
```

Definition at line 18 of file microROS.h.

5.10.2.7 ETAT_MODE_TOPIC

```
#define ETAT_MODE_TOPIC "etat/mode"
```

Topic name of mode publisher

Definition at line 12 of file microROS.h.

5.10.2.8 ETAT_SPEED_TOPIC

```
#define ETAT_SPEED_TOPIC "etat/speed"
```

Topic name of speed publisher

Definition at line 13 of file microROS.h.

5.10.2.9 TELECOMMANDE_DIR_TOPIC

```
#define TELECOMMANDE_DIR_TOPIC "direction"
```

Definition at line 16 of file microROS.h.

5.10.3 Function Documentation

5.10.3.1 createPublisher()

```
void createPublisher (
            rcl_publisher_t * publisher,
            const rcl_node_t * node,
            const rosidl_message_type_support_t * type_support,
            const char * topic_name,
            std_msgs__msg__Int32 * msg )
```

Create a publisher with default options

Parameters

publisher	microRos structure that represent a publisher
node	microRos structure that represent a node
type_support	microRos structure that represent the type of message
topic_name	The name of the topic
msg	microRos structure that represent the message

Definition at line 5 of file microROS.c.

5.10.3.2 createSubscriber()

```
void createSubscriber (
             rcl_subscription_t * subscription,
             rcl_node_t * node,
             const rosidl_message_type_support_t * type_support,
             const char * topic_name,
             std_msgs__msg__Int32 * msg )
```

Create a subscriber with default options

Parameters

subscription	microRos structure that represent a subscriber
node	microRos structure that represent a node
type_support	microRos structure that represent the type of message
topic_name	The name of the topic
msg	microRos structure that represent the message

Definition at line 24 of file microROS.c.

5.11 microROS.h

Go to the documentation of this file.

```

#ifndef DEF_MICROROS
#define DEF_MICROROS

    #include "main.h"
    #define ARRAY_LEN 100
    #define CAPTEUR_DIR_TOPIC "capteur/dir"
    #define ETAT_MODE_TOPIC "etat/mode"
    #define ETAT_SPEED_TOPIC "etat/speed"
    #define CAMERA_X_TOPIC "camera/X"
    #define CAMERA_Y_TOPIC "camera/Y"
    #define TELECOMMANDE_DIR_TOPIC "direction"  //"telecommande/dir"
    #define CONFIG_MODE_TOPIC "mode"  //"config/mode"
    #define CONFIG_SPEED_TOPIC "speed"  //"config/speed"

    void createPublisher(rcl_publisher_t* publisher,
              const rcl_node_t* node,
              const rosidl_message_type_support_t* type_support,
              const char* topic_name,
              std_msgs__msg__Int32* msg);

    void createSubscriber(rcl_subscription_t* subscription,
              rcl_node_t* node,
              const rosidl_message_type_support_t* type_support,
              const char* topic_name,
              std_msgs__msg__Int32* msg);

#endif //DEF_MICROROS
```

5.12 motorCommand.h

```
/*
 * MotorCommand.h
 */

#ifndef INC_MOTORCOMMAND_H_
#define INC_MOTORCOMMAND_H_

#include "main.h"


void motorCommand_Init(void);
void motorLeft_SetDuty(int);
void motorRight_SetDuty(int);



#endif /* INC_MOTORCOMMAND_H_ */
```

5.13 quadEncoder.h

```
/*
 * QuadEncoder.h
 */

#ifndef INC_QUADENCODER_H_
#define INC_QUADENCODER_H_

#include "main.h"

void quadEncoder_Init(void);
int16_t quadEncoder_GetPos16L(void);
int16_t quadEncoder_GetPos16R(void);
int32_t quadEncoder_GetPos32L(void);
int32_t quadEncoder_GetPos32R(void);
int16_t quadEncoder_GetSpeedL(void);
int16_t quadEncoder_GetSpeedR(void);
void quadEncoder_CallbackIndexL(void);
void quadEncoder_CallbackIndexR(void);
void quadEncoder_PosCalcL(int*);
void quadEncoder_PosCalcR(int*);

#endif /* INC_QUADENCODER_H_ */
```

5.14 retarget.h

```
/*
 * retarget.h
 *
 *  Created on: Oct 10, 2023
 *      Author: rospc
 */

#ifndef INC_RETARGET_H_
#define INC_RETARGET_H_

// All credit to Carmine Noviello for this code
// https://github.com/cnoviello/mastering-stm32/blob/master/nucleo-f030R8/system/include/retarget/retarget.h

#ifndef _RETARGET_H__
#define _RETARGET_H__

#include "stm32f4xx_hal.h"
#include <sys/stat.h>

void RetargetInit(UART_HandleTypeDef *huart);
int _isatty(int fd);
int _write(int fd, char* ptr, int len);
int _close(int fd);
int _lseek(int fd, int ptr, int dir);
int _read(int fd, char* ptr, int len);
int _fstat(int fd, struct stat* st);
int _getpid(void);
int _kill(int pid, int sig);

#endif //#ifndef _RETARGET_H__

#endif /* INC_RETARGET_H_ */
```

5.15 stm32f4xx_hal_conf.h

```
/* USER CODE BEGIN Header */
/* USER CODE END Header */

/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __STM32F4xx_HAL_CONF_H
#define __STM32F4xx_HAL_CONF_H

#ifdef __cplusplus
 extern "C" {
#endif

/* Exported types ------------------------------------------------------------*/
/* Exported constants --------------------------------------------------------*/

/* ########################## Module Selection ############################## */
#define HAL_MODULE_ENABLED

 #define HAL_ADC_MODULE_ENABLED
/* #define HAL_CRYP_MODULE_ENABLED   */
/* #define HAL_CAN_MODULE_ENABLED   */
/* #define HAL_CRC_MODULE_ENABLED   */
/* #define HAL_CAN_LEGACY_MODULE_ENABLED   */
/* #define HAL_CRYP_MODULE_ENABLED   */
/* #define HAL_DAC_MODULE_ENABLED   */
/* #define HAL_DCMI_MODULE_ENABLED   */
/* #define HAL_DMA2D_MODULE_ENABLED   */
/* #define HAL_ETH_MODULE_ENABLED   */
/* #define HAL_NAND_MODULE_ENABLED   */
/* #define HAL_NOR_MODULE_ENABLED   */
/* #define HAL_PCCARD_MODULE_ENABLED   */
/* #define HAL_SRAM_MODULE_ENABLED   */
/* #define HAL_SDRAM_MODULE_ENABLED   */
/* #define HAL_HASH_MODULE_ENABLED   */
#define HAL_I2C_MODULE_ENABLED
/* #define HAL_I2S_MODULE_ENABLED   */
/* #define HAL_IWDG_MODULE_ENABLED   */
/* #define HAL_LTDC_MODULE_ENABLED   */
/* #define HAL_RNG_MODULE_ENABLED   */
/* #define HAL_RTC_MODULE_ENABLED   */
/* #define HAL_SAI_MODULE_ENABLED   */
/* #define HAL_SD_MODULE_ENABLED   */
/* #define HAL_MMC_MODULE_ENABLED   */
/* #define HAL_SPI_MODULE_ENABLED   */
#define HAL_TIM_MODULE_ENABLED
#define HAL_UART_MODULE_ENABLED
/* #define HAL_USART_MODULE_ENABLED   */
/* #define HAL_IRDA_MODULE_ENABLED   */
/* #define HAL_SMARTCARD_MODULE_ENABLED   */
/* #define HAL_SMBUS_MODULE_ENABLED   */
/* #define HAL_WWDG_MODULE_ENABLED   */
/* #define HAL_PCD_MODULE_ENABLED   */
/* #define HAL_HCD_MODULE_ENABLED   */
/* #define HAL_DSI_MODULE_ENABLED   */
/* #define HAL_QSPI_MODULE_ENABLED   */
/* #define HAL_QSPI_MODULE_ENABLED   */
/* #define HAL_CEC_MODULE_ENABLED   */
/* #define HAL_FMPI2C_MODULE_ENABLED   */
/* #define HAL_FMPSMBUS_MODULE_ENABLED   */
/* #define HAL_SPDIFRX_MODULE_ENABLED   */
/* #define HAL_DFSDM_MODULE_ENABLED   */
/* #define HAL_LPTIM_MODULE_ENABLED   */
#define HAL_GPIO_MODULE_ENABLED
#define HAL_EXTI_MODULE_ENABLED
#define HAL_DMA_MODULE_ENABLED
#define HAL_RCC_MODULE_ENABLED
#define HAL_FLASH_MODULE_ENABLED
#define HAL_PWR_MODULE_ENABLED
#define HAL_CORTEX_MODULE_ENABLED

```

```
/* ########################## HSE/HSI Values adaptation ##################### */
#if !defined   (HSE_VALUE)
  #define HSE_VALUE    8000000U
#endif /* HSE_VALUE */

#if !defined   (HSE_STARTUP_TIMEOUT)
  #define HSE_STARTUP_TIMEOUT    100U
#endif /* HSE_STARTUP_TIMEOUT */

#if !defined   (HSI_VALUE)
  #define HSI_VALUE    ((uint32_t)16000000U)
#endif /* HSI_VALUE */

#if !defined   (LSI_VALUE)
 #define LSI_VALUE  32000U
#endif /* LSI_VALUE */
#if !defined   (LSE_VALUE)
 #define LSE_VALUE  32768U
#endif /* LSE_VALUE */

#if !defined   (LSE_STARTUP_TIMEOUT)
  #define LSE_STARTUP_TIMEOUT    5000U
#endif /* LSE_STARTUP_TIMEOUT */

#if !defined   (EXTERNAL_CLOCK_VALUE)
  #define EXTERNAL_CLOCK_VALUE    12288000U
#endif /* EXTERNAL_CLOCK_VALUE */

/* Tip: To avoid modifying this file each time you need to use different HSE,
   ===  you can define the HSE value in your toolchain compiler preprocessor. */

/* ########################### System Configuration ######################### */
#define  VDD_VALUE            3300U
#define  TICK_INT_PRIORITY            15U
#define  USE_RTOS                     0U
#define  PREFETCH_ENABLE              1U
#define  INSTRUCTION_CACHE_ENABLE     1U
#define  DATA_CACHE_ENABLE            1U

#define  USE_HAL_ADC_REGISTER_CALLBACKS         0U /* ADC register callback disabled       */
#define  USE_HAL_CAN_REGISTER_CALLBACKS         0U /* CAN register callback disabled       */
#define  USE_HAL_CEC_REGISTER_CALLBACKS         0U /* CEC register callback disabled       */
#define  USE_HAL_CRYP_REGISTER_CALLBACKS        0U /* CRYP register callback disabled      */
#define  USE_HAL_DAC_REGISTER_CALLBACKS         0U /* DAC register callback disabled       */
#define  USE_HAL_DCMI_REGISTER_CALLBACKS        0U /* DCMI register callback disabled      */
#define  USE_HAL_DFSDM_REGISTER_CALLBACKS       0U /* DFSDM register callback disabled     */
#define  USE_HAL_DMA2D_REGISTER_CALLBACKS       0U /* DMA2D register callback disabled     */
#define  USE_HAL_DSI_REGISTER_CALLBACKS         0U /* DSI register callback disabled       */
#define  USE_HAL_ETH_REGISTER_CALLBACKS         0U /* ETH register callback disabled       */
#define  USE_HAL_HASH_REGISTER_CALLBACKS        0U /* HASH register callback disabled      */
#define  USE_HAL_HCD_REGISTER_CALLBACKS         0U /* HCD register callback disabled       */
#define  USE_HAL_I2C_REGISTER_CALLBACKS         0U /* I2C register callback disabled       */
#define  USE_HAL_FMPI2C_REGISTER_CALLBACKS      0U /* FMPI2C register callback disabled    */
#define  USE_HAL_FMPSMBUS_REGISTER_CALLBACKS    0U /* FMPSMBUS register callback disabled  */
#define  USE_HAL_I2S_REGISTER_CALLBACKS         0U /* I2S register callback disabled       */
#define  USE_HAL_IRDA_REGISTER_CALLBACKS        0U /* IRDA register callback disabled      */
#define  USE_HAL_LPTIM_REGISTER_CALLBACKS       0U /* LPTIM register callback disabled     */
#define  USE_HAL_LTDC_REGISTER_CALLBACKS        0U /* LTDC register callback disabled      */
#define  USE_HAL_MMC_REGISTER_CALLBACKS         0U /* MMC register callback disabled       */
#define  USE_HAL_NAND_REGISTER_CALLBACKS        0U /* NAND register callback disabled      */
#define  USE_HAL_NOR_REGISTER_CALLBACKS         0U /* NOR register callback disabled       */
#define  USE_HAL_PCCARD_REGISTER_CALLBACKS      0U /* PCCARD register callback disabled    */
#define  USE_HAL_PCD_REGISTER_CALLBACKS         0U /* PCD register callback disabled       */
#define  USE_HAL_QSPI_REGISTER_CALLBACKS        0U /* QSPI register callback disabled      */
#define  USE_HAL_RNG_REGISTER_CALLBACKS         0U /* RNG register callback disabled       */
#define  USE_HAL_RTC_REGISTER_CALLBACKS         0U /* RTC register callback disabled       */
#define  USE_HAL_SAI_REGISTER_CALLBACKS         0U /* SAI register callback disabled       */
#define  USE_HAL_SD_REGISTER_CALLBACKS          0U /* SD register callback disabled        */
#define  USE_HAL_SMARTCARD_REGISTER_CALLBACKS   0U /* SMARTCARD register callback disabled */
#define  USE_HAL_SDRAM_REGISTER_CALLBACKS       0U /* SDRAM register callback disabled     */
#define  USE_HAL_SRAM_REGISTER_CALLBACKS        0U /* SRAM register callback disabled      */
#define  USE_HAL_SPDIFRX_REGISTER_CALLBACKS     0U /* SPDIFRX register callback disabled   */
#define  USE_HAL_SMBUS_REGISTER_CALLBACKS       0U /* SMBUS register callback disabled     */
#define  USE_HAL_SPI_REGISTER_CALLBACKS         0U /* SPI register callback disabled       */
#define  USE_HAL_TIM_REGISTER_CALLBACKS         0U /* TIM register callback disabled       */
#define  USE_HAL_UART_REGISTER_CALLBACKS        0U /* UART register callback disabled      */
#define  USE_HAL_USART_REGISTER_CALLBACKS       0U /* USART register callback disabled     */
#define  USE_HAL_WWDG_REGISTER_CALLBACKS        0U /* WWDG register callback disabled      */

/* ########################## Assert Selection ############################## */
/* #define USE_FULL_ASSERT    1U */

/* ################## Ethernet peripheral configuration ##################### */

/* Section 1 : Ethernet peripheral configuration */

/* MAC ADDRESS: MAC_ADDR0:MAC_ADDR1:MAC_ADDR2:MAC_ADDR3:MAC_ADDR4:MAC_ADDR5 */
```

```
#define MAC_ADDR0   2U
#define MAC_ADDR1   0U
#define MAC_ADDR2   0U
#define MAC_ADDR3   0U
#define MAC_ADDR4   0U
#define MAC_ADDR5   0U

/* Definition of the Ethernet driver buffers size and count */
#define ETH_RX_BUF_SIZE                ETH_MAX_PACKET_SIZE /* buffer size for receive               */
#define ETH_TX_BUF_SIZE                ETH_MAX_PACKET_SIZE /* buffer size for transmit              */
#define ETH_RXBUFNB                    4U       /* 4 Rx buffers of size ETH_RX_BUF_SIZE  */
#define ETH_TXBUFNB                    4U       /* 4 Tx buffers of size ETH_TX_BUF_SIZE  */

/* Section 2: PHY configuration section */

/* DP83848_PHY_ADDRESS Address*/
#define DP83848_PHY_ADDRESS           0x01U
/* PHY Reset delay these values are based on a 1 ms Systick interrupt*/
#define PHY_RESET_DELAY                 0x000000FFU
/* PHY Configuration delay */
#define PHY_CONFIG_DELAY                0x00000FFFU

#define PHY_READ_TO                     0x0000FFFFU
#define PHY_WRITE_TO                    0x0000FFFFU

/* Section 3: Common PHY Registers */

#define PHY_BCR                         ((uint16_t)0x0000U)
#define PHY_BSR                         ((uint16_t)0x0001U)
#define PHY_RESET                       ((uint16_t)0x8000U)
#define PHY_LOOPBACK                    ((uint16_t)0x4000U)
#define PHY_FULLDUPLEX_100M             ((uint16_t)0x2100U)
#define PHY_HALFDUPLEX_100M             ((uint16_t)0x2000U)
#define PHY_FULLDUPLEX_10M              ((uint16_t)0x0100U)
#define PHY_HALFDUPLEX_10M              ((uint16_t)0x0000U)
#define PHY_AUTONEGOTIATION             ((uint16_t)0x1000U)
#define PHY_RESTART_AUTONEGOTIATION     ((uint16_t)0x0200U)
#define PHY_POWERDOWN                   ((uint16_t)0x0800U)
#define PHY_ISOLATE                     ((uint16_t)0x0400U)
#define PHY_AUTONEGO_COMPLETE           ((uint16_t)0x0020U)
#define PHY_LINKED_STATUS               ((uint16_t)0x0004U)
#define PHY_JABBER_DETECTION            ((uint16_t)0x0002U)
/* Section 4: Extended PHY Registers */
#define PHY_SR                          ((uint16_t)0x10U)
#define PHY_SPEED_STATUS                ((uint16_t)0x0002U)
#define PHY_DUPLEX_STATUS               ((uint16_t)0x0004U)
/* ################## SPI peripheral configuration ########################## */

/* CRC FEATURE: Use to activate CRC feature inside HAL SPI Driver
* Activated: CRC code is present inside driver
* Deactivated: CRC code cleaned from driver
*/

#define USE_SPI_CRC                     0U

/* Includes ------------------------------------------------------------------*/
#ifdef HAL_RCC_MODULE_ENABLED
  #include "stm32f4xx_hal_rcc.h"
#endif /* HAL_RCC_MODULE_ENABLED */

#ifdef HAL_GPIO_MODULE_ENABLED
  #include "stm32f4xx_hal_gpio.h"
#endif /* HAL_GPIO_MODULE_ENABLED */

#ifdef HAL_EXTI_MODULE_ENABLED
  #include "stm32f4xx_hal_exti.h"
#endif /* HAL_EXTI_MODULE_ENABLED */

#ifdef HAL_DMA_MODULE_ENABLED
  #include "stm32f4xx_hal_dma.h"
#endif /* HAL_DMA_MODULE_ENABLED */

#ifdef HAL_CORTEX_MODULE_ENABLED
  #include "stm32f4xx_hal_cortex.h"
#endif /* HAL_CORTEX_MODULE_ENABLED */

#ifdef HAL_ADC_MODULE_ENABLED
  #include "stm32f4xx_hal_adc.h"
#endif /* HAL_ADC_MODULE_ENABLED */

#ifdef HAL_CAN_MODULE_ENABLED
  #include "stm32f4xx_hal_can.h"
#endif /* HAL_CAN_MODULE_ENABLED */

#ifdef HAL_CAN_LEGACY_MODULE_ENABLED
  #include "stm32f4xx_hal_can_legacy.h"
#endif /* HAL_CAN_LEGACY_MODULE_ENABLED */
```

```

#ifdef HAL_CRC_MODULE_ENABLED
  #include "stm32f4xx_hal_crc.h"
#endif /* HAL_CRC_MODULE_ENABLED */

#ifdef HAL_CRYP_MODULE_ENABLED
  #include "stm32f4xx_hal_cryp.h"
#endif /* HAL_CRYP_MODULE_ENABLED */

#ifdef HAL_DMA2D_MODULE_ENABLED
  #include "stm32f4xx_hal_dma2d.h"
#endif /* HAL_DMA2D_MODULE_ENABLED */

#ifdef HAL_DAC_MODULE_ENABLED
  #include "stm32f4xx_hal_dac.h"
#endif /* HAL_DAC_MODULE_ENABLED */

#ifdef HAL_DCMI_MODULE_ENABLED
  #include "stm32f4xx_hal_dcmi.h"
#endif /* HAL_DCMI_MODULE_ENABLED */

#ifdef HAL_ETH_MODULE_ENABLED
  #include "stm32f4xx_hal_eth.h"
#endif /* HAL_ETH_MODULE_ENABLED */

#ifdef HAL_FLASH_MODULE_ENABLED
  #include "stm32f4xx_hal_flash.h"
#endif /* HAL_FLASH_MODULE_ENABLED */

#ifdef HAL_SRAM_MODULE_ENABLED
  #include "stm32f4xx_hal_sram.h"
#endif /* HAL_SRAM_MODULE_ENABLED */

#ifdef HAL_NOR_MODULE_ENABLED
  #include "stm32f4xx_hal_nor.h"
#endif /* HAL_NOR_MODULE_ENABLED */

#ifdef HAL_NAND_MODULE_ENABLED
  #include "stm32f4xx_hal_nand.h"
#endif /* HAL_NAND_MODULE_ENABLED */

#ifdef HAL_PCCARD_MODULE_ENABLED
  #include "stm32f4xx_hal_pccard.h"
#endif /* HAL_PCCARD_MODULE_ENABLED */

#ifdef HAL_SDRAM_MODULE_ENABLED
  #include "stm32f4xx_hal_sdram.h"
#endif /* HAL_SDRAM_MODULE_ENABLED */

#ifdef HAL_HASH_MODULE_ENABLED
 #include "stm32f4xx_hal_hash.h"
#endif /* HAL_HASH_MODULE_ENABLED */

#ifdef HAL_I2C_MODULE_ENABLED
 #include "stm32f4xx_hal_i2c.h"
#endif /* HAL_I2C_MODULE_ENABLED */

#ifdef HAL_SMBUS_MODULE_ENABLED
 #include "stm32f4xx_hal_smbus.h"
#endif /* HAL_SMBUS_MODULE_ENABLED */

#ifdef HAL_I2S_MODULE_ENABLED
 #include "stm32f4xx_hal_i2s.h"
#endif /* HAL_I2S_MODULE_ENABLED */

#ifdef HAL_IWDG_MODULE_ENABLED
 #include "stm32f4xx_hal_iwdg.h"
#endif /* HAL_IWDG_MODULE_ENABLED */

#ifdef HAL_LTDC_MODULE_ENABLED
 #include "stm32f4xx_hal_ltdc.h"
#endif /* HAL_LTDC_MODULE_ENABLED */

#ifdef HAL_PWR_MODULE_ENABLED
 #include "stm32f4xx_hal_pwr.h"
#endif /* HAL_PWR_MODULE_ENABLED */

#ifdef HAL_RNG_MODULE_ENABLED
 #include "stm32f4xx_hal_rng.h"
#endif /* HAL_RNG_MODULE_ENABLED */

#ifdef HAL_RTC_MODULE_ENABLED
 #include "stm32f4xx_hal_rtc.h"
#endif /* HAL_RTC_MODULE_ENABLED */

#ifdef HAL_SAI_MODULE_ENABLED
 #include "stm32f4xx_hal_sai.h"
```

```
#endif /* HAL_SAI_MODULE_ENABLED */

#ifdef HAL_SD_MODULE_ENABLED
 #include "stm32f4xx_hal_sd.h"
#endif /* HAL_SD_MODULE_ENABLED */

#ifdef HAL_SPI_MODULE_ENABLED
 #include "stm32f4xx_hal_spi.h"
#endif /* HAL_SPI_MODULE_ENABLED */

#ifdef HAL_TIM_MODULE_ENABLED
 #include "stm32f4xx_hal_tim.h"
#endif /* HAL_TIM_MODULE_ENABLED */

#ifdef HAL_UART_MODULE_ENABLED
 #include "stm32f4xx_hal_uart.h"
#endif /* HAL_UART_MODULE_ENABLED */

#ifdef HAL_USART_MODULE_ENABLED
 #include "stm32f4xx_hal_usart.h"
#endif /* HAL_USART_MODULE_ENABLED */

#ifdef HAL_IRDA_MODULE_ENABLED
 #include "stm32f4xx_hal_irda.h"
#endif /* HAL_IRDA_MODULE_ENABLED */

#ifdef HAL_SMARTCARD_MODULE_ENABLED
 #include "stm32f4xx_hal_smartcard.h"
#endif /* HAL_SMARTCARD_MODULE_ENABLED */

#ifdef HAL_WWDG_MODULE_ENABLED
 #include "stm32f4xx_hal_wwdg.h"
#endif /* HAL_WWDG_MODULE_ENABLED */

#ifdef HAL_PCD_MODULE_ENABLED
 #include "stm32f4xx_hal_pcd.h"
#endif /* HAL_PCD_MODULE_ENABLED */

#ifdef HAL_HCD_MODULE_ENABLED
 #include "stm32f4xx_hal_hcd.h"
#endif /* HAL_HCD_MODULE_ENABLED */

#ifdef HAL_DSI_MODULE_ENABLED
 #include "stm32f4xx_hal_dsi.h"
#endif /* HAL_DSI_MODULE_ENABLED */

#ifdef HAL_QSPI_MODULE_ENABLED
 #include "stm32f4xx_hal_qspi.h"
#endif /* HAL_QSPI_MODULE_ENABLED */

#ifdef HAL_CEC_MODULE_ENABLED
 #include "stm32f4xx_hal_cec.h"
#endif /* HAL_CEC_MODULE_ENABLED */

#ifdef HAL_FMPI2C_MODULE_ENABLED
 #include "stm32f4xx_hal_fmpi2c.h"
#endif /* HAL_FMPI2C_MODULE_ENABLED */

#ifdef HAL_FMPSMBUS_MODULE_ENABLED
 #include "stm32f4xx_hal_fmpsmbus.h"
#endif /* HAL_FMPSMBUS_MODULE_ENABLED */

#ifdef HAL_SPDIFRX_MODULE_ENABLED
 #include "stm32f4xx_hal_spdifrx.h"
#endif /* HAL_SPDIFRX_MODULE_ENABLED */

#ifdef HAL_DFSDM_MODULE_ENABLED
 #include "stm32f4xx_hal_dfsdm.h"
#endif /* HAL_DFSDM_MODULE_ENABLED */

#ifdef HAL_LPTIM_MODULE_ENABLED
 #include "stm32f4xx_hal_lptim.h"
#endif /* HAL_LPTIM_MODULE_ENABLED */

#ifdef HAL_MMC_MODULE_ENABLED
 #include "stm32f4xx_hal_mmc.h"
#endif /* HAL_MMC_MODULE_ENABLED */

/* Exported macro ------------------------------------------------------------*/
#ifdef  USE_FULL_ASSERT
  #define assert_param(expr) ((expr) ? (void)0U : assert_failed((uint8_t *)__FILE__, __LINE__))
/* Exported functions ------------------------------------------------------- */
  void assert_failed(uint8_t* file, uint32_t line);
#else
  #define assert_param(expr) ((void)0U)
#endif /* USE_FULL_ASSERT */

```

```
00487 #ifdef __cplusplus
00488 }
00489 #endif
00490
00491 #endif /* __STM32F4xx_HAL_CONF_H */
```

5.16 C:/Users/Administrator/Documents/GitHub/ENIB_Robot_Mobile_↩ Ros2/WORKSPACE_F411_uROS6/base_robot/Core/Inc/stm32f4xx↩ _it.h File Reference

This file contains the headers of the interrupt handlers.

Functions

- void NMI_Handler (void)
- void HardFault_Handler (void)
- void MemManage_Handler (void)
- void BusFault_Handler (void)
- void UsageFault_Handler (void)

  This function handles Undefined instruction or illegal state.
- void DebugMon_Handler (void)
- void DMA1_Stream5_IRQHandler (void)
- void DMA1_Stream6_IRQHandler (void)
- void **TIM1_UP_TIM10_IRQHandler** (void)
- void USART1_IRQHandler (void)
- void USART2_IRQHandler (void)
- void DMA2_Stream2_IRQHandler (void)
- void DMA2_Stream7_IRQHandler (void)

5.16.1 Detailed Description

This file contains the headers of the interrupt handlers.

Attention

Definition in file stm32f4xx_it.h.

5.16.2 Function Documentation

5.16.2.1 BusFault_Handler()

```
void BusFault_Handler (
             void  )
```

Definition at line 38 of file stm32f4xx_it.c.

5.16.2.2 DebugMon_Handler()

```
void DebugMon_Handler (
             void  )
```

Definition at line 55 of file stm32f4xx_it.c.

5.16.2.3 DMA1_Stream5_IRQHandler()

```
void DMA1_Stream5_IRQHandler (
             void  )
```

Definition at line 66 of file stm32f4xx_it.c.

5.16.2.4 DMA1_Stream6_IRQHandler()

```
void DMA1_Stream6_IRQHandler (
             void  )
```

Definition at line 72 of file stm32f4xx_it.c.

5.16.2.5 DMA2_Stream2_IRQHandler()

```
void DMA2_Stream2_IRQHandler (
             void  )
```

Definition at line 97 of file stm32f4xx_it.c.

5.16.2.6 DMA2_Stream7_IRQHandler()

```
void DMA2_Stream7_IRQHandler (
             void  )
```

Definition at line 102 of file stm32f4xx_it.c.

5.16.2.7 HardFault_Handler()

```
void HardFault_Handler (
             void  )
```

Definition at line 22 of file stm32f4xx_it.c.

5.16.2.8 MemManage_Handler()

```
void MemManage_Handler (
             void  )
```

Definition at line 30 of file stm32f4xx_it.c.

5.16.2.9 NMI_Handler()

```
void NMI_Handler (
             void  )
```

Definition at line 15 of file stm32f4xx_it.c.

5.16.2.10 UsageFault_Handler()

```
void UsageFault_Handler (
             void  )
```

This function handles Undefined instruction or illegal state.

Definition at line 48 of file stm32f4xx_it.c.

5.16.2.11 USART1_IRQHandler()

```
void USART1_IRQHandler (
             void  )
```

Definition at line 87 of file stm32f4xx_it.c.

5.16.2.12 USART2_IRQHandler()

```
void USART2_IRQHandler (
             void  )
```

Definition at line 92 of file stm32f4xx_it.c.

5.17 stm32f4xx_it.h

Go to the documentation of this file.

```
/* USER CODE BEGIN Header */
/* USER CODE END Header */

/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __STM32F4xx_IT_H
#define __STM32F4xx_IT_H

#ifdef __cplusplus
 extern "C" {
#endif

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* Exported types ------------------------------------------------------------*/
/* USER CODE BEGIN ET */

/* USER CODE END ET */

/* Exported constants --------------------------------------------------------*/
/* USER CODE BEGIN EC */

/* USER CODE END EC */

/* Exported macro ------------------------------------------------------------*/
/* USER CODE BEGIN EM */

/* USER CODE END EM */

/* Exported functions prototypes ---------------------------------------------*/
void NMI_Handler(void);
void HardFault_Handler(void);
void MemManage_Handler(void);
void BusFault_Handler(void);
void UsageFault_Handler(void);
void DebugMon_Handler(void);
void DMA1_Stream5_IRQHandler(void);
void DMA1_Stream6_IRQHandler(void);
void TIM1_UP_TIM10_IRQHandler(void);
void USART1_IRQHandler(void);
void USART2_IRQHandler(void);
void DMA2_Stream2_IRQHandler(void);
void DMA2_Stream7_IRQHandler(void);
/* USER CODE BEGIN EFP */

/* USER CODE END EFP */

#ifdef __cplusplus
}
#endif

#endif /* __STM32F4xx_IT_H */
```

5.18 systemclock.h

```
/*
 * systemclock.h
 *
 *  Created on: Mar 13, 2023
 *      Author: kerhoas
 */

#ifndef INC_SYSTEMCLOCK_H_
#define INC_SYSTEMCLOCK_H_

void SystemClock_Config(void);

#endif /* INC_SYSTEMCLOCK_H_ */
```

5.19 util.h

```
/*
 * utils.h
 */

#ifndef INC_UTIL_H_
#define INC_UTIL_H_

#include "main.h"

void num2str(char *s, unsigned int number, unsigned int base, unsigned int size, int sp);
unsigned int str2num(char *s, unsigned base);
void reverse(char *str, int len);
int intToStr(int x, char str[], int d);
void float2str( char *res, float n, int afterpoint);
double myPow(double x, int n) ;
void flush_ch(char* ch, int ch_size);
int size_ch(char* ch, int ch_size_max);


#endif /* INC_UTIL_H_ */
```

5.20 VL53L0X.h

```
//#define bool  uint8_t
#define true  1
#define false 0

#define SYSRANGE_START 0x00
```

```
#define SYSTEM_THRESH_HIGH 0x0C
#define SYSTEM_THRESH_LOW 0x0E
#define SYSTEM_SEQUENCE_CONFIG 0x01
#define SYSTEM_RANGE_CONFIG 0x09
#define SYSTEM_INTERMEASUREMENT_PERIOD 0x04


#define SYSTEM_INTERRUPT_GPIO_CONFIG 0x0A

//GPIO Config
#define GPIO_HV_MUX_ACTIVE_HIGH 0x84
#define SYSTEM_INTERRUPT_CLEAR 0x0B
#define I2C_MODE 0x88

// Result registers
#define RESULT_INTERRUPT_STATUS 0x13
#define RESULT_RANGE_STATUS 0x14
#define RESULT_CORE_AMBIENT_WINDOW_EVENTS_RTN 0xBC
#define RESULT_CORE_RANGING_TOTAL_EVENTS_RTN 0xC0
#define RESULT_CORE_AMBIENT_WINDOW_EVENTS_REF 0xD0
#define RESULT_CORE_RANGING_TOTAL_EVENTS_REF 0xD4
#define RESULT_PEAK_SIGNAL_RATE_REF 0xB6

//Algo Register
#define ALGO_PART_TO_PART_RANGE_OFFSET_MM 0x28

//Check limit register
#define MSRC_CONFIG_CONTROL 0x60
#define PRE_RANGE_CONFIG_MIN_SNR 0x27
#define PRE_RANGE_CONFIG_VALID_PHASE_LOW 0x56
#define PRE_RANGE_CONFIG_VALID_PHASE_HIGH 0x57
#define PRE_RANGE_MIN_COUNT_RATE_RTN_LIMIT 0x64
#define FINAL_RANGE_CONFIG_MIN_SNR 0x67
#define FINAL_RANGE_CONFIG_VALID_PHASE_LOW 0x47
#define FINAL_RANGE_CONFIG_VALID_PHASE_HIGH 0x48
#define FINAL_RANGE_CONFIG_MIN_COUNT_RATE_RTN_LIMIT 0x44

// PRE RANGE registers
#define PRE_RANGE_CONFIG_SIGMA_THRESH_HI 0x61
#define PRE_RANGE_CONFIG_SIGMA_THRESH_LO 0x62
#define PRE_RANGE_CONFIG_VCSEL_PERIOD 0x50
#define PRE_RANGE_CONFIG_TIMEOUT_MACROP_HI 0x51
#define PRE_RANGE_CONFIG_TIMEOUT_MACROP_LO 0x52

//Internal tuning registers
#define INTERNAL_TUNING_1 0x91
#define INTERNAL_TUNING_2 0xFF


//Other registers
#define SYSTEM_HISTOGRAM_BIN 0x81
#define HISTOGRAM_CONFIG_INITIAL_PHASE_SELECT 0x33
#define HISTOGRAM_CONFIG_READOUT_CTRL 0x55
#define FINAL_RANGE_CONFIG_VCSEL_PERIOD 0x70
#define FINAL_RANGE_CONFIG_TIMEOUT_MACROP_HI 0x71
#define FINAL_RANGE_CONFIG_TIMEOUT_MACROP_LO 0x72
#define CROSSTALK_COMPENSATION_PEAK_RATE_MCPS 0x20
#define MSRC_CONFIG_TIMEOUT_MACROP 0x46
#define GLOBAL_CONFIG_SPAD_ENABLES_REF0 0x0B0
#define GLOBAL_CONFIG_SPAD_ENABLES_REF1 0x0B1
#define GLOBAL_CONFIG_SPAD_ENABLES_REF2 0x0B2
#define GLOBAL_CONFIG_SPAD_ENABLES_REF3 0x0B3
#define GLOBAL_CONFIG_SPAD_ENABLES_REF4 0x0B4
#define GLOBAL_CONFIG_SPAD_ENABLES_REF5 0x0B5
#define GLOBAL_CONFIG_REF_EN_START_SELECT 0xB6
#define DYNAMIC_SPAD_NUM_REQUESTED_REF_SPAD 0x4E
#define DYNAMIC_SPAD_REF_EN_START_OFFSET 0x4F
#define POWER_MANAGEMENT_GO1_POWER_FORCE 0x80
#define VHV_CONFIG_PAD_SCL_SDA__EXTSUP_HV 0x89
#define ALGO_PHASECAL_LIM 0x30
#define ALGO_PHASECAL_CONFIG_TIMEOUT 0x30


#define SYSTEM_THRESH_HIGH                           0x0C
#define SYSTEM_THRESH_LOW                           0x0E

#define  SYSTEM_SEQUENCE_CONFIG                       0x01
#define  SYSTEM_RANGE_CONFIG                          0x09
#define  SYSTEM_INTERMEASUREMENT_PERIOD               0x04

#define  SYSTEM_INTERRUPT_CONFIG_GPIO                 0x0A

#define  GPIO_HV_MUX_ACTIVE_HIGH                      0x84

#define  SYSTEM_INTERRUPT_CLEAR                       0x0B

#define RESULT_INTERRUPT_STATUS                     0x13
```

```
#define  RESULT_RANGE_STATUS                          0x14

#define  RESULT_CORE_AMBIENT_WINDOW_EVENTS_RTN        0xBC
#define  RESULT_CORE_RANGING_TOTAL_EVENTS_RTN         0xC0
#define  RESULT_CORE_AMBIENT_WINDOW_EVENTS_REF        0xD0
#define  RESULT_CORE_RANGING_TOTAL_EVENTS_REF         0xD4
#define  RESULT_PEAK_SIGNAL_RATE_REF                  0xB6

#define  ALGO_PART_TO_PART_RANGE_OFFSET_MM            0x28

//#define  I2C_SLAVE_DEVICE_ADDRESS                     0x8A #################
#define  I2C_SLAVE_DEVICE_ADDRESS                     0x53

#define  MSRC_CONFIG_CONTROL                          0x60

#define  PRE_RANGE_CONFIG_MIN_SNR                     0x27
#define  PRE_RANGE_CONFIG_VALID_PHASE_LOW             0x56
#define  PRE_RANGE_CONFIG_VALID_PHASE_HIGH            0x57
#define  PRE_RANGE_MIN_COUNT_RATE_RTN_LIMIT           0x64

#define  FINAL_RANGE_CONFIG_MIN_SNR                   0x67
#define  FINAL_RANGE_CONFIG_VALID_PHASE_LOW           0x47
#define  FINAL_RANGE_CONFIG_VALID_PHASE_HIGH          0x48
#define  FINAL_RANGE_CONFIG_MIN_COUNT_RATE_RTN_LIMIT  0x44

#define  PRE_RANGE_CONFIG_SIGMA_THRESH_HI             0x61
#define  PRE_RANGE_CONFIG_SIGMA_THRESH_LO             0x62

#define  PRE_RANGE_CONFIG_VCSEL_PERIOD                0x50
#define  PRE_RANGE_CONFIG_TIMEOUT_MACROP_HI          0x51
#define  PRE_RANGE_CONFIG_TIMEOUT_MACROP_LO           0x52

#define  SYSTEM_HISTOGRAM_BIN                         0x81
#define  HISTOGRAM_CONFIG_INITIAL_PHASE_SELECT        0x33
#define  HISTOGRAM_CONFIG_READOUT_CTRL                0x55

#define  FINAL_RANGE_CONFIG_VCSEL_PERIOD              0x70
#define  FINAL_RANGE_CONFIG_TIMEOUT_MACROP_HI         0x71
#define  FINAL_RANGE_CONFIG_TIMEOUT_MACROP_LO         0x72
#define CROSSTALK_COMPENSATION_PEAK_RATE_MCPS       0x20

#define MSRC_CONFIG_TIMEOUT_MACROP                   0x46

#define  SOFT_RESET_GO2_SOFT_RESET_N                  0xBF
#define  IDENTIFICATION_MODEL_ID                      0xC0
#define  IDENTIFICATION_REVISION_ID                  0xC2

#define  OSC_CALIBRATE_VAL                            0xF8

#define  GLOBAL_CONFIG_VCSEL_WIDTH                    0x32
#define  GLOBAL_CONFIG_SPAD_ENABLES_REF_0             0xB0
#define  GLOBAL_CONFIG_SPAD_ENABLES_REF_1             0xB1
#define  GLOBAL_CONFIG_SPAD_ENABLES_REF_2             0xB2
#define  GLOBAL_CONFIG_SPAD_ENABLES_REF_3            0xB3
#define  GLOBAL_CONFIG_SPAD_ENABLES_REF_4             0xB4
#define  GLOBAL_CONFIG_SPAD_ENABLES_REF_5             0xB5

#define  GLOBAL_CONFIG_REF_EN_START_SELECT            0xB6
#define  DYNAMIC_SPAD_NUM_REQUESTED_REF_SPAD          0x4E
#define  DYNAMIC_SPAD_REF_EN_START_OFFSET             0x4F
#define  POWER_MANAGEMENT_GO1_POWER_FORCE             0x80

#define  VHV_CONFIG_PAD_SCL_SDA__EXTSUP_HV            0x89

#define  ALGO_PHASECAL_LIM                            0x30
#define  ALGO_PHASECAL_CONFIG_TIMEOUT                 0x30

//------------------------------------------------------------
// Defines
//------------------------------------------------------------
// I use a 8-bit number for the address, LSB must be 0 so that I can
// OR over the last bit correctly based on reads and writes
#define ADDRESS_DEFAULT 0b01010010

// Record the current time to check an upcoming timeout against
#define startTimeout() (g_timeoutStartMs = millis())

// Check if timeout is enabled (set to nonzero value) and has expired
#define checkTimeoutExpired() (g_ioTimeout > 0 && ((uint16_t)millis() - g_timeoutStartMs) > g_ioTimeout)

// Decode VCSEL (vertical cavity surface emitting laser) pulse period in PCLKs
// from register value
// based on VL53L0X_decode_vcsel_period()
#define decodeVcselPeriod(reg_val)      (((reg_val) + 1) « 1)

// Encode VCSEL pulse period register value from period in PCLKs
// based on VL53L0X_encode_vcsel_period()
#define encodeVcselPeriod(period_pclks) (((period_pclks) » 1) - 1)

// Calculate macro period in *nanoseconds* from VCSEL period in PCLKs
// based on VL53L0X_calc_macro_period_ps()
// PLL_period_ps = 1655; macro_period_vclks = 2304
#define calcMacroPeriod(vcsel_period_pclks) ((((uint32_t)2304 * (vcsel_period_pclks) * 1655) + 500) / 1000)

// register addresses from API vl53l0x_device.h (ordered as listed there)


typedef enum { VcselPeriodPreRange, VcselPeriodFinalRange }vcselPeriodType;

// Additional info for one measurement
typedef struct{
  uint16_t rawDistance; //uncorrected distance  [mm],   uint16_t
  uint16_t signalCnt;   //Signal  Counting Rate [mcps], uint16_t, fixpoint9.7
  uint16_t ambientCnt;  //Ambient Counting Rate [mcps], uint16_t, fixpoint9.7
  uint16_t spadCnt;     //Effective SPAD return count,  uint16_t, fixpoint8.8
  uint8_t  rangeStatus; //Ranging status (0-15)
} statInfo_t;


//---------------------------------------------------------------
// API Functions
//---------------------------------------------------------------
// configures chip i2c and lib for `new_addr` (8 bit, LSB=0)
void setAddress(uint8_t new_addr);
// Returns the current I2C address.
uint8_t getAddress(void);

// Iniitializes and configures the sensor.
// If the optional argument io_2v8 is 1, the sensor is configured for 2V8 mode (2.8 V I/O);
// if 0, the sensor is left in 1V8 mode. Returns 1 if the initialization completed successfully.
uint8_t initVL53L0X();

// Sets the return signal rate limit to the given value in units of MCPS (mega counts per second).
// This is the minimum amplitude of the signal reflected from the target and received by the sensor
//  necessary for it to report a valid reading. Setting a lower limit increases the potential range
// of the sensor but also increases the likelihood of getting an inaccurate reading because of
//  reflections from objects other than the intended target. This limit is initialized to 0.25 MCPS
//  by default. The return value is a boolean indicating whether the requested limit was valid.
uint8_t setSignalRateLimit(float limit_Mcps);

// Returns the current return signal rate limit in MCPS.
float getSignalRateLimit(void);

// Set the measurement timing budget in microseconds, which is the time allowed
// for one measurement; the ST API and this library take care of splitting the
// timing budget among the sub-steps in the ranging sequence. A longer timing
// budget allows for more accurate measurements. Increasing the budget by a
// factor of N decreases the range measurement standard deviation by a factor of
// sqrt(N). Defaults to about 33 milliseconds; the minimum is 20 ms.
// based on VL53L0X_set_measurement_timing_budget_micro_seconds()
uint8_t setMeasurementTimingBudget(uint32_t budget_us);

// Returns the current measurement timing budget in microseconds.
uint32_t getMeasurementTimingBudget(void);

// Sets the VCSEL (vertical cavity surface emitting laser) pulse period for the given period type
// (VcselPeriodPreRange or VcselPeriodFinalRange) to the given value (in PCLKs).
// Longer periods increase the potential range of the sensor. Valid values are (even numbers only):
// Pre: 12 to 18 (initialized to 14 by default)
// Final: 8 to 14 (initialized to 10 by default)
// The return value is a boolean indicating whether the requested period was valid.
uint8_t setVcselPulsePeriod(vcselPeriodType type, uint8_t period_pclks);

// Returns the current VCSEL pulse period for the given period type.
uint8_t getVcselPulsePeriod(vcselPeriodType type);

// Starts continuous ranging measurements. If the argument period_ms is 0,
// continuous back-to-back mode is used (the sensor takes measurements as often as possible);
// if it is nonzero, continuous timed mode is used, with the specified inter-measurement period
// in milliseconds determining how often the sensor takes a measurement.
void startContinuous(uint32_t period_ms);

// Stops continuous mode.
void stopContinuous(void);

// Returns a range reading in millimeters when continuous mode is active.
// Additional measurement data will be copied into `extraStats` if it is non-zero.
uint16_t readRangeContinuousMillimeters(/* statInfo_t *extraStats*/ );

// Performs a single-shot ranging measurement and returns the reading in millimeters.
// Additional measurement data will be copied into `extraStats` if it is non-zero.
uint16_t readRangeSingleMillimeters( /*statInfo_t *extraStats */);
```

```

// Sets a timeout period in milliseconds after which read operations will abort
// if the sensor is not ready. A value of 0 disables the timeout.
void setTimeout(uint16_t timeout);

// Returns the current timeout period setting.
uint16_t getTimeout(void);

// Indicates whether a read timeout has occurred since the last call to timeoutOccurred().
bool timeoutOccurred(void);

//---------------------------------------------------------
// I2C communication Functions
//---------------------------------------------------------
void writeReg(uint8_t reg, uint8_t value);        // Write an 8-bit register
void writeReg16Bit(uint8_t reg, uint16_t value);  // Write a 16-bit register
void writeReg32Bit(uint8_t reg, uint32_t value);  // Write a 32-bit register
uint8_t readReg(uint8_t reg);                     // Read an 8-bit register
uint16_t readReg16Bit(uint8_t reg);               // Read a 16-bit register
uint32_t readReg32Bit(uint8_t reg);               // Read a 32-bit register
// Write `count` number of bytes from `src` to the sensor, starting at `reg`
void writeMulti(uint8_t reg, uint8_t const *src, uint8_t count);

// TCC: Target CentreCheck
// MSRC: Minimum Signal Rate Check
// DSS: Dynamic Spad Selection
typedef struct {
  uint8_t tcc, msrc, dss, pre_range, final_range;
}SequenceStepEnables;

typedef struct {
  uint16_t pre_range_vcsel_period_pclks, final_range_vcsel_period_pclks;

  uint16_t msrc_dss_tcc_mclks, pre_range_mclks, final_range_mclks;
  uint32_t msrc_dss_tcc_us,    pre_range_us,    final_range_us;
}SequenceStepTimeouts;

```

5.21 captDistIR.c

```
/*
 * IRMeasure.c
 */


#include "captDistIR.h"

ADC_HandleTypeDef   adcHandle;
ADC_HandleTypeDef   adcHandle_12;
ADC_HandleTypeDef   adcHandle_13;
ADC_ChannelConfTypeDef   sConfig;

//=================================================================
//          ADC INIT FOR IR SENSOR SHARP GP2D12
//=================================================================

void  captDistIR_Init(void)
{
    adcHandle.Instance      = ADC1;

    adcHandle.Init.ClockPrescaler = ADC_CLOCKPRESCALER_PCLK_DIV2;
    adcHandle.Init.DataAlign = ADC_DATAALIGN_RIGHT;
    adcHandle.Init.Resolution = ADC_RESOLUTION12b;
    // Don't do continuous conversions - do them on demand
    adcHandle.Init.ContinuousConvMode    = DISABLE;  // Continuous mode disabled to have only 1 conversion at each conversion trig
    // Disable the scan conversion so we do one at a time */
    adcHandle.Init.ScanConvMode = DISABLE;
    //Say how many channels would be used by the sequencer
    adcHandle.Init.NbrOfConversion = 2;
    adcHandle.Init.DiscontinuousConvMode = DISABLE;  // Parameter discarded because sequencer is disabled
    adcHandle.Init.NbrOfDiscConversion = 2;
    adcHandle.Init.ExternalTrigConvEdge =  ADC_EXTERNALTRIGCONVEDGE_NONE ;
    //Start conversion by software, not an external trigger
    adcHandle.Init.ExternalTrigConv = 0;
    adcHandle.Init.DMAContinuousRequests = DISABLE;
    adcHandle.Init.EOCSelection = DISABLE;

    HAL_ADC_Init(&adcHandle);
}

//=================================================================
//          IR GET (POLL METHOD)
//=================================================================

int  captDistIR_Get(int* tab)
{
    sConfig.Channel      = ADC_CHANNEL_4;
    sConfig.Rank         = 1;
    sConfig.SamplingTime = ADC_SAMPLETIME_56CYCLES;
    HAL_ADC_ConfigChannel(&adcHandle, &sConfig);

    HAL_ADC_Start(&adcHandle);                  //Start the conversion
    HAL_ADC_PollForConversion(&adcHandle,10);   //Processing the conversion
    tab[0]=HAL_ADC_GetValue(&adcHandle);        //Return the converted data

    sConfig.Channel      = ADC_CHANNEL_8;
    sConfig.Rank         = 1;
    sConfig.SamplingTime = ADC_SAMPLETIME_56CYCLES;
    HAL_ADC_ConfigChannel(&adcHandle, &sConfig);

    HAL_ADC_Start(&adcHandle);                      //Start the conversion
    HAL_ADC_PollForConversion(&adcHandle,10);       //Processing the conversion
    tab[1]=HAL_ADC_GetValue(&adcHandle);            //Return the converted data

    return 0;
}
//=================================================================
```

5.22 dma_transport.c

```
#include <uxr/client/transport.h>

#include <rmw_microxrcedds_c/config.h>

#include "main.h"
#include "cmsis_os.h"

#include <unistd.h>
#include <stdio.h>
#include <string.h>
#include <stdbool.h>

#ifdef RMW_UXRCE_TRANSPORT_CUSTOM

// --- micro-ROS Transports ---
#define UART_DMA_BUFFER_SIZE 2048

static uint8_t dma_buffer[UART_DMA_BUFFER_SIZE];
static size_t dma_head = 0, dma_tail = 0;

bool cubemx_transport_open(struct uxrCustomTransport * transport){
    UART_HandleTypeDef * uart = (UART_HandleTypeDef*) transport->args;
    HAL_UART_Receive_DMA(uart, dma_buffer, UART_DMA_BUFFER_SIZE);
    return true;
}

bool cubemx_transport_close(struct uxrCustomTransport * transport){
    UART_HandleTypeDef * uart = (UART_HandleTypeDef*) transport->args;
    HAL_UART_DMAStop(uart);
    return true;
}

size_t cubemx_transport_write(struct uxrCustomTransport* transport, uint8_t * buf, size_t len, uint8_t * err){
    UART_HandleTypeDef * uart = (UART_HandleTypeDef*) transport->args;

    HAL_StatusTypeDef ret;
    if (uart->gState == HAL_UART_STATE_READY){
        ret = HAL_UART_Transmit_DMA(uart, buf, len);
        while (ret == HAL_OK && uart->gState != HAL_UART_STATE_READY){
            osDelay(1);
        }

        return (ret == HAL_OK) ? len : 0;
    }else{
        return 0;
    }
}

size_t cubemx_transport_read(struct uxrCustomTransport* transport, uint8_t* buf, size_t len, int timeout, uint8_t* err){
    UART_HandleTypeDef * uart = (UART_HandleTypeDef*) transport->args;

    int ms_used = 0;
    do
    {
        __disable_irq();
        dma_tail = UART_DMA_BUFFER_SIZE - __HAL_DMA_GET_COUNTER(uart->hdmarx);
        __enable_irq();
        ms_used++;
        osDelay(portTICK_RATE_MS);
    } while (dma_head == dma_tail && ms_used < timeout);

    size_t wrote = 0;
    while ((dma_head != dma_tail) && (wrote < len)){
        buf[wrote] = dma_buffer[dma_head];
        dma_head = (dma_head + 1) % UART_DMA_BUFFER_SIZE;
        wrote++;
    }

    return wrote;
}

#endif //RMW_UXRCE_TRANSPORT_CUSTOM
```

5.23 drv_gpio.c

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

void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};

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

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin : B1_Pin */
  GPIO_InitStruct.Pin = B1_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(B1_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pin : LD2_Pin */
  GPIO_InitStruct.Pin = LD2_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(LD2_GPIO_Port, &GPIO_InitStruct);

}

extern void quadEncoder_CallbackIndexL(void);
extern void quadEncoder_CallbackIndexR(void);

void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    switch(GPIO_Pin)
    {
    case GPIO_PIN_0 :
        quadEncoder_CallbackIndexR();
                    break;

    case GPIO_PIN_1 :

                    break;

    case GPIO_PIN_3:
                    break;

    case GPIO_PIN_10:
        quadEncoder_CallbackIndexL();
                    break;

    case GPIO_PIN_13 :      // USER BUTTON
                    break;

    default :       break;


    }
}
```

5.24 drv_i2c.c

```c
#include "main.h"
#include <string.h>
#include "drv_i2c.h"


I2C_HandleTypeDef hi2c1;

void MX_I2C1_Init(void)
{

  hi2c1.Instance = I2C1;
  hi2c1.Init.ClockSpeed = 100000;
  hi2c1.Init.DutyCycle = I2C_DUTYCYCLE_2;
  hi2c1.Init.OwnAddress1 = 0;
  hi2c1.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
  hi2c1.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
  hi2c1.Init.OwnAddress2 = 0;
  hi2c1.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
  hi2c1.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;
  if (HAL_I2C_Init(&hi2c1) != HAL_OK)
  {
    Error_Handler();
  }

}





//======================================================================
// Transmit n_data bytes to i2c slave
//======================================================================
int i2c1_WriteBuffer(uint16_t addrSlave, uint8_t *data, int n_data)
{
    int status;
    status = HAL_I2C_Master_Transmit(&hi2c1, addrSlave, data, n_data , 100);
    return status;
}
//======================================================================
// Receive n_data bytes from i2c slave
//======================================================================
int i2c1_ReadBuffer(uint16_t addrSlave, uint8_t *data, int n_data)
{
    int status;
    status = HAL_I2C_Master_Receive(&hi2c1, addrSlave, data, n_data , 100);
    return status;
}
//======================================================================
// Receive n_data bytes - located at regAddr - from i2c slave
//======================================================================
int i2c1_ReadRegBuffer(uint16_t addrSlave, uint8_t  regAddr,  uint8_t *data, int n_data)
{
    int status;
    uint8_t RegAddr;
    RegAddr=regAddr;
    do{
        status=HAL_I2C_Master_Transmit(&hi2c1, addrSlave, &RegAddr, 1, 100);
        if( status )
            break;
        status =HAL_I2C_Master_Receive(&hi2c1, addrSlave, data, n_data, n_data*100);
    }while(0);
    return status;
}

//======================================================================
// Write n_data bytes - have to be written at regAddr - to i2c slave
//======================================================================
int i2c1_WriteRegBuffer(uint16_t addrSlave, uint8_t  regAddr,  uint8_t *data, int n_data)
{
    int status;
    uint8_t RegAddr[0x10];
    RegAddr[0]=regAddr;
    memcpy(RegAddr+1, data, n_data);
   status=HAL_I2C_Master_Transmit(&hi2c1, addrSlave, RegAddr, n_data+1, 100);
     return status;
}

//======================================================================
// Write 1 byte at regAddr Slave - Interrupt Method
//======================================================================
void i2c1_WriteRegByte_IT(uint16_t addrSlave, uint8_t  regAddr,  uint8_t data)
{

uint8_t datas_to_send[2];

datas_to_send[0]=regAddr;
datas_to_send[1]=data;

    while(HAL_I2C_Master_Transmit_IT(&hi2c1, addrSlave, datas_to_send, 2)!= HAL_OK){}
    while (HAL_I2C_GetState(&hi2c1) != HAL_I2C_STATE_READY){}
}
//======================================================================
// Read 1 byte from regAddr Slave - Interrupt Method
//======================================================================
void i2c1_ReadRegBuffer_IT(uint16_t addrSlave, uint8_t  regAddr,  uint8_t* datas, int len)
{
    while(HAL_I2C_Master_Transmit_IT(&hi2c1, addrSlave, &regAddr, 1)!= HAL_OK){}
    while (HAL_I2C_GetState(&hi2c1) != HAL_I2C_STATE_READY){}

    while(HAL_I2C_Master_Receive_IT(&hi2c1, addrSlave, datas, len)!= HAL_OK){}
    while( HAL_I2C_GetState(&hi2c1) != HAL_I2C_STATE_READY ){}
}

//======================================================================
// Write 1 byte to regAddr (16 bits) Slave
//======================================================================
int i2c1_WriteReg16Byte(uint16_t addrSlave, uint16_t  regAddr,  uint8_t data)
{
    int status;
    uint8_t buffer[3];
    buffer[0]=regAddr»8;
    buffer[1]=regAddr&0xFF;
    buffer[2]=data;

    status = HAL_I2C_Master_Transmit(&hi2c1, addrSlave, buffer, 3 , 100);
    return status;

}
//======================================================================
// Write 16 bits word to regAddr (16 bits) Slave
//======================================================================
int i2c1_WriteReg16Word16(uint16_t addrSlave, uint16_t  regAddr,  uint16_t data)
{
    int status;
    uint8_t buffer[4];
    buffer[0]=regAddr»8;
    buffer[1]=regAddr&0xFF;
    buffer[2]=data»8;
    buffer[3]=data&0xFF;

    status = HAL_I2C_Master_Transmit(&hi2c1, addrSlave, buffer, 4 , 100);
    return status;
}
//======================================================================
// Write 32 bits word to regAddr (16 bits) Slave
//======================================================================
int i2c1_WriteReg16Word32(uint16_t addrSlave, uint16_t  regAddr,  uint32_t data)
{
    int status;
    uint8_t buffer[4];
    buffer[0]=regAddr»8;
    buffer[1]=regAddr&0xFF;
    buffer[2]=data»24;
    buffer[3]=(data»16)&0xFF;
    buffer[4]=(data»8)&0xFF;;
    buffer[5]=data&0xFF;

    status = HAL_I2C_Master_Transmit(&hi2c1, addrSlave, buffer, 6 , 100);
    return status;
}
//======================================================================
// Read 1 byte from regAddr (16 bits) Slave
//======================================================================
int i2c1_ReadReg16Byte(uint16_t addrSlave, uint16_t  regAddr, uint8_t *data)
{
    int status;
    uint8_t buffer[2];

    buffer[0]=regAddr»8;
    buffer[1]=regAddr&0xFF;

    status=HAL_I2C_Master_Transmit(&hi2c1, addrSlave, buffer, 2, 100);
     if(!status ) {
        status =HAL_I2C_Master_Receive(&hi2c1, addrSlave, buffer, 1, 100);
        if( !status ){
            *data=buffer[0];
        }
     }

    return status;
}
```

```
//=====================================================================
// Read 16 bits word from regAddr (16 bits) Slave
//=====================================================================
int i2c1_ReadReg16Word16(uint16_t addrSlave, uint16_t  regAddr, uint16_t *data)
{
    int status;
    uint8_t buffer[2];

    buffer[0]=regAddr»8;
    buffer[1]=regAddr&0xFF;

    status=HAL_I2C_Master_Transmit(&hi2c1, addrSlave, buffer, 2, 100);
     if(!status ) {
        status =HAL_I2C_Master_Receive(&hi2c1, addrSlave, buffer, 2, 100);
        if( !status ){
              //VL6180x register are Big endian if cpu is be direct read direct into *data is possible
             *data=((uint16_t)buffer[0]«8)|(uint16_t)buffer[1];
        }
     }

    return status;
}
//=====================================================================
// Read 32 bits word from regAddr (16 bits) Slave
//=====================================================================
int i2c1_ReadReg16Word32(uint16_t addrSlave, uint16_t  regAddr, uint32_t *data)
{
    int status;
    uint8_t buffer[4];

    buffer[0]=regAddr»8;
    buffer[1]=regAddr&0xFF;

    status=HAL_I2C_Master_Transmit(&hi2c1, addrSlave, buffer, 2, 100);
     if(!status ) {
        status =HAL_I2C_Master_Receive(&hi2c1, addrSlave, buffer, 4, 100);
        if( !status ){
              //VL6180x register are Big endian if cpu is be direct read direct into *data is possible
*data=((uint32_t)buffer[0]«24)|((uint32_t)buffer[1]«16)|((uint32_t)buffer[2]«8)|((uint32_t)buffer[3]);
        }
     }

    return status;
}

//=====================================================================
// Read n_data bytes from regAddr (16 bits) Slave
//=====================================================================
int i2c1_ReadReg16Buffer(uint16_t addrSlave, uint16_t  regAddr,  uint8_t *data, int n_data)
{
    int status;
    uint8_t buffer[2];

    buffer[0]=regAddr»8;
    buffer[1]=regAddr&0xFF;

    status=HAL_I2C_Master_Transmit(&hi2c1, addrSlave, buffer, 2, 100);
    if( !status ){
        status=HAL_I2C_Master_Receive(&hi2c1, addrSlave, data, n_data, n_data*100);
    }



    return status;
}
```

5.25 drv_uart.c

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

UART_HandleTypeDef huart1;
UART_HandleTypeDef huart2;
DMA_HandleTypeDef hdma_usart1_rx;
DMA_HandleTypeDef hdma_usart1_tx;
DMA_HandleTypeDef hdma_usart2_rx;
DMA_HandleTypeDef hdma_usart2_tx;
```

```


void MX_USART1_UART_Init(void)
{
  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();
  }
}


void MX_USART2_UART_Init(void)
{
  huart2.Instance = USART2;
  huart2.Init.BaudRate = 115200;
  huart2.Init.WordLength = UART_WORDLENGTH_8B;
  huart2.Init.StopBits = UART_STOPBITS_1;
  huart2.Init.Parity = UART_PARITY_NONE;
  huart2.Init.Mode = UART_MODE_TX_RX;
  huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart2.Init.OverSampling = UART_OVERSAMPLING_16;
  if (HAL_UART_Init(&huart2) != HAL_OK)
  {
    Error_Handler();
  }
}

void MX_DMA_Init(void)
{

  /* DMA controller clock enable */
  __HAL_RCC_DMA2_CLK_ENABLE();
  __HAL_RCC_DMA1_CLK_ENABLE();

  /* DMA interrupt init */
  /* DMA1_Stream5_IRQn interrupt configuration */
  HAL_NVIC_SetPriority(DMA1_Stream5_IRQn, 5, 0);
  HAL_NVIC_EnableIRQ(DMA1_Stream5_IRQn);
  /* DMA1_Stream6_IRQn interrupt configuration */
  HAL_NVIC_SetPriority(DMA1_Stream6_IRQn, 5, 0);
  HAL_NVIC_EnableIRQ(DMA1_Stream6_IRQn);
  /* DMA2_Stream2_IRQn interrupt configuration */
  HAL_NVIC_SetPriority(DMA2_Stream2_IRQn, 5, 0);
  HAL_NVIC_EnableIRQ(DMA2_Stream2_IRQn);
  /* DMA2_Stream7_IRQn interrupt configuration */
  HAL_NVIC_SetPriority(DMA2_Stream7_IRQn, 5, 0);
  HAL_NVIC_EnableIRQ(DMA2_Stream7_IRQn);

}
```

5.26 freertos.c

```
/* USER CODE BEGIN Header */
/* USER CODE END Header */

/* Includes ------------------------------------------------------------------*/
#include "FreeRTOS.h"
#include "task.h"
#include "main.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */

/* 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 ---------------------------------------------------------*/
/* USER CODE BEGIN Variables */

/* USER CODE END Variables */

/* Private function prototypes -----------------------------------------------*/
/* USER CODE BEGIN FunctionPrototypes */

/* USER CODE END FunctionPrototypes */

/* Private application code --------------------------------------------------*/
/* USER CODE BEGIN Application */

/* USER CODE END Application */

```

5.27 groveLCD.c

```
/*
 * groveLCD.c
 *
 *  Created on: Jan 8, 2020
 *      Author: kerhoas
 */

#include "groveLCD.h"
#include <math.h>
#include "util.h"

  uint8_t _displayfunction;
  uint8_t _displaycontrol;
  uint8_t _displaymode;
  uint8_t _initialized;
  uint8_t _numlines,_currline;

//=================================================================
void groveLCD_test()
{
    uint8_t tab[2];
    tab[1] = 100;
  i2c1_WriteRegBuffer(RGB_ADDRESS, REG_RED, tab, 1);

}
//=================================================================
void i2c_send_byte(unsigned char dta)
{
    i2c1_WriteBuffer(LCD_ADDRESS, &dta, 1);
}
//=================================================================
void i2c_send_byteS(unsigned char *dta, unsigned char len)
{
    i2c1_WriteBuffer(LCD_ADDRESS, dta, len);
}
//=================================================================
void groveLCD_begin(uint8_t cols, uint8_t lines, uint8_t dotsize)
{
    if (lines > 1) {
        _displayfunction |= LCD_2LINE;
    }
    _numlines = lines;
    _currline = 0;

    // for some 1 line displays you can select a 10 pixel high font
    if ((dotsize != 0) && (lines == 1)) {
        _displayfunction |= LCD_5x10DOTS;
    }

    // SEE PAGE 45/46 FOR INITIALIZATION SPECIFICATION!
    // according to datasheet, we need at least 40ms after power rises above 2.7V
    // before sending commands. Arduino can turn on way befer 4.5V so we'll wait 50
    HAL_Delay(50);


    // this is according to the hitachi HD44780 datasheet
    // page 45 figure 23

    // Send function set command sequence
    groveLCD_command(LCD_FUNCTIONSET | _displayfunction);
    HAL_Delay(5);  // wait more than 4.1ms

    // second try
    groveLCD_command(LCD_FUNCTIONSET | _displayfunction);
    HAL_Delay(5);

    // third go
    groveLCD_command(LCD_FUNCTIONSET | _displayfunction);


    // finally, set # lines, font size, etc.
    groveLCD_command(LCD_FUNCTIONSET | _displayfunction);

    // turn the display on with no cursor or blinking default
    _displaycontrol = LCD_DISPLAYON | LCD_CURSOROFF | LCD_BLINKOFF;
    groveLCD_display();

    // clear it off
    groveLCD_clear();

    // Initialize to default text direction (for romance languages)
    _displaymode = LCD_ENTRYLEFT | LCD_ENTRYSHIFTDECREMENT;
    // set the entry mode
    groveLCD_command(LCD_ENTRYMODESET | _displaymode);


    // backlight init
    groveLCD_setReg(REG_MODE1, 0);
    // set LEDs controllable by both PWM and GRPPWM registers
    groveLCD_setReg(REG_OUTPUT, 0xFF);
    // set MODE2 values
    // 0010 0000 -> 0x20  (DMBLNK to 1, ie blinky mode)
    groveLCD_setReg(REG_MODE2, 0x20);

    groveLCD_setColorWhite();

}
//=================================================================
void groveLCD_setColorAll(){groveLCD_setRGB(0, 0, 0);}
void groveLCD_setColorWhite(){groveLCD_setRGB(255, 255, 255);}
//=================================================================

/********** high level commands, for the user! */
void groveLCD_clear()
{
    groveLCD_command(LCD_CLEARDISPLAY);        // clear display, set cursor position to zero
    HAL_Delay(2000);          // this command takes a long time!
}
//=================================================================
void groveLCD_home()
{
    groveLCD_command(LCD_RETURNHOME);        // set cursor position to zero
    HAL_Delay(2000);        // this command takes a long time!
}
//=================================================================
void groveLCD_setCursor(uint8_t col, uint8_t row)
{
    col = (row == 0 ? col|0x80 : col|0xc0);
    unsigned char dta[2] = {0x80, col};
    i2c_send_byteS(dta, 2);
}
//=================================================================
// Turn the display on/off (quickly)
void groveLCD_noDisplay()
{
    _displaycontrol &= ~LCD_DISPLAYON;
    groveLCD_command(LCD_DISPLAYCONTROL | _displaycontrol);
}
//=================================================================
void groveLCD_display() {
    _displaycontrol |= LCD_DISPLAYON;
    groveLCD_command(LCD_DISPLAYCONTROL | _displaycontrol);
}
//=================================================================
// Turns the underline cursor on/off
void groveLCD_noCursor()
{
    _displaycontrol &= ~LCD_CURSORON;
    groveLCD_command(LCD_DISPLAYCONTROL | _displaycontrol);
}
//=================================================================
void groveLCD_cursor() {
    _displaycontrol |= LCD_CURSORON;
    groveLCD_command(LCD_DISPLAYCONTROL | _displaycontrol);
}
//=================================================================
// Turn on and off the blinking cursor
```

```
void groveLCD_noBlink()
{
    _displaycontrol &= ~LCD_BLINKON;
    groveLCD_command(LCD_DISPLAYCONTROL | _displaycontrol);
}
//=================================================================
void groveLCD_blink()
{
    _displaycontrol |= LCD_BLINKON;
    groveLCD_command(LCD_DISPLAYCONTROL | _displaycontrol);
}
//=================================================================
// These commands scroll the display without changing the RAM
void groveLCD_scrollDisplayLeft(void)
{
    groveLCD_command(LCD_CURSORSHIFT | LCD_DISPLAYMOVE | LCD_MOVELEFT);
}
//=================================================================
void groveLCD_scrollDisplayRight(void)
{
    groveLCD_command(LCD_CURSORSHIFT | LCD_DISPLAYMOVE | LCD_MOVERIGHT);
}
//=================================================================
// This is for text that flows Left to Right
void groveLCD_leftToRight(void)
{
    _displaymode |= LCD_ENTRYLEFT;
    groveLCD_command(LCD_ENTRYMODESET | _displaymode);
}
//=================================================================
// This is for text that flows Right to Left
void groveLCD_rightToLeft(void)
{
    _displaymode &= ~LCD_ENTRYLEFT;
    groveLCD_command(LCD_ENTRYMODESET | _displaymode);
}
//=================================================================
// This will 'right justify' text from the cursor
void groveLCD_autoscroll(void)
{
    _displaymode |= LCD_ENTRYSHIFTINCREMENT;
    groveLCD_command(LCD_ENTRYMODESET | _displaymode);
}
//=================================================================
// This will 'left justify' text from the cursor
void groveLCD_noAutoscroll(void)
{
    _displaymode &= ~LCD_ENTRYSHIFTINCREMENT;
    groveLCD_command(LCD_ENTRYMODESET | _displaymode);
}
//=================================================================
// Allows us to fill the first 8 CGRAM locations
// with custom characters
void groveLCD_createChar(uint8_t location, uint8_t charmap[])
{
    location &= 0x7; // we only have 8 locations 0-7
    groveLCD_command(LCD_SETCGRAMADDR | (location « 3));

    unsigned char dta[9];
    dta[0] = 0x40;
    for(int i=0; i<8; i++)
    {
        dta[i+1] = charmap[i];
    }
    i2c_send_byteS(dta, 9);
}
//=================================================================
// Control the backlight LED blinking
void groveLCD_blinkLED(void)
{
    // blink period in seconds = (<reg 7> + 1) / 24
    // on/off ratio = <reg 6> / 256
    groveLCD_setReg(0x07, 0x17);  // blink every second
    groveLCD_setReg(0x06, 0x7f);  // half on, half off
}
//=================================================================
void groveLCD_noBlinkLED(void)
{
    groveLCD_setReg(0x07, 0x00);
    groveLCD_setReg(0x06, 0xff);
}
//=================================================================
/*********** mid level commands, for sending data/cmds */

// send command
void groveLCD_command(uint8_t value)
{
    unsigned char dta[2] = {0x80, value};
    i2c_send_byteS(dta, 2);
}
//=================================================================
// send data
int groveLCD_write(uint8_t value)
{
    unsigned char dta[2] = {0x40, value};
    i2c_send_byteS(dta, 2);
    return 1; // assume sucess
}
//=================================================================
void groveLCD_putString(char* s)
{
    while(*s != '\0')
    {
        groveLCD_write(*s);
        s++;
    }
}
//=================================================================
void groveLCD_setReg(unsigned char addr, unsigned char dta)
{
    i2c1_WriteRegBuffer(RGB_ADDRESS, addr,  &dta, 1);
}
//=================================================================
void groveLCD_setRGB(unsigned char r, unsigned char g, unsigned char b)
{
    groveLCD_setReg(REG_RED, r);
    groveLCD_setReg(REG_GREEN, g);
    groveLCD_setReg(REG_BLUE, b);
}
//=================================================================
const unsigned char color_define[4][3] =
{
    {255, 255, 255},            // white
    {255, 0, 0},                // red
    {0, 255, 0},                // green
    {0, 0, 255},                // blue
};
//=================================================================
void groveLCD_setColor(unsigned char color)
{
    if(color > 3)return ;
    groveLCD_setRGB(color_define[color][0], color_define[color][1], color_define[color][2]);
}
//============================================================
void groveLCD_term_printf(const char* fmt, ...)
{
    __gnuc_va_list         ap;
    char          *p;
    char           ch;
    unsigned long  ul;
    unsigned long long ull;
    unsigned long  size;
    unsigned int   sp;
    char           s[60];
    int first=0;

    va_start(ap, fmt);

    while (*fmt != '\0') {
        if (*fmt =='%') {
            size=0; sp=1;
            if (*++fmt=='0') {fmt++; sp=0;} // parse %04d --> sp=0
            ch=*fmt;
            if ((ch>'0') && (ch<='9')) {    // parse %4d --> size=4
                char tmp[10];
                int i=0;
                while ((ch>='0') && (ch<='9')) {
                    tmp[i++]=ch;
                    ch=*++fmt;
                }
                tmp[i]='\0';
                size=str2num(tmp,10);
            }
            switch (ch) {
                case '%':
                    groveLCD_write('%');
                    break;
                case 'c':
                    ch = va_arg(ap, int);
                    groveLCD_write(ch);
                    break;
                case 's':
                    p = va_arg(ap, char *);
                    groveLCD_putString(p);
                    break;
                case 'd':
                    ul = va_arg(ap, long);
                    if ((long)ul < 0) {
                        groveLCD_write('-');
                        ul = -(long)ul;
                        //size--;
                    }
                    num2str(s, ul, 10, size, sp);
                    groveLCD_putString(s);
                    break;
                case 'u':
                    ul = va_arg(ap, unsigned int);
                    num2str(s, ul, 10, size, sp);
                    groveLCD_putString(s);
                    break;
                case 'o':
                    ul = va_arg(ap, unsigned int);
                    num2str(s, ul, 8, size, sp);
                    groveLCD_putString(s);
                    break;
                case 'p':
                    groveLCD_write('0');
                    groveLCD_write('x');
                    ul = va_arg(ap, unsigned int);
                    num2str(s, ul, 16, size, sp);
                    groveLCD_putString(s);
                    break;
                case 'x':
                    ul = va_arg(ap, unsigned int);
                    num2str(s, ul, 16, size, sp);
                    groveLCD_putString(s);
                    break;
                case 'f':
                    if(first==0){ ull = va_arg(ap, long long unsigned int); first = 1;}
                    ull = va_arg(ap, long long unsigned int);
                    int sign = ( ull & 0x80000000 ) » 31;
                    int m = (ull & 0x000FFFFF) ; // should be 0x007FFFFF
                    float mf = (float)m ;
                    mf = mf / pow(2.0,20.0);
                    mf = mf + 1.0;
                    int e = ( ull & 0x78000000 ) » 23 ; // should be int e = ( ul & 0x7F800000 ) » 23;
                    e = e | (( ull & 0x000F00000 ) » 20);
                    e = e - 127;
                    float f = mf*myPow(2.0,e);
                    if(sign==1){ groveLCD_write('-'); }
                    float2str((char*)s, f, 5);
                    groveLCD_putString((char*)s);
                    break;

                default:
                    groveLCD_write(*fmt);
            }
        } else groveLCD_write(*fmt);
        fmt++;
    }
    va_end(ap);
}
//===================================================================




```

5.28 C:/Users/Administrator/Documents/GitHub/ENIB_Robot_Mobile_↩ Ros2/WORKSPACE_F411_uROS6/base_robot/Core/Src/main.c File Reference

file that contain the main code

```
#include "main.h"
#include "motorCommand.h"
#include "quadEncoder.h"
#include "captDistIR.h"
#include "VL53L0X.h"
#include "groveLCD.h"
```

Data Structures

- struct AMessage
- struct MicroRosPubMsg
- struct MicroRosSubMsg

Macros

- #define SAMPLING_PERIOD_ms 5

config exo

- #define EXSTARTUP 0
- #define EXTEST_UART2 1
- #define EXCORRECTOR 2
- #define EXTESTCORRECTOR 3
- #define EXTEST_VL53 4
- #define EXTEST_MICROROS 5
- #define EXFINAL 6
- #define SYNCHRO_EX EXFINAL

config robot

- #define ROS_DOMAIN_ID 0
- #define LCD 0
- #define VL53 0
- #define MICROROS 1
- #define DEBUG_PRINTF 0
- #define DEBUG_MOTOR 0

config correcteur

- #define Te SAMPLING_PERIOD_ms
- #define LKp 0.001
- #define LKi (5.0/(0.1∗40.0))
- #define RKp 0.001
- #define RKi (5.0/(0.1∗40.0))

config default speed for each mode

- #define CMD 1000
- #define VITESSE_KART CMD/2
- #define VITESSE_OBS CMD
- #define VITESSE_CAM CMD/3

config camera settings

- #define CAMERA_X_MIN 0
- #define CAMERA_X_MAX 640
- #define CAMERA_Y_MIN 0
- #define CAMERA_Y_MAX 480

config default behaviour

- #define DEFAULT_MODE MODE_ZIG
- #define DEFAULT_SPEED LOW
- #define DEFAULT_DIR STOP

config test value

- #define NB 200
- #define TEST_CORRECTOR_DUTY 150
- #define TEST_CORRECTOR_SPEEDL -100
- #define TEST_CORRECTOR_SPEEDR -100
- #define TEST_LEFT_MOTOR 1

Enumerations

- enum { **MODE_OBS** , **MODE_ZIG** , **MODE_CAM** , **LAST_MODE** }
- enum {
  STOP_VIT , **LOW** , **FAST** , **SONIC** ,
  LAST_SPEED }
- enum {
  AVANT , **GAUCHE** , **RECULE** , **DROITE** ,
  STOP , **AVANT_GAUCHE** , **AVANT_DROITE** , **RECULE_GAUCHE** ,
  RECULE_DROITE , **LAST_DIR** }

Functions

- void SystemClock_Config (void)
- bool **cubemx_transport_open** (struct uxrCustomTransport ∗transport)
- bool **cubemx_transport_close** (struct uxrCustomTransport ∗transport)
- size_t **cubemx_transport_write** (struct uxrCustomTransport ∗transport, const uint8_t ∗buf, size_t len, uint8_t ∗err)
- size_t **cubemx_transport_read** (struct uxrCustomTransport ∗transport, uint8_t ∗buf, size_t len, int timeout, uint8_t ∗err)
- void ∗ microros_allocate (size_t size, void ∗state)
- void microros_deallocate (void ∗pointer, void ∗state)
- void ∗ microros_reallocate (void ∗pointer, size_t size, void ∗state)
- void ∗ microros_zero_allocate (size_t number_of_elements, size_t size_of_element, void ∗state)
- void CHECKMRRET (rcl_ret_t ret, char ∗msg)
- void SubscriberCallbackFunction (const void ∗msgin)
- void microros_task (void ∗argument)
- void task_Motor_Left (void ∗pvParameters)
- void task_Motor_Right (void ∗pvParameters)
- void task_Supervision (void ∗pvParameters)
- int main (void)
- void test_uart2 (void ∗pvParameters)
- void test_vl53 (void ∗pvParameters)
- void test_motor (void ∗pvParameters)
- void HAL_TIM_PeriodElapsedCallback (TIM_HandleTypeDef ∗htim)

  Period elapsed callback in non blocking mode.
- void Error_Handler (void)

Variables

- UART_HandleTypeDef huart1
- UART_HandleTypeDef huart2
- DMA_HandleTypeDef hdma_usart1_rx
- DMA_HandleTypeDef hdma_usart1_tx
- DMA_HandleTypeDef hdma_usart2_rx
- DMA_HandleTypeDef hdma_usart2_tx
- I2C_HandleTypeDef hi2c1
- osThreadId_t defaultTaskHandle
- const osThreadAttr_t defaultTask_attributes
- int16_t tab_speed [NB]

semaphore

- xSemaphoreHandle xSem_Supervision = NULL

queueHandle

- xQueueHandle q_mot_L = NULL
- xQueueHandle q_mot_R = NULL
- xQueueHandle qhMR_sub = NULL
- xQueueHandle qhMR_pub = NULL
- xQueueHandle qhLCD = NULL
- xQueueHandle qhVl53 = NULL

5.28.1 Detailed Description

file that contain the main code

Definition in file main.c.

5.28.2 Macro Definition Documentation

5.28.2.1 CAMERA_X_MAX

```
#define CAMERA_X_MAX 640
```

Define maximal x position return by camera

Definition at line 72 of file main.c.

5.28.2.2 CAMERA_X_MIN

```
#define CAMERA_X_MIN 0
```

Define minimal x position return by camera

Definition at line 71 of file main.c.

5.28.2.3 CAMERA_Y_MAX

```
#define CAMERA_Y_MAX 480
```

Define maximal y position return by camera

Definition at line 74 of file main.c.

5.28.2.4 CAMERA_Y_MIN

```
#define CAMERA_Y_MIN 0
```

Define minimal y position return by camera

Definition at line 73 of file main.c.

5.28.2.5 CMD

```
#define CMD 1000
```

Can be use as default speed

Definition at line 65 of file main.c.

5.28.2.6 DEBUG_MOTOR

```
#define DEBUG_MOTOR 0
```

Activate motor debug print

Definition at line 55 of file main.c.

5.28.2.7 DEBUG_PRINTF

```
#define DEBUG_PRINTF 0
```

Activate debug print

Definition at line 54 of file main.c.

5.28.2.8 DEFAULT_DIR

```
#define DEFAULT_DIR STOP
```

Default direction at startup

Definition at line 81 of file main.c.

5.28.2.9 DEFAULT_MODE

```
#define DEFAULT_MODE MODE_ZIG
```

Default mode at startup

Definition at line 79 of file main.c.

5.28.2.10 DEFAULT_SPEED

```
#define DEFAULT_SPEED LOW
```

Default speed at startup

Definition at line 80 of file main.c.

5.28.2.11 EXCORRECTOR

```
#define EXCORRECTOR 2
```

Code to calibrate your correcteur

Definition at line 41 of file main.c.

5.28.2.12 EXFINAL

```
#define EXFINAL 6
```

Final code

Definition at line 45 of file main.c.

5.28.2.13 EXSTARTUP

```
#define EXSTARTUP 0
```

startup code

Definition at line 39 of file main.c.

5.28.2.14 EXTEST_MICROROS

```
#define EXTEST_MICROROS 5
```

Test Micro ROS subscriber and publisher

Definition at line 44 of file main.c.

5.28.2.15 EXTEST_UART2

```
#define EXTEST_UART2 1
```

Test printf and scanf function

Definition at line 40 of file main.c.

5.28.2.16 EXTEST_VL53

```
#define EXTEST_VL53 4
```

Test VL530X sensor

Definition at line 43 of file main.c.

5.28.2.17 EXTESTCORRECTOR

```
#define EXTESTCORRECTOR 3
```

Code to test your correcteur

Definition at line 42 of file main.c.

5.28.2.18 LCD

```
#define LCD 0
```

Activate LCD task

Definition at line 51 of file main.c.

5.28.2.19 LKi

```
#define LKi (5.0/(0.1*40.0))
```

Ki factor for the left motor

Definition at line 60 of file main.c.

5.28.2.20 LKp

```
#define LKp 0.001
```

Kp factor for the left motor

Definition at line 59 of file main.c.

5.28.2.21 MICROROS

```
#define MICROROS 1
```

Activate MicroROS task

Definition at line 53 of file main.c.

5.28.2.22 NB

```
#define NB 200
```

Number of samples in correcteur calibration task

Definition at line 84 of file main.c.

5.28.2.23 RKi

```
#define RKi (5.0/(0.1*40.0))
```

Kp factor for the right motor

Definition at line 62 of file main.c.

5.28.2.24 RKp

```
#define RKp 0.001
```

Kp factor for the right motor

Definition at line 61 of file main.c.

5.28.2.25 ROS_DOMAIN_ID

```
#define ROS_DOMAIN_ID 0
```

Define ROS domain id

Definition at line 50 of file main.c.

5.28.2.26 SAMPLING_PERIOD_ms

```
#define SAMPLING_PERIOD_ms 5
```

Define the delay beetween two execution of the same task

Definition at line 37 of file main.c.

5.28.2.27 SYNCHRO_EX

```
#define SYNCHRO_EX EXFINAL
```

Define wich config are executed

Definition at line 47 of file main.c.

5.28.2.28 Te

```
#define Te SAMPLING_PERIOD_ms
```

Definition at line 58 of file main.c.

5.28.2.29 TEST_CORRECTOR_DUTY

```
#define TEST_CORRECTOR_DUTY 150
```

Duty cycle to apply to calibrate the correcteur

Definition at line 85 of file main.c.

5.28.2.30 TEST_CORRECTOR_SPEEDL

```
#define TEST_CORRECTOR_SPEEDL -100
```

Speed to test the left correcteur

Definition at line 86 of file main.c.

5.28.2.31 TEST_CORRECTOR_SPEEDR

```
#define TEST_CORRECTOR_SPEEDR -100
```

Speed to test the right correcteur

Definition at line 87 of file main.c.

5.28.2.32 TEST_LEFT_MOTOR

```
#define TEST_LEFT_MOTOR 1
```

Calibrate left motor correcteur or not

Definition at line 88 of file main.c.

5.28.2.33 VITESSE_CAM

```
#define VITESSE_CAM CMD/3
```

Default speed for camera mode

Definition at line 68 of file main.c.

5.28.2.34 VITESSE_KART

```
#define VITESSE_KART CMD/2
```

Default speed for manual mode

Definition at line 66 of file main.c.

5.28.2.35 VITESSE_OBS

```
#define VITESSE_OBS CMD
```

Default speed for obstacle mode

Definition at line 67 of file main.c.

5.28.2.36 VL53

```
#define VL53 0
```

Activate VL530X task

Definition at line 52 of file main.c.

5.28.3 Enumeration Type Documentation

5.28.3.1 anonymous enum

```
anonymous enum
```

enumerate mode of robot

Definition at line 31 of file main.c.

5.28.3.2 anonymous enum

```
anonymous enum
```

enumerate speed

Definition at line 33 of file main.c.

5.28.3.3 anonymous enum

```
anonymous enum
```

enumerate direction

Definition at line 35 of file main.c.

5.28.4 Function Documentation

5.28.4.1 CHECKMRRET()

```
void CHECKMRRET (
             rcl_ret_t ret,
             char * msg )
```

check if microRos function success else print msg in console

Parameters

ret	return value of microRos function
msg	message to print if fail

Definition at line 153 of file main.c.

5.28.4.2 Error_Handler()

```
void Error_Handler (
             void  )
```

Definition at line 941 of file main.c.

5.28.4.3 HAL_TIM_PeriodElapsedCallback()

```
void HAL_TIM_PeriodElapsedCallback (
             TIM_HandleTypeDef * htim )
```

Period elapsed callback in non blocking mode.

Note

> This function is called when TIM1 interrupt took place, inside HAL_TIM_IRQHandler(). It makes a direct call to HAL_IncTick() to increment a global variable "uwTick" used as application time base.

Parameters

htim	: TIM handle

Return values

None	

Definition at line 933 of file main.c.

5.28.4.4 main()

```
int main (
             void  )
```

Init all GPIO and drivers, start the task, init semaphore and queue and launch the kernel

- Config EXSTARTUP
  - Launch microRos, supervision, left motor, right motor and lcd task
- Config EXTEST_UART2

  - Launch test_uart2 task
- Config EXCORRECTOR
  - Launch test_motor task
- Config EXTESTCORRECTOR
  - Launch supervision, left motor and right motor task
- Config EXTEST_VL53
  - Launch test_vl53 task
- Config EXTEST_MICROROS
  - Launch microRos task
- Config EXFINAL
  - Launch microRos, supervision, left motor, right motor, vl53 and lcd task

Definition at line 775 of file main.c.

5.28.4.5 microros_allocate()

```
void * microros_allocate (
             size_t size,
             void * state )
```

Definition at line 14 of file microros_allocators.c.

5.28.4.6 microros_deallocate()

```
void microros_deallocate (
             void * pointer,
             void * state )
```

Definition at line 22 of file microros_allocators.c.

5.28.4.7 microros_reallocate()

```
void * microros_reallocate (
             void * pointer,
             size_t size,
             void * state )
```

Definition at line 31 of file microros_allocators.c.

5.28.4.8 microros_task()

```
void microros_task (
            void * argument )
```

- All config
  - Create the node *STM32_node*
  - Set the Domain id of microRos
- Config EXSTARTUP :
  - Create a publisher and send a message on it
- Config EXTEST_MICROROS :
  - Create a publisher, a subscriber and an executor
  - Init the executor and add the subscriber to it
  - Run the executor and send the receive message on the publisher
- Config EXFINAL :
  - Create 3 publishers, 5 subscriber and an executor
  - Init the executor and add the 5 subscribers to it
  - run the executor and if they are no elements waiting to be read by the task decision put the receive information in the queue If decison task send data then publish data to microRos

Parameters

argument	

Definition at line 166 of file main.c.

5.28.4.9 microros_zero_allocate()

```
void * microros_zero_allocate (
            size_t number_of_elements,
            size_t size_of_element,
            void * state )
```

Definition at line 44 of file microros_allocators.c.

5.28.4.10 SubscriberCallbackFunction()

```
void SubscriberCallbackFunction (
            const void * msgin )
```

callback call by microros when a message is receive here use as debug and just print the receive msg

Parameters

message	receive

Definition at line 155 of file main.c.

5.28.4.11 SystemClock_Config()

```
void SystemClock_Config (
             void  )
```

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

Definition at line 11 of file systemclock.c.

5.28.4.12 task_Motor_Left()

```
void task_Motor_Left (
             void * pvParameters )
```

Task use to control the left motor of the robot

Parameters

argument	

Definition at line 365 of file main.c.

5.28.4.13 task_Motor_Right()

```
void task_Motor_Right (
             void * pvParameters )
```

Task use to control the right motor of the robot

Parameters

argument	

Definition at line 391 of file main.c.

5.28.4.14 task_Supervision()

```
void task_Supervision (
             void * pvParameters )
```

Brain of the robot. get information for MicroRos and VL53 task, then send speed to left and right motor, lcd and microRos task

- Config EXSTARTUP :
  - Make the robot drive forward until an obstacle are found
- Config EXTESTCORRECTOR :
  - Make the robot drive forward at speed set by config
- Config EXFINAL :
  - Make robot switch beetween 3 behaviour depending of the mode
  - Obstacle : drive and avoid obstacles
  - Manual : drive in direction set in ihm
  - Camera : follow an object

Parameters

argument	

Definition at line 476 of file main.c.

5.28.4.15 test_motor()

```
void test_motor (
             void * pvParameters )
```

Use to set the duty cycle and register the motor speed at each Te

Definition at line 895 of file main.c.

5.28.4.16 test_uart2()

```
void test_uart2 (
             void * pvParameters )
```

Use to test printf and scanf function

Definition at line 872 of file main.c.

5.28.4.17 test_vl53()

```
void test_vl53 (
             void * pvParameters )
```

Use to test the VL53 sensor

Definition at line 884 of file main.c.

5.28.5 Variable Documentation

5.28.5.1 defaultTask_attributes

```
const osThreadAttr_t defaultTask_attributes
```

Initial value:

```
= {
  .name = "defaultTask",
  .stack_size = 3000 * 4,
  .priority = (osPriority_t) osPriorityNormal,
}
```

Definition at line 24 of file main.c.

5.28.5.2 defaultTaskHandle

```
osThreadId_t defaultTaskHandle
```

Definition at line 23 of file main.c.

5.28.5.3 hdma_usart1_rx

```
DMA_HandleTypeDef hdma_usart1_rx  [extern]
```

Definition at line 6 of file drv_uart.c.

5.28.5.4 hdma_usart1_tx

```
DMA_HandleTypeDef hdma_usart1_tx  [extern]
```

Definition at line 7 of file drv_uart.c.

5.28.5.5 hdma_usart2_rx

```
DMA_HandleTypeDef hdma_usart2_rx  [extern]
```

Definition at line 8 of file drv_uart.c.

5.28.5.6 hdma_usart2_tx

```
DMA_HandleTypeDef hdma_usart2_tx  [extern]
```

Definition at line 9 of file drv_uart.c.

5.28.5.7 hi2c1

```
I2C_HandleTypeDef hi2c1  [extern]
```

Definition at line 6 of file drv_i2c.c.

5.28.5.8 huart1

```
UART_HandleTypeDef huart1  [extern]
```

Definition at line 4 of file drv_uart.c.

5.28.5.9 huart2

```
UART_HandleTypeDef huart2  [extern]
```

Definition at line 5 of file drv_uart.c.

5.28.5.10 q_mot_L

```
xQueueHandle q_mot_L = NULL
```

Queue to communicate with left motor task

Definition at line 96 of file main.c.

5.28.5.11 q_mot_R

```
xQueueHandle q_mot_R = NULL
```

Queue to communicate with right motor task

Definition at line 97 of file main.c.

5.28.5.12 qhLCD

```
xQueueHandle qhLCD = NULL
```

Queue to communicate with LCD task

Definition at line 100 of file main.c.

5.28.5.13 qhMR_pub

```
xQueueHandle qhMR_pub = NULL
```

Queue to communicate with microRos task

Definition at line 99 of file main.c.

5.28.5.14 qhMR_sub

```
xQueueHandle qhMR_sub = NULL
```

Queue to get information from microRos task

Definition at line 98 of file main.c.

5.28.5.15 qhVl53

```
xQueueHandle qhVl53 = NULL
```

Queue to communicate with VL53 task

Definition at line 101 of file main.c.

5.28.5.16 tab_speed

```
int16_t tab_speed[NB]
```

use to store speed of motor during calibration of the correcteur

Definition at line 104 of file main.c.

5.28.5.17 xSem_Supervision

```
xSemaphoreHandle xSem_Supervision = NULL
```

Semaphore use in decision task

Definition at line 93 of file main.c.

5.29 main.c

Go to the documentation of this file.

```
00001
00005 #include "main.h"
00006
00007 #include "motorCommand.h"
00008 #include "quadEncoder.h"
00009 #include "captDistIR.h"
00010 #include "VL53L0X.h"
00011 #include "groveLCD.h"
00012
00013 extern UART_HandleTypeDef huart1;
00014 extern UART_HandleTypeDef huart2;
00015 extern DMA_HandleTypeDef hdma_usart1_rx;
00016 extern DMA_HandleTypeDef hdma_usart1_tx;
00017 extern DMA_HandleTypeDef hdma_usart2_rx;
00018 extern DMA_HandleTypeDef hdma_usart2_tx;
00019
00020 extern I2C_HandleTypeDef hi2c1;
00021
00022 /* Definitions for defaultTask */
00023 osThreadId_t defaultTaskHandle;
00024 const osThreadAttr_t defaultTask_attributes = {
00025   .name = "defaultTask",
00026   .stack_size = 3000 * 4,
  .priority = (osPriority_t) osPriorityNormal,
};

enum {MODE_OBS, MODE_ZIG, MODE_CAM, LAST_MODE};
enum {STOP_VIT, LOW, FAST, SONIC, LAST_SPEED};
enum {AVANT, GAUCHE, RECULE, DROITE, STOP, AVANT_GAUCHE, AVANT_DROITE, RECULE_GAUCHE, RECULE_DROITE, LAST_DIR};

#define SAMPLING_PERIOD_ms 5
#define EXSTARTUP 0
#define EXTEST_UART2 1
#define EXCORRECTOR 2
#define EXTESTCORRECTOR 3
#define EXTEST_VL53 4
#define EXTEST_MICROROS 5
#define EXFINAL 6
#define SYNCHRO_EX EXFINAL
#define ROS_DOMAIN_ID 0
#define LCD 0
#define VL53 0
#define MICROROS 1
#define DEBUG_PRINTF 0
#define DEBUG_MOTOR 0
#define Te SAMPLING_PERIOD_ms
#define LKp 0.001
#define LKi (5.0/(0.1*40.0))
#define RKp 0.001
#define RKi (5.0/(0.1*40.0))
#define CMD 1000
#define VITESSE_KART CMD/2
#define VITESSE_OBS CMD
#define VITESSE_CAM CMD/3
#define CAMERA_X_MIN 0
#define CAMERA_X_MAX 640
#define CAMERA_Y_MIN 0
#define CAMERA_Y_MAX 480
//#define CAMERA_X_TIER (CAMERA_X_MAX-CAMERA_X_MIN)/3 /**< */
//#define CAMERA_Y_TIER (CAMERA_Y_MAX-CAMERA_Y_MIN)/3 /**< */
#define DEFAULT_MODE MODE_ZIG
#define DEFAULT_SPEED LOW
#define DEFAULT_DIR STOP
#define NB 200
#define TEST_CORRECTOR_DUTY 150
#define TEST_CORRECTOR_SPEEDL -100
#define TEST_CORRECTOR_SPEEDR -100
#define TEST_LEFT_MOTOR 1
// Déclaration des objets synchronisants !! Ne pas oublier de les créer
xSemaphoreHandle xSem_Supervision = NULL;
xQueueHandle q_mot_L = NULL;
xQueueHandle q_mot_R = NULL;
xQueueHandle qhMR_sub = NULL;
xQueueHandle qhMR_pub = NULL;
xQueueHandle qhLCD = NULL;
xQueueHandle qhVl53 = NULL;
int16_t tab_speed[NB];
typedef struct
{
    char command;
    int data;
} AMessage;

typedef struct
{
    char dir;
    int mode;
    int speed;
} MicroRosPubMsg;

typedef struct
{
    int dir;
    int x;
    int y;
    int mode;
    int speed;
} MicroRosSubMsg;
//Robot function
void SystemClock_Config(void);

//Micro-Ros function
bool cubemx_transport_open(struct uxrCustomTransport * transport);
bool cubemx_transport_close(struct uxrCustomTransport * transport);
size_t cubemx_transport_write(struct uxrCustomTransport* transport, const uint8_t * buf, size_t len, uint8_t * err);
size_t cubemx_transport_read(struct uxrCustomTransport* transport, uint8_t* buf, size_t len, int timeout, uint8_t* err);

```

```
void * microros_allocate(size_t size, void * state);
void microros_deallocate(void * pointer, void * state);
void * microros_reallocate(void * pointer, size_t size, void * state);
void * microros_zero_allocate(size_t number_of_elements, size_t size_of_element, void * state);

void CHECKMRRET(rcl_ret_t ret, char* msg){if (ret != RCL_RET_OK){ printf("Error : %d\r\nMsg : %s\r\n", (int)ret, msg); }}

void SubscriberCallbackFunction(const void *msgin){
#if SYNCHRO_EX == EXTEST_MICROROS
    std_msgs__msg__String * msg = (std_msgs__msg__String * )msgin;
    printf("\r\nMessage recue : %s\r\n", msg->data->data);
#elif SYNCHRO_EX == EXFINAL
    std_msgs__msg__Int32 * msg = (std_msgs__msg__Int32 * )msgin;
    printf("\r\nMessage recue : %ld\r\n", msg->data);
#endif //SYNCHRO_EX
}

// https://github.com/lFatality/stm32_micro_ros_setup
void microros_task(void *argument)
{
    // micro-ROS app variable
    rclc_support_t support; //Contain information about
    rcl_allocator_t allocator;
    rcl_node_t node; //microRos structure wich represent a node ROS
    rcl_node_options_t node_opt; //microRos structure wich represent option of a node ROS
    rclc_executor_t executor; //The executor is use to receive message

    // micro-ROS configuration
    rmw_uros_set_custom_transport(
        true,
        (void *) &huart1,
        cubemx_transport_open,
        cubemx_transport_close,
        cubemx_transport_write,
        cubemx_transport_read);

    rcl_allocator_t freeRTOS_allocator = rcutils_get_zero_initialized_allocator();
    freeRTOS_allocator.allocate = microros_allocate;
    freeRTOS_allocator.deallocate = microros_deallocate;
    freeRTOS_allocator.reallocate = microros_reallocate;
    freeRTOS_allocator.zero_allocate =  microros_zero_allocate;

    if (!rcutils_set_default_allocator(&freeRTOS_allocator)) {
        printf("Error on default allocators (line %d)\r\n", __LINE__);
    }

    allocator = rcl_get_default_allocator();
    //create init_options
    CHECKMRRET(rclc_support_init(&support, 0, NULL, &allocator), "error on init support");
    // create node
    //CHECKMRRET(rclc_node_init_default(&node, "STM32_node", "", &support), "error on init node");
    node_opt = rcl_node_get_default_options();
    node_opt.domain_id = ROS_DOMAIN_ID;
    CHECKMRRET(rclc_node_init_with_options(&node, "STM32_node", "", &support, &node_opt), "error on init node");


#if SYNCHRO_EX == EXSTARTUP
    static int counter = 0;
    rcl_ret_t ret;
    rcl_publisher_t publisher;
    std_msgs__msg__String msg;

    CHECKMRRET(rclc_publisher_init_default(&publisher, &node, ROSIDL_GET_MSG_TYPE_SUPPORT(std_msgs, msg, String), "cubemx_publisher"),
            "Error when create publisher");

    for (;;)
    {
        sprintf(msg.data.data, "Hello from micro-ROS #%d", counter++);
        msg.data.size = strlen(msg.data.data);
        ret = rcl_publish(&publisher, &msg, NULL);
        if (ret != RCL_RET_OK)
            printf("Error publishing (line %d)\r\n", __LINE__);
        vTaskDelay(SAMPLING_PERIOD_ms);
    }
#elif SYNCHRO_EX == EXTEST_MICROROS
    //micro-ros topic variable
    rcl_ret_t ret;
    rcl_publisher_t publisher;
    rcl_subscription_t subscriber;
    std_msgs__msg__String msg;

    /* Default publisher */
    CHECKMRRET(rclc_publisher_init_default(&publisher, &node, ROSIDL_GET_MSG_TYPE_SUPPORT(std_msgs, msg, String), "cubemx_publisher"),
            "Error when create publisher");
        /* ---------------------- */
    /* Default subscriber */
    subscriber = rcl_get_zero_initialized_subscription();
    CHECKMRRET(rclc_subscription_init_default(&subscriber, &node, ROSIDL_GET_MSG_TYPE_SUPPORT(std_msgs, msg, String),"cubemx_subscriber"),
            "Error when create subscriber");
    /* ---------------------- */
    /* Init string msg */
    msg.data.data = (char * ) malloc(ARRAY_LEN * sizeof(char));
    msg.data.size = 0;
    msg.data.capacity = ARRAY_LEN;
    /* --------------- */
    // Init executor and add subscriber to it
    CHECKMRRET(rclc_executor_init(&executor, &support.context, 1, &allocator), "Error on init executor");
    CHECKMRRET(rclc_executor_add_subscription(&executor, &subscriber, &msg, &SubscriberCallbackFunction, ON_NEW_DATA), "error add subscriber");

    for (;;)
    {
        ret = rclc_executor_spin_some(&executor, 100*1000*1000);
        vTaskDelay(SAMPLING_PERIOD_ms);
    }
#elif SYNCHRO_EX == EXFINAL
    //Init the queue mesage
    MicroRosPubMsg MsgToPub = {'N', 0, 0};
    MicroRosSubMsg SubToMsg = {DEFAULT_DIR, 0, 0, DEFAULT_MODE, DEFAULT_SPEED};
    /* PUBLISHER */
    //Use to publish the direction of robot in sensor mode
    rcl_publisher_t capteur_dir_pub;
    char* capteur_dir_topic = CAPTEUR_DIR_TOPIC;
    std_msgs__msg__Int32 capteur_dir_msg;
    //Use to publish the actual mode of the robot
    rcl_publisher_t etat_mode_pub;
    char* etat_mode_topic = ETAT_MODE_TOPIC;
    std_msgs__msg__Int32 etat_mode_msg;
    //Use to publish the actual speed of the robot
    rcl_publisher_t etat_speed_pub;
    char* etat_speed_topic = ETAT_SPEED_TOPIC;
    std_msgs__msg__Int32 etat_speed_msg;
    /* SUBSCRIBER */
    //Use to receive the x position of object see by the camera
    rcl_subscription_t camera_x_sub;
    char* camera_x_topic = CAMERA_X_TOPIC;
    std_msgs__msg__Int32 camera_x_msg;
    //Use to receive the y position of object see by the camera
    rcl_subscription_t camera_y_sub;
    char* camera_y_topic = CAMERA_Y_TOPIC;
    std_msgs__msg__Int32 camera_y_msg;
    //Use to receive the remote control in remote mode
    rcl_subscription_t telecommande_dir_sub;
    char* telecommande_dir_topic = TELECOMMANDE_DIR_TOPIC;
    std_msgs__msg__Int32 telecommande_dir_msg;
    //Use to receive the mode config
    rcl_subscription_t config_mode_sub;
    char* config_mode_topic = CONFIG_MODE_TOPIC;
    std_msgs__msg__Int32 config_mode_msg;
    //Use to receive the speed config
    rcl_subscription_t config_speed_sub;
    char* config_speed_topic = CONFIG_SPEED_TOPIC;
    std_msgs__msg__Int32 config_speed_msg;

    // create publisher
    createPublisher(&capteur_dir_pub, &node,
        ROSIDL_GET_MSG_TYPE_SUPPORT(std_msgs, msg, Int32),
        capteur_dir_topic, &capteur_dir_msg);

    createPublisher(&etat_mode_pub, &node,
        ROSIDL_GET_MSG_TYPE_SUPPORT(std_msgs, msg, Int32),
        etat_mode_topic, &etat_mode_msg);

    createPublisher(&etat_speed_pub, &node,
        ROSIDL_GET_MSG_TYPE_SUPPORT(std_msgs, msg, Int32),
        etat_speed_topic, &etat_speed_msg);

    //create subscriber
    createSubscriber(&camera_x_sub, &node,
        ROSIDL_GET_MSG_TYPE_SUPPORT(std_msgs, msg, Int32),
        camera_x_topic, &camera_x_msg);

    createSubscriber(&camera_y_sub, &node,
            ROSIDL_GET_MSG_TYPE_SUPPORT(std_msgs, msg, Int32),
            camera_y_topic, &camera_y_msg);

    createSubscriber(&telecommande_dir_sub, &node,
            ROSIDL_GET_MSG_TYPE_SUPPORT(std_msgs, msg, Int32),
            telecommande_dir_topic, &telecommande_dir_msg);

    createSubscriber(&config_mode_sub, &node,
            ROSIDL_GET_MSG_TYPE_SUPPORT(std_msgs, msg, Int32),
            config_mode_topic, &config_mode_msg);

    createSubscriber(&config_speed_sub, &node,
            ROSIDL_GET_MSG_TYPE_SUPPORT(std_msgs, msg, Int32),
            config_speed_topic, &config_speed_msg);

    //Init the executor
    CHECKMRRET(rclc_executor_init(&executor, &support.context, 5, &allocator), "Error on init executor");
    /*Add subscriber to executor to let it check if message is receive on this
    topic and store the data on the message structure after call the callback*/
    CHECKMRRET(rclc_executor_add_subscription(&executor, &camera_x_sub, &camera_x_msg, &SubscriberCallbackFunction, ON_NEW_DATA), "error add camera_x_sub");
    CHECKMRRET(rclc_executor_add_subscription(&executor, &camera_y_sub, &camera_y_msg, &SubscriberCallbackFunction, ON_NEW_DATA), "error add camera_y_sub");
    CHECKMRRET(rclc_executor_add_subscription(&executor, &telecommande_dir_sub, &telecommande_dir_msg, &SubscriberCallbackFunction, ON_NEW_DATA), "error add telecommande_dir_sub");
    CHECKMRRET(rclc_executor_add_subscription(&executor, &config_mode_sub, &config_mode_msg, &SubscriberCallbackFunction, ON_NEW_DATA), "error add config_mode_sub");
    CHECKMRRET(rclc_executor_add_subscription(&executor, &config_speed_sub, &config_speed_msg, &SubscriberCallbackFunction, ON_NEW_DATA), "error add config_speed_sub");

    for(;;)
    {
        if (!uxQueueMessagesWaiting(qhMR_sub)) //If no message in 'output' queue
            xQueueSend(qhMR_sub, ( void * ) &SubToMsg, portMAX_DELAY);
        rclc_executor_spin_some(&executor, 1*1000*1000); //Execute executor
        SubToMsg.dir = telecommande_dir_msg.data;
        SubToMsg.x = camera_x_msg.data;
        SubToMsg.y = camera_y_msg.data;
        SubToMsg.mode = config_mode_msg.data;
        SubToMsg.speed = config_speed_msg.data;

        if (uxQueueMessagesWaiting(qhMR_pub)) //If no message in 'input' queue
        {
            xQueueReceive(qhMR_pub, &MsgToPub, portMAX_DELAY); //Receive data
            capteur_dir_msg.data = (int)MsgToPub.dir;
            etat_mode_msg.data = MsgToPub.mode;
            etat_speed_msg.data = MsgToPub.speed;
            //Publish data
            CHECKMRRET(rcl_publish(&capteur_dir_pub, &capteur_dir_msg, NULL), "erreur publish capteur_dir_pub");
            CHECKMRRET(rcl_publish(&etat_mode_pub, &etat_mode_msg, NULL), "erreur publish etat_mode_pub");
            CHECKMRRET(rcl_publish(&etat_speed_pub, &etat_speed_msg, NULL), "erreur publish etat_speed_pub");
            #if DEBUG_PRINTF
            printf("\r\nReceive from decision :\r\nDirection : %d\r\nMode : %d\r\nSpeed : %d\r\n", capteur_dir_msg.data, etat_mode_msg.data, etat_speed_msg.data);
            #endif //DEBUG_PRINTF
        }
        vTaskDelay(SAMPLING_PERIOD_ms);
    }
#endif //SYNCHRO_EX
}

void task_Motor_Left(void *pvParameters)
{
    int16_t consigne = 0; //Store the desirate speed

    float ui = 0.0; //Integral term of the correcteur
    float up = 0.0; //Proportionnal term of the correcteur
    int err = 0; //Error term of the correcteur
    int speed = 0; //Actual speed of motor

    for (;;)
    {
        xQueueReceive(q_mot_L, &consigne, portMAX_DELAY); //receive wanted speed

        speed = quadEncoder_GetSpeedL(); //Get actual speed
        //Calculate term of correcteur
        err=consigne-speed;
        up=LKp*(float)err;
        ui=ui+LKp*LKi*(float)err;

        motorLeft_SetDuty(100+(int)(up+ui)); //Set duty cycle of the motor

        xSemaphoreGive(xSem_Supervision); //Give semaphore to liberate the decision task
        vTaskDelay(SAMPLING_PERIOD_ms);
    }
}

void task_Motor_Right(void *pvParameters)
```

```
{
    int16_t consigne = 0; //Store the desirate speed

    float ui = 0.0; //Integral term of the correcteur
    float up = 0.0; //Proportionnal term of the correcteur
    int err = 0; //Error term of the correcteur
    int speed = 0; //Actual speed of motor

    for (;;)
    {
        xQueueReceive(q_mot_R, &consigne, portMAX_DELAY); //receive wanted speed

        speed = quadEncoder_GetSpeedR(); //Get actual speed
        //Calculate term of correcteur
        err=consigne-speed;
        up=RKp*(float)err;
        ui=ui+RKp*RKi*(float)err;

        motorRight_SetDuty(100+(int)(up+ui)); //Set duty cycle of the motor

        xSemaphoreGive(xSem_Supervision);//Give semaphore to liberate the decision task
        vTaskDelay(SAMPLING_PERIOD_ms);
    }
}

#if VL53
void task_VL53(void *pvParameters)
{
    static uint16_t dist;
    static const int SEUIL = 20;
    int obs = 0;

    for(;;)
    {
        dist = readRangeSingleMillimeters()/10;
        if (dist < SEUIL && dist != 0)
            obs = 1;
        else
            obs = 0;

        if (!uxQueueMessagesWaiting(qhVl53))
            xQueueSend(qhVl53, (void *)&obs, portMAX_DELAY);

        vTaskDelay(SAMPLING_PERIOD_ms);
    }
}
#endif //VL53


#if LCD
void task_Grove_LCD(void *pvParameters)
{
#if SYNCHRO_EX == EXSTARTUP
    for (;;)
    {
         groveLCD_setCursor(0,0);
         groveLCD_term_printf("TEST LCD");
         vTaskDelay(100);
    }
#elif SYNCHRO_EX == EXFINAL
    AMessage pxRxedMessage;

    for(;;)
    {
        if (uxQueueMessagesWaiting(qhLCD))
        {
            xQueueReceive(qhLCD, &pxRxedMessage, portMAX_DELAY);
            int mode = pxRxedMessage.data;
            char direction=pxRxedMessage.command;
            groveLCD_setCursor(0,0);
            if (mode == MODE_OBS)
                groveLCD_term_printf("M:Obstacle  D:%c", direction);
            else if (mode == MODE_ZIG)
                groveLCD_term_printf("M:Manuel        ");
            else if (mode == MODE_CAM)
                groveLCD_term_printf("M:Camera        ");
        }

        vTaskDelay(SAMPLING_PERIOD_ms);
    }
#endif //SYNCHRO_EX
}
#endif //LCD

void task_Supervision(void *pvParameters)
{
#if SYNCHRO_EX == EXSTARTUP
```

```
    int16_t consigne_G=0;
    int16_t consigne_D=0;

    int tab_mes_ir[2];
    uint16_t mes_vl53=0;

    vTaskDelay(100);
    for (;;)
    {
        captDistIR_Get(tab_mes_ir);
        //mes_vl53 = readRangeSingleMillimeters()/10;

        if((tab_mes_ir[0]>2000)||(tab_mes_ir[1]>2000))
        { // !! obstacle
            consigne_G=0;
            consigne_D=0;
        }
        else
        {
            consigne_G=1000;
            consigne_D=1000;
        }

        xQueueSend( q_mot_L, ( void * ) &consigne_G,  portMAX_DELAY );
        xSemaphoreTake( xSem_Supervision, portMAX_DELAY );

        xQueueSend( q_mot_R, ( void * ) &consigne_D,  portMAX_DELAY );
        xSemaphoreTake( xSem_Supervision, portMAX_DELAY );

        vTaskDelay(SAMPLING_PERIOD_ms);
    }
#elif SYNCHRO_EX == EXTESTCORRECTOR
    int16_t speedLeft = TEST_CORRECTOR_SPEEDL;
    int16_t speedRight = TEST_CORRECTOR_SPEEDR;

    for (;;)
    {
        xQueueSend(q_mot_L, (void *)&speedLeft, portMAX_DELAY);
        xSemaphoreTake(xSem_Supervision, portMAX_DELAY);

        xQueueSend(q_mot_R, (void *)&speedRight, portMAX_DELAY);
        xSemaphoreTake(xSem_Supervision, portMAX_DELAY);

        vTaskDelay(SAMPLING_PERIOD_ms);
    }
#elif SYNCHRO_EX == EXFINAL
    int16_t speedLeft;
    int16_t speedRight;

    int table[2];
    #if VL53
    int vl53 = 0;
    #endif //VL53

    static int obs = 0;
    static char dir = 'f';
    static int direction = DEFAULT_DIR;
    static int speed = DEFAULT_SPEED;
    static int mode = DEFAULT_MODE;
    static int x = 0;
    static int y = 0;

#if LCD
     AMessage pxMessage;
#endif

#if MICROROS
    MicroRosSubMsg SubToMsg;
    MicroRosPubMsg MsgToPub;
#endif //MICROROS

    for (;;)
    {
        #if MICROROS
        if (uxQueueMessagesWaiting(qhMR_sub))
        {
            xQueueReceive(qhMR_sub, &SubToMsg, portMAX_DELAY);
            if (SubToMsg.mode >= 0 && SubToMsg.mode < LAST_MODE)
                mode = SubToMsg.mode;
            if (SubToMsg.dir >= 0 && SubToMsg.dir < LAST_DIR)
                direction = SubToMsg.dir;
            if (SubToMsg.speed > 0 && SubToMsg.speed < LAST_SPEED)
                speed = SubToMsg.speed;
            x = SubToMsg.x;
            y = SubToMsg.y;
            #if DEBUG_PRINTF
            printf("%cc%c[2J%c[0;0HVariable to make decision : \n\rDirection : %d\r\nMode: %d\r\nSpeed
: %d\r\nX: %d\r\nY : %d\r\n", 0x1b, 0x1b, 0x1b, direction, mode, speed, x, y);
            #endif //DEBUG_PRINTF
        }
        #endif //MICROROS

        if (mode == MODE_ZIG)
        {
            dir = 'N';
            obs = 0;
            switch(direction)
            {
                case STOP:
                    speedLeft = 0;
                    speedRight = 0;
                    break;
                case AVANT:
                    speedLeft = VITESSE_KART*speed;
                    speedRight = VITESSE_KART*speed;
                    break;
                case RECULE:
                    speedLeft = -VITESSE_KART*speed;
                    speedRight = -VITESSE_KART*speed;
                    break;
                case DROITE:
                    speedLeft = VITESSE_KART*speed;
                    speedRight = -VITESSE_KART*speed;
                    break;
                case GAUCHE:
                    speedLeft = -VITESSE_KART*speed;
                    speedRight = VITESSE_KART*speed;
                    break;
                case AVANT_GAUCHE:
                    speedLeft = (VITESSE_KART/2)*speed;
                    speedRight = VITESSE_KART*speed;
                    break;
                case AVANT_DROITE:
                    speedLeft = VITESSE_KART*speed;
                    speedRight = (VITESSE_KART/2)*speed;
                    break;
                case RECULE_GAUCHE:
                    speedLeft = -VITESSE_KART*speed;
                    speedRight = (-VITESSE_KART/2)*speed;
                    break;
                case RECULE_DROITE:
                    speedLeft = (-VITESSE_KART/2)*speed;
                    speedRight = -VITESSE_KART*speed;
                    break;
                default:
                    speedLeft = 0;
                    speedRight = 0;
                    break;
            }
        }
        else if (mode == MODE_OBS)
        {
            captDistIR_Get(table);
            #if VL53
            if (uxQueueMessagesWaiting(qhVl53))
                xQueueReceive(qhVl53, &vl53, portMAX_DELAY);
            else
                vl53 = 0;

            if (vl53 == 1) //Il y a un obstacle
            {
                speedLeft = 0;
                speedRight = 0;
                dir = 'S';
                obs = 1;
            }
            else
            #endif //VL53
            if (table[0] > 1000 || table[1] > 1000)
            {
                if (obs > 10)
                {
                    speedLeft = VITESSE_OBS;
                    speedRight = -VITESSE_OBS/2;
                    dir = 'G';
                }
                else
                {
                    speedLeft = 0;
                    speedRight = 0;

                    if (table[0] > table[1] && table[0] > 1000)
                    {
                        dir = 'G';
                        speedLeft = VITESSE_OBS/2;
                        speedRight = -VITESSE_OBS/2;
                        if (obs%2 == 0)
                            obs++;
                    }
                    else if (table[0] < table[1] && table[1] > 1000)
                    {
                        dir = 'D';
                        speedLeft = -VITESSE_OBS/2;
                        speedRight = VITESSE_OBS/2;
                        if (obs%2 == 1)
                            obs++;
                    }
                }
            }
            else
            {
                speedLeft = VITESSE_OBS;
                speedRight = VITESSE_OBS;
                dir = 'F';
                obs = 0;
            }
        }
        else if (mode == MODE_CAM)
        {
            dir = 'N';
            obs = 0;

            if(x < 0 || y < 0){
                speedLeft = 0;
                speedRight = 0;
            }
            else {
                speedLeft = VITESSE_CAM - ((CAMERA_X_MAX/2 - x))/3; // (int) (((float)((x-CAMERA_X_MAX/2)/CAMERA_X_MAX))*500);
                speedRight = VITESSE_CAM + ((CAMERA_X_MAX/2 - x))/3; // (int) (((float)(x/CAMERA_X_MAX))*500);
            }



            /*if (x > CAMERA_X_MIN+CAMERA_X_TIER && x < CAMERA_X_MAX-CAMERA_X_TIER && y > CAMERA_Y_MIN && y <CAMERA_Y_MIN+CAMERA_Y_TIER) //AVANT
            {
                speedLeft = VITESSE_CAM;
                speedRight = VITESSE_CAM;
            }
            else if (x > CAMERA_X_MAX-CAMERA_X_TIER && x < CAMERA_X_MAX && y > CAMERA_Y_MIN && y < CAMERA_Y_MIN+CAMERA_Y_TIER) //AVANT_DROITE:
            {
                speedLeft = VITESSE_CAM;
                speedRight = VITESSE_CAM/2;
            }
            else if (x > CAMERA_X_MIN && x < CAMERA_X_MIN+CAMERA_X_TIER && y > CAMERA_Y_MIN && y < CAMERA_Y_MIN+CAMERA_Y_TIER) //AVANT_GAUCHE:
            {
                speedLeft = VITESSE_CAM/2;
                speedRight = VITESSE_CAM;
            }
            else if (x > CAMERA_X_MIN+CAMERA_X_TIER && x < CAMERA_X_MAX-CAMERA_X_TIER && y > CAMERA_Y_MIN+CAMERA_Y_TIER && y <CAMERA_Y_MAX-CAMERA_Y_TIER) //STOP
            {
                speedLeft = 0;
                speedRight = 0;
            }
            else if (x > CAMERA_X_MAX-CAMERA_X_TIER && x < CAMERA_X_MAX && y > CAMERA_Y_MIN+CAMERA_Y_TIER && y <CAMERA_Y_MAX-CAMERA_Y_TIER) //DROITE
            {
                speedLeft = VITESSE_CAM;
                speedRight = -VITESSE_CAM;
            }
            else if (x > CAMERA_X_MIN && x < CAMERA_X_MIN+CAMERA_X_TIER && y > CAMERA_Y_MIN+CAMERA_Y_TIER && y <CAMERA_Y_MAX-CAMERA_Y_TIER) //GAUCHE
            {
                speedLeft = -VITESSE_CAM;
                speedRight = VITESSE_CAM;
            }
            else if (x > CAMERA_X_MIN+CAMERA_X_TIER && x < CAMERA_X_MAX-CAMERA_X_TIER && y > CAMERA_Y_MAX-CAMERA_Y_TIER && y < CAMERA_Y_MAX) //RECULE:
            {
                speedLeft = -VITESSE_CAM;
                speedRight = -VITESSE_CAM;
            }
            else if (x > CAMERA_X_MAX-CAMERA_X_TIER && x < CAMERA_X_MAX && y > CAMERA_Y_MAX-CAMERA_Y_TIER && y < CAMERA_Y_MAX) //RECULE_DROITE:
            {
                speedLeft = -VITESSE_CAM/2;
                speedRight = -VITESSE_CAM;
            }
            else if (x > CAMERA_X_MIN && x < CAMERA_X_MIN+CAMERA_X_TIER && y > CAMERA_Y_MAX-CAMERA_Y_TIER && y < CAMERA_Y_MAX) //RECULE_GAUCHE:
            {
                speedLeft = -VITESSE_CAM;
                speedRight = -VITESSE_CAM/2;
            }
            else
            {
                speedLeft = 0;
                speedRight = 0;
            }*/
        }

        #if DEBUG_MOTOR
        printf("Motor L : %d || R : %d\r\n", speedLeft, speedRight);
        #endif

        xQueueSend( q_mot_L, ( void * ) &speedLeft,  portMAX_DELAY );
        xSemaphoreTake( xSem_Supervision, portMAX_DELAY );

        xQueueSend( q_mot_R, ( void * ) &speedRight,  portMAX_DELAY );
        xSemaphoreTake( xSem_Supervision, portMAX_DELAY );

    #if MICROROS
        MsgToPub.dir = dir;
        MsgToPub.mode = speedLeft; //mode;
        MsgToPub.speed = speedRight; //speed;
        if (!uxQueueMessagesWaiting(qhMR_pub))
            xQueueSend(qhMR_pub, ( void * ) &MsgToPub, portMAX_DELAY);
    #endif //MICROROS

    #if LCD
        if (!uxQueueMessagesWaiting(qhLCD))
        {
            pxMessage.data=mode;
            pxMessage.command=dir;
            xQueueSend( qhLCD, ( void * ) &pxMessage, portMAX_DELAY);
        }
    #endif //LCD

        vTaskDelay(SAMPLING_PERIOD_ms);
    }
#endif //SYNCHRO_EX
}

int main(void)
{
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
  MX_DMA_Init();
  MX_USART2_UART_Init();
  MX_I2C1_Init();
  MX_USART1_UART_Init();

  RetargetInit(&huart2); //make printf and scanf work with uart2
  printf("%cc%c[2J%c[0;0HTitouan//Jeremy//Louanne\r\n", 0x1b, 0x1b, 0x1b);

  motorCommand_Init();
  quadEncoder_Init();
  captDistIR_Init();

  HAL_Delay(500);

#if VL53
  initVL53L0X();
  for (int i=0 ; i<20 ; i++)
  {
      printf("%d\r\n", readRangeSingleMillimeters()/10);
  }
  HAL_Delay(500);
#endif //VL53

  // Test Ecran LCD
#if LCD
  groveLCD_begin(16,2,0); // !! cette fonction prend du temps
  HAL_Delay(100);
  groveLCD_setCursor(0,0);
  groveLCD_setColor(1);
  groveLCD_term_printf("Titouan//Jeremy//Louanne");
  HAL_Delay(1000);
#endif //LCD

  osKernelInitialize();
  //defaultTaskHandle = osThreadNew(microros_task, NULL, &defaultTask_attributes);
```

```

#if SYNCHRO_EX == EXSTARTUP
    #if MICROROS
    xTaskCreate( microros_task, ( const portCHAR * ) "microros_task", 3000 /* stack size */, NULL, 24, NULL);
    #endif //MICROROS
    xTaskCreate( task_Supervision, ( const portCHAR * ) "task Supervision", 128 /* stack size */, NULL, 27, NULL);
    xTaskCreate( task_Motor_Left, ( const portCHAR * ) "task Mot L", 128 /* stack size */, NULL, 25, NULL);
    xTaskCreate( task_Motor_Right, ( const portCHAR * ) "task Mot R", 128 /* stack size */, NULL, 26, NULL);
    #if LCD
    xTaskCreate( task_Grove_LCD, ( const portCHAR * ) "task Mot R", 128 /* stack size */, NULL, 23, NULL);
    #endif
#elif SYNCHRO_EX == EXTEST_UART2
    xTaskCreate(test_uart2, ( const portCHAR * ) "task print uart 2", 128 /* stack size */, NULL, tskIDLE_PRIORITY, NULL);
#elif SYNCHRO_EX == EXCORRECTOR
    xTaskCreate(test_motor, ( const portCHAR * ) "task test motor", 128 /* stack size */, NULL, tskIDLE_PRIORITY, NULL);
#elif SYNCHRO_EX == EXTESTCORRECTOR
    xTaskCreate(task_Supervision, ( const portCHAR * ) "task Supervision", 128 /* stack size */, NULL, 27, NULL);
    xTaskCreate(task_Motor_Left, ( const portCHAR * ) "task Motor Left", 128 /* stack size */, NULL, 25, NULL);
    xTaskCreate(task_Motor_Right, ( const portCHAR * ) "task Motor Right", 128 /* stack size */, NULL, 26, NULL);
#elif SYNCHRO_EX == EXTEST_VL53
    xTaskCreate(test_vl53, ( const portCHAR * ) "test_vl53", 128 /* stack size */, NULL, tskIDLE_PRIORITY, NULL);
#elif SYNCHRO_EX == EXTEST_MICROROS
    xTaskCreate(microros_task, ( const portCHAR * ) "microros_task", 3000 /* stack size */, NULL, tskIDLE_PRIORITY, NULL);
#elif SYNCHRO_EX == EXFINAL
    #if MICROROS
    xTaskCreate(microros_task, ( const portCHAR * ) "microros_task", 3000 /* stack size */, NULL, 24, NULL);
    #endif //MICROROS
    xTaskCreate(task_Supervision, ( const portCHAR * ) "task Supervision", 128 /* stack size */, NULL, 27, NULL);
    xTaskCreate(task_Motor_Left, ( const portCHAR * ) "task Motor Left", 128 /* stack size */, NULL, 25, NULL);
    xTaskCreate(task_Motor_Right, ( const portCHAR * ) "task Motor Right", 128 /* stack size */, NULL, 26, NULL);

    #if VL53
    xTaskCreate(task_VL53, ( const portCHAR * ) "task VL53", 128 /* stack size */, NULL, 23, NULL);
    #endif //VL53

    #if LCD
    xTaskCreate(task_Grove_LCD, ( const portCHAR * ) "task LCD", 128 /* stack size */, NULL, 23, NULL);
    #endif //LCD
#endif //SYNCHRO_EX

    vSemaphoreCreateBinary(xSem_Supervision);

    q_mot_L = xQueueCreate(1, sizeof(int16_t));
    q_mot_R = xQueueCreate(1, sizeof(int16_t));
    qhVl53 = xQueueCreate(1, sizeof(int));

    qhMR_sub = xQueueCreate(1, sizeof(MicroRosSubMsg));
    qhMR_pub = xQueueCreate(1, sizeof(MicroRosPubMsg));
    qhLCD = xQueueCreate(1, sizeof(AMessage));

  osKernelStart();
  while(1)
  {

  }
}

void test_uart2(void *pvParameters)
{
    char buf[100] = "";
    for(;;)
    {
        printf("Veuillez saisir votre nom :\r\n");
        scanf("%s", buf);
        printf("bonjour et bienvenue %s\r\n", buf);
        vTaskDelay(SAMPLING_PERIOD_ms);
    }
}

void test_vl53(void *pvParameters)
```

```
{
    uint16_t val;

    for(;;)
    {
        val = readRangeSingleMillimeters()/10;
        printf("Distance capteur : %d\r\n", val);
    }
}

void test_motor(void *pvParameters)
{
    int16_t  consigne = TEST_CORRECTOR_DUTY;
    if (consigne < 0 || consigne > 200)
        consigne = 150;
    int speed = 0;
    int i = 0;

    for (;;)
    {
        #if TEST_LEFT_MOTOR
        motorLeft_SetDuty(consigne);
        speed = quadEncoder_GetSpeedL();
        #else
        motorRight_SetDuty(consigne);
        speed = quadEncoder_GetSpeedR();
        #endif

        if(i<NB)
        {
            tab_speed[i]=speed;
            i++;
        }
        else
            printf("sampling end");
        vTaskDelay(SAMPLING_PERIOD_ms);
    }
}

//=========================================================================
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
  if (htim->Instance == TIM4)
  {
    HAL_IncTick();
  }
}
//=========================================================================
void Error_Handler(void)
{
  __disable_irq();
  while (1)
  {}
}
//=========================================================================
#ifdef  USE_FULL_ASSERT
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 */
```

5.30 microROS.c

```
#include "main.h"

#define STRING 0

void createPublisher(rcl_publisher_t* publisher,
    const rcl_node_t* node,
    const rosidl_message_type_support_t* type_support,
    const char* topic_name,
    std_msgs__msg__Int32* msg)
{
    rcl_ret_t ret = rclc_publisher_init_default(publisher, node, type_support, topic_name);
    printf("Publisher %s is created with result %d\r\n", topic_name, (int)ret);

#if STRING == 1
    (*msg).data.data = (char * ) malloc(ARRAY_LEN * sizeof(char));
    (*msg).data.size = 0;
    (*msg).data.capacity = ARRAY_LEN;
#else
    (*msg).data = 0;
#endif

}

void createSubscriber(rcl_subscription_t* subscription,
    rcl_node_t* node,
    const rosidl_message_type_support_t* type_support,
    const char* topic_name,
    std_msgs__msg__Int32* msg)
{
    *subscription = rcl_get_zero_initialized_subscription();

    rcl_ret_t ret = rclc_subscription_init_default(subscription, node,
        type_support, topic_name);
    printf("Subscription %s is created with result %d\r\n", topic_name, (int)ret);

#if STRING == 1
    (*msg).data.data = (char * ) malloc(ARRAY_LEN * sizeof(char));
    (*msg).data.size = 0;
    (*msg).data.capacity = ARRAY_LEN;
#else
    (*msg).data = 0;
#endif
}
```

5.31 microros_allocators.c

```

#include <unistd.h>
#include "cmsis_os.h"

int absoluteUsedMemory = 0;
int usedMemory = 0;

void *pvPortMallocMicroROS( size_t xWantedSize );
void vPortFreeMicroROS( void *pv );
void *pvPortReallocMicroROS( void *pv, size_t xWantedSize );
size_t getBlockSize( void *pv );
void *pvPortCallocMicroROS( size_t num, size_t xWantedSize );

void * microros_allocate(size_t size, void * state){
  (void) state;
  // printf("-- Alloc %d (prev: %d B)\n",size, xPortGetFreeHeapSize());
  absoluteUsedMemory += size;
  usedMemory += size;
  return pvPortMallocMicroROS(size);
}

void microros_deallocate(void * pointer, void * state){
  (void) state;
  // printf("-- Free %d (prev: %d B)\n",getBlockSize(pointer), xPortGetFreeHeapSize());
  if (NULL != pointer){
    usedMemory -= getBlockSize(pointer);
    vPortFreeMicroROS(pointer);
  }
}

void * microros_reallocate(void * pointer, size_t size, void * state){
  (void) state;
  // printf("-- Realloc %d -> %d (prev: %d B)\n",getBlockSize(pointer),size, xPortGetFreeHeapSize());
  absoluteUsedMemory += size;
  usedMemory += size;
  if (NULL == pointer){
    return pvPortMallocMicroROS(size);
  } else {
    usedMemory -= getBlockSize(pointer);
    return pvPortReallocMicroROS(pointer,size);
  }
}

void * microros_zero_allocate(size_t number_of_elements, size_t size_of_element, void * state){
  (void) state;
  // printf("-- Calloc %d x %d = %d -> (prev: %d B)\n",number_of_elements,size_of_element, number_of_elements*size_of_element, xPortGetFreeHeapSize());
  absoluteUsedMemory += number_of_elements*size_of_element;
  usedMemory += number_of_elements*size_of_element;
  return pvPortCallocMicroROS(number_of_elements,size_of_element);
}
```

5.32 microros_time.c

```
#include <unistd.h>
#include <time.h>
#include "cmsis_os.h"

#define MICROSECONDS_PER_SECOND    ( 1000000LL )
#define NANOSECONDS_PER_SECOND     ( 1000000000LL )
#define NANOSECONDS_PER_TICK       ( NANOSECONDS_PER_SECOND / configTICK_RATE_HZ )
void UTILS_NanosecondsToTimespec( int64_t llSource,
                                  struct timespec * const pxDestination )
{
    long lCarrySec = 0;

    /* Convert to timespec. */
    pxDestination->tv_sec = ( time_t ) ( llSource / NANOSECONDS_PER_SECOND );
    pxDestination->tv_nsec = ( long ) ( llSource % NANOSECONDS_PER_SECOND );

    /* Subtract from tv_sec if tv_nsec < 0. */
    if( pxDestination->tv_nsec < 0L )
    {
        /* Compute the number of seconds to carry. */
        lCarrySec = ( pxDestination->tv_nsec / ( long ) NANOSECONDS_PER_SECOND ) + 1L;

        pxDestination->tv_sec -= ( time_t ) ( lCarrySec );
        pxDestination->tv_nsec += lCarrySec * ( long ) NANOSECONDS_PER_SECOND;
    }
}

int clock_gettime( int clock_id,
                   struct timespec * tp )
{
    TimeOut_t xCurrentTime = { 0 };

    /* Intermediate variable used to convert TimeOut_t to struct timespec.
     * Also used to detect overflow issues. It must be unsigned because the
     * behavior of signed integer overflow is undefined. */
    uint64_t ullTickCount = 0ULL;

    /* Silence warnings about unused parameters. */
    ( void ) clock_id;

    /* Get the current tick count and overflow count. vTaskSetTimeOutState()
     * is used to get these values because they are both static in tasks.c. */
    vTaskSetTimeOutState( &xCurrentTime );

    /* Adjust the tick count for the number of times a TickType_t has overflowed.
     * portMAX_DELAY should be the maximum value of a TickType_t. */
    ullTickCount = ( uint64_t ) ( xCurrentTime.xOverflowCount ) « ( sizeof( TickType_t ) * 8 );

    /* Add the current tick count. */
    ullTickCount += xCurrentTime.xTimeOnEntering;

    /* Convert ullTickCount to timespec. */
    UTILS_NanosecondsToTimespec( ( int64_t ) ullTickCount * NANOSECONDS_PER_TICK, tp );

    return 0;
}
```

5.33 motorCommand.c

```
/*
 * MotorCommand.c
 */

#include "motorCommand.h"

static TIM_HandleTypeDef    TimHandle;
static TIM_OC_InitTypeDef   sConfigOC;

//=================================================================
//          PWM INIT
// TIMER 3 (PWM)  : CH1 et CH2
// ENABLE : Sortie Logique (GPIO) PA7
//=================================================================

void motorCommand_Init(void)
{
    unsigned int uwPrescalerValue = 0;

    /* Compute the prescaler value to have TIM4 counter clock equal to 10MHz */
      uwPrescalerValue = (unsigned int) ((SystemCoreClock / 10000000) - 1);
      TimHandle.Instance = TIM3;
      TimHandle.Init.Period = 200 - 1; // 100MHz/200=50kHz
      TimHandle.Init.Prescaler = uwPrescalerValue;
      TimHandle.Init.ClockDivision = 0;
      TimHandle.Init.CounterMode = TIM_COUNTERMODE_UP;

      HAL_TIM_Base_Init(&TimHandle);

      sConfigOC.OCMode = TIM_OCMODE_PWM1;
      sConfigOC.Pulse = 0x5;// Specifies the pulse value to be loaded into the Capture Compare Register. This parameter can be a number between Min_Data = 0x0000 and Max_Data = 0xFFFF */

      sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
      sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;

     HAL_TIM_PWM_ConfigChannel(&TimHandle, &sConfigOC, TIM_CHANNEL_1);
     HAL_TIM_PWM_ConfigChannel(&TimHandle, &sConfigOC, TIM_CHANNEL_2);

     // CHANGEMENT DU RAPPORT CYCLIQUE
     __HAL_TIM_SetCompare(&TimHandle, TIM_CHANNEL_1, 100);  // 100 : moteurs au repos
     __HAL_TIM_SetCompare(&TimHandle, TIM_CHANNEL_2, 100);

      HAL_TIM_PWM_Start(&TimHandle, TIM_CHANNEL_1); // MOTOR RIGHT
      HAL_TIM_PWM_Start(&TimHandle, TIM_CHANNEL_2); // MOTOR LEFT

      // ENABLE MOTEUR (SI INVERSEUR)
      //HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, 0);
      HAL_GPIO_WritePin(GPIOB, GPIO_PIN_3, 0);
}

//=================================================================
//          SET DUTY CYCLE LEFT
//=================================================================
void motorLeft_SetDuty(int duty)
{
    __HAL_TIM_SetCompare(&TimHandle, TIM_CHANNEL_1, duty);
}
//=================================================================
//          SET DUTY CYCLE RIGHT
//=================================================================
void motorRight_SetDuty(int duty)
{
    __HAL_TIM_SetCompare(&TimHandle, TIM_CHANNEL_2, duty);
}
//=================================================================


```

5.34 quadEncoder.c

```
/*
 * QuadEncoder.c
 */
#include "quadEncoder.h"

#define SAMPLING_PERIOD_ms      5
#define TE_ms   SAMPLING_PERIOD_ms
#define USE_QUAD_ENCODER_1250_CPR 1

#if USE_QUAD_ENCODER_1250_CPR
#define COUNT_PER_ROUND 1250
#define MAX_CNT_PER_REV (COUNT_PER_ROUND * 4 - 1)
#define MAX_COUNT (int)(((unsigned long)MAX_CNT_PER_REV*6555)/1000)
#define HALF_MAX_COUNT (MAX_COUNT»1)
#define COEFF   6555
#endif

#if USE_QUAD_ENCODER_1000_CPR
#define COUNT_PER_ROUND 1000
#define MAX_CNT_PER_REV (COUNT_PER_ROUND * 4 - 1)
#define MAX_COUNT (int)(((unsigned long)MAX_CNT_PER_REV*8192)/1000)
#define HALF_MAX_COUNT (MAX_COUNT»1)
#define COEFF 8192
#endif

#if USE_QUAD_ENCODER_500_CPR
#define COUNT_PER_ROUND 500
#define MAX_CNT_PER_REV (COUNT_PER_ROUND * 4 - 1)
#define MAX_COUNT (int)(((unsigned long)MAX_CNT_PER_REV*16392)/1000)
#define HALF_MAX_COUNT (MAX_COUNT»1)
#define COEFF 16392
#endif

#if USE_QUAD_ENCODER_250_CPR
```

```c
#define COUNT_PER_ROUND 250
#define MAX_CNT_PER_REV (COUNT_PER_ROUND * 4 - 1)
#define MAX_COUNT (int)(((unsigned long)MAX_CNT_PER_REV*32768)/1000)
#define HALF_MAX_COUNT (MAX_COUNT»1)
#define COEFF 32768
#endif



TIM_HandleTypeDef    TimEncoderHandleLeft;
TIM_HandleTypeDef    TimEncoderHandleRight;

/*******************************************************************************
 * TIMER 1, CHANNEL 1 et 2  --> RIGHT
 * TIMER 2, CHANNEL 1 et 2  --> LEFT
****************************************************************************** */
int indexL=0;
static int indexR=0;

//================================================================
//      TIMER INIT
//================================================================

void quadEncoder_Init(void)
{
    TIM_Encoder_InitTypeDef sConfig;
    //--------------------------------------------------
    // TIMER 1
    //--------------------------------------------------
    TimEncoderHandleLeft.Instance = TIM1;
    TimEncoderHandleLeft.Init.Prescaler = 0;
    TimEncoderHandleLeft.Init.CounterMode = TIM_COUNTERMODE_UP;
    TimEncoderHandleLeft.Init.Period = COUNT_PER_ROUND*4;
    TimEncoderHandleLeft.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;

    sConfig.EncoderMode = TIM_ENCODERMODE_TI12;
    sConfig.IC1Polarity = TIM_INPUTCHANNELPOLARITY_RISING;
    sConfig.IC1Selection = TIM_ICSELECTION_DIRECTTI;
    sConfig.IC1Prescaler = TIM_ICPSC_DIV4;
    sConfig.IC1Filter = 0x0F;
    sConfig.IC2Polarity = TIM_INPUTCHANNELPOLARITY_RISING;
    sConfig.IC2Selection = TIM_ICSELECTION_DIRECTTI;//TIM_ICSELECTION_DIRECTTI; //TIM_TI1SELECTION_XORCOMBINATION
    sConfig.IC2Prescaler = TIM_ICPSC_DIV4;
    sConfig.IC2Filter = 0x0F;

    HAL_TIM_Encoder_Init(&TimEncoderHandleLeft, &sConfig);

    __HAL_TIM_SetCounter(&TimEncoderHandleLeft, 0);

    HAL_TIM_Encoder_Start(&TimEncoderHandleLeft,TIM_CHANNEL_1);
    HAL_TIM_Encoder_Start(&TimEncoderHandleLeft,TIM_CHANNEL_2);

    //--------------------------------------------------
    // TIMER 2
    //--------------------------------------------------
    TimEncoderHandleRight.Instance = TIM2;
    TimEncoderHandleRight.Init.Prescaler = 0;
    TimEncoderHandleRight.Init.CounterMode = TIM_COUNTERMODE_UP;
    TimEncoderHandleRight.Init.Period = COUNT_PER_ROUND*4;
    TimEncoderHandleRight.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;

    sConfig.EncoderMode = TIM_ENCODERMODE_TI12;
    sConfig.IC1Polarity = TIM_INPUTCHANNELPOLARITY_RISING;
    sConfig.IC1Selection = TIM_ICSELECTION_DIRECTTI;
    sConfig.IC1Prescaler = TIM_ICPSC_DIV4;
    sConfig.IC1Filter = 0x0F;
    sConfig.IC2Polarity = TIM_INPUTCHANNELPOLARITY_RISING;
    sConfig.IC2Selection = TIM_ICSELECTION_DIRECTTI;//TIM_ICSELECTION_DIRECTTI; //TIM_TI1SELECTION_XORCOMBINATION
    sConfig.IC2Prescaler = TIM_ICPSC_DIV4;
    sConfig.IC2Filter = 0x0F;

    HAL_TIM_Encoder_Init(&TimEncoderHandleRight, &sConfig);

    __HAL_TIM_SetCounter(&TimEncoderHandleRight, 0);

    HAL_TIM_Encoder_Start(&TimEncoderHandleRight,TIM_CHANNEL_1);
    HAL_TIM_Encoder_Start(&TimEncoderHandleRight,TIM_CHANNEL_2);
}

//================================================================
//      POSITION LEFT CALC
//================================================================

void quadEncoder_PosCalcL(int* AngPos)
{
```

```

int POSCNTcopy = 0;
POSCNTcopy = (int)TIM1->CNT;
AngPos[1] = AngPos[0];
AngPos[0] = (unsigned int)(((unsigned long)POSCNTcopy * COEFF)/1000); // 0 <= POSCNT <= 4999 to 0 <= AngPos <= 32767
}

//================================================================
//      POSITION RIGHT CALC
//================================================================

void quadEncoder_PosCalcR(int* AngPos)
{

int POSCNTcopy = 0;
POSCNTcopy = (int)TIM2->CNT;
AngPos[1] = AngPos[0];
AngPos[0] = (unsigned int)(((unsigned long)POSCNTcopy * COEFF)/1000); // 0 <= POSCNT <= 4999 to 0 <= AngPos <= 32767
}

//================================================================
//      POSITION LEFT 16 BITS
//================================================================

int16_t quadEncoder_GetPos16L(void)
{
    uint16_t PosL = 0;
    PosL=TIM1->CNT;
    return (int16_t)PosL;

    }

//================================================================
//      POSITION RIGHT 16 BITS
//================================================================

int16_t quadEncoder_GetPos16R(void)
{
    uint16_t PosR = 0;
    PosR=TIM2->CNT;
    return (int16_t)PosR;
}
//================================================================
//      POSITION LEFT 32 BITS   (pos 16 bits + nombre de tours)
//================================================================

int32_t quadEncoder_GetPos32L(void)
{
    int32_t  PosL = 0;
    PosL=indexL*4*COUNT_PER_ROUND + (int32_t) quadEncoder_GetPos16L();
    return PosL;
}

//================================================================
//      POSITION RIGHT 32 BITS  (pos 16 bits + nombre de tours)
//================================================================

int32_t quadEncoder_GetPos32R(void)
{
    int32_t PosR = 0;
    PosR=indexR*4*COUNT_PER_ROUND + (int32_t) quadEncoder_GetPos16R();
    return PosR;
}

//================================================================
//      SPEED LEFT
//--> must be called every Te
//================================================================

int16_t quadEncoder_GetSpeedL(void)
{
    static int AngPos[2] = {0,0};
    static int16_t SpeedL=0;

    quadEncoder_PosCalcL(AngPos);
    SpeedL = AngPos[0] - AngPos[1];
    if (SpeedL >= 0)
    {
        if (SpeedL >= HALF_MAX_COUNT)
            {
            SpeedL = SpeedL - MAX_COUNT;
            }
    }
    else
    {
        if (SpeedL < -HALF_MAX_COUNT)
            {
            SpeedL = SpeedL + MAX_COUNT;
            }
    }

    //***********************************
    // CONVERT RPM
    // 1 tour = 32767
    // Nbre de Tours pendant Te: DELTA_pos/32767
    // Nbre de Tours pendant 1s (Te en ms) : (DELTA_pos/32767)*(1000/Te)
    // Nbre de Tours par minute : : (DELTA_pos/32767)*((60*1000)/Te)

    SpeedL=(SpeedL*60*1000)/(32767*TE_ms);
    return SpeedL;
}

//================================================================
//      SPEED RIGHT
//-->  must be called every Te
//================================================================

int16_t quadEncoder_GetSpeedR(void)
{
    static int AngPos[2] = {0,0};
    static int16_t SpeedR=0;


    quadEncoder_PosCalcR(AngPos);
    SpeedR = AngPos[0] - AngPos[1];
    if (SpeedR >= 0)
    {
        if (SpeedR >= HALF_MAX_COUNT)
            {
            SpeedR = SpeedR - MAX_COUNT;
            }
    }
    else
    {
        if (SpeedR < -HALF_MAX_COUNT)
            {
            SpeedR = SpeedR + MAX_COUNT;
            }
    }
    //***********************************
    // CONVERT RPM
    // 1 tour = 32767
    // Nbre de Tours pendant Te: DELTA_pos/32767
    // Nbre de Tours pendant 1s (Te en ms) : (DELTA_pos/32767)*(1000/Te)
    // Nbre de Tours par minute : : (DELTA_pos/32767)*((60*1000)/Te)

    SpeedR=(SpeedR*60*1000)/(32767*TE_ms);
    return SpeedR;
}

//================================================================
//      MAJ index Left
//================================================================

void quadEncoder_CallbackIndexL()
{
                if (__HAL_TIM_DIRECTION_STATUS(&TimEncoderHandleLeft)==1)
                {
                    indexL--;
                }
                else
                {
                    indexL++;
                }


                __HAL_TIM_SetCounter(&TimEncoderHandleLeft, 0);     // RAZ Counter
                HAL_TIM_Encoder_Start(&TimEncoderHandleLeft,TIM_CHANNEL_1);
                HAL_TIM_Encoder_Start(&TimEncoderHandleLeft,TIM_CHANNEL_2);

}
//================================================================
//      MAJ index Right
//================================================================

void quadEncoder_CallbackIndexR()
{
                if (__HAL_TIM_DIRECTION_STATUS(&TimEncoderHandleRight)==1)
                {
                    indexR--;

                }
                else
                {
                    indexR++;
                }

                __HAL_TIM_SetCounter(&TimEncoderHandleRight, 0);            // RAZ Counter
                HAL_TIM_Encoder_Start(&TimEncoderHandleRight,TIM_CHANNEL_1);
                HAL_TIM_Encoder_Start(&TimEncoderHandleRight,TIM_CHANNEL_2);

}
//==================================================================



```

5.35 retarget.c

```
// All credit to Carmine Noviello for this code
// https://github.com/cnoviello/mastering-stm32/blob/master/nucleo-f030R8/system/src/retarget/retarget.c

#include <_ansi.h>
#include <_syslist.h>
#include <errno.h>
#include <sys/time.h>
#include <sys/times.h>
#include <limits.h>
#include <signal.h>
#include <../Inc/retarget.h>
#include <stdint.h>
#include <stdio.h>

#if !defined(OS_USE_SEMIHOSTING)

#define STDIN_FILENO  0
#define STDOUT_FILENO 1
#define STDERR_FILENO 2

UART_HandleTypeDef *gHuart;

void RetargetInit(UART_HandleTypeDef *huart) {
  gHuart = huart;

  /* Disable I/O buffering for STDOUT stream, so that
   * chars are sent out as soon as they are printed. */
  setvbuf(stdout, NULL, _IONBF, 0);
}

int _isatty(int fd) {
  if (fd >= STDIN_FILENO && fd <= STDERR_FILENO)
    return 1;

  errno = EBADF;
  return 0;
}

int _write(int fd, char* ptr, int len) {
  HAL_StatusTypeDef hstatus;

  if (fd == STDOUT_FILENO || fd == STDERR_FILENO) {
    hstatus = HAL_UART_Transmit(gHuart, (uint8_t *) ptr, len, HAL_MAX_DELAY);
    if (hstatus == HAL_OK)
      return len;
    else
      return EIO;
  }
  errno = EBADF;
  return -1;
}

int _close(int fd) {
  if (fd >= STDIN_FILENO && fd <= STDERR_FILENO)
    return 0;

  errno = EBADF;
  return -1;
}

int _lseek(int fd, int ptr, int dir) {
  (void) fd;
  (void) ptr;
  (void) dir;

  errno = EBADF;
  return -1;
}

int _read(int fd, char* ptr, int len) {
  HAL_StatusTypeDef hstatus;

  if (fd == STDIN_FILENO) {
    hstatus = HAL_UART_Receive(gHuart, (uint8_t *) ptr, 1, HAL_MAX_DELAY);
    if (hstatus == HAL_OK)
      return 1;
    else
      return EIO;
  }
  errno = EBADF;
  return -1;
}

int _fstat(int fd, struct stat* st) {
  if (fd >= STDIN_FILENO && fd <= STDERR_FILENO) {
    st->st_mode = S_IFCHR;
    return 0;
  }

  errno = EBADF;
  return 0;
}

int _getpid(void)
{
    return 1;
}

int _kill(int pid, int sig)
{
    errno = EINVAL;
    return -1;
}

#endif //#if !defined(OS_USE_SEMIHOSTING)
```

5.36 stm32f4xx_hal_msp.c

```
#include "main.h"

#define USART2_IRQ_PRIO 9
#define USART6_IRQ_PRIO 10
//#define EXTI1_IRQ_PRIO    7
#define EXTI0_IRQ_PRIO  6
#define EXTI15_10_IRQ_PRIO  7
#define I2C1_ER_IRQ_PRIO    2
#define I2C1_EV_IRQ_PRIO    11
#define TIM5_IRQ_PRIO       12

extern DMA_HandleTypeDef hdma_usart1_rx;
extern DMA_HandleTypeDef hdma_usart1_tx;
extern DMA_HandleTypeDef hdma_usart2_rx;
extern DMA_HandleTypeDef hdma_usart2_tx;

void HAL_PWM_Timer3_MspInit(void);
void HAL_Encoder_Timer1_MspInit(void);
void HAL_Encoder_Timer2_MspInit(void);
void HAL_adcir_MspInit(void);

void HAL_MspInit(void)
{
  __HAL_RCC_SYSCFG_CLK_ENABLE();
  __HAL_RCC_PWR_CLK_ENABLE();

  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOH_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();

  /* System interrupt init*/
  /* PendSV_IRQn interrupt configuration */
  HAL_NVIC_SetPriority(PendSV_IRQn, 15, 0);

  HAL_PWM_Timer3_MspInit();
  HAL_Encoder_Timer1_MspInit();
  HAL_Encoder_Timer2_MspInit();
  HAL_adcir_MspInit();
}
```

```

/******************************************************************
            ENCODER - TIMER1
PWM1/1  --> PA8     -- Encodeur Voie A
PWM1/2  --> PA9     -- Encodeur Voie B
EXTI1   --> PB10        -- Index encodeur
******************************************************************/
void HAL_Encoder_Timer1_MspInit(void)
{
      GPIO_InitTypeDef  GPIO_InitStruct;

      __TIM1_CLK_ENABLE();

      GPIO_InitStruct.Pin = GPIO_PIN_8 | GPIO_PIN_9;
      GPIO_InitStruct.Mode = GPIO_MODE_AF_PP; // hal_gpio.h
      GPIO_InitStruct.Pull = GPIO_PULLUP;
      GPIO_InitStruct.Speed = GPIO_SPEED_MEDIUM;
      GPIO_InitStruct.Alternate =  GPIO_AF1_TIM1 ; // hal_gpio_ex.h

      HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

      GPIO_InitStruct.Pin = GPIO_PIN_10;
      GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
      GPIO_InitStruct.Pull = GPIO_NOPULL;

      HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

      /* Enable and set EXTI Line0 Interrupt to the lowest priority */
      HAL_NVIC_SetPriority(EXTI15_10_IRQn, EXTI15_10_IRQ_PRIO, 0);
      HAL_NVIC_EnableIRQ(EXTI15_10_IRQn);
}
/******************************************************************
                ENCODER - TIMER2
PWM2/1  --> PA0     -- Encodeur Voie A
PWM2/2  --> PA1     -- Encodeur Voie B
EXTI    --> PC0     -- Index Moteur
******************************************************************/
void HAL_Encoder_Timer2_MspInit(void)
{
      GPIO_InitTypeDef  GPIO_InitStruct;

      __TIM2_CLK_ENABLE();

      GPIO_InitStruct.Pin = GPIO_PIN_0 | GPIO_PIN_1;
      GPIO_InitStruct.Mode = GPIO_MODE_AF_PP; // hal_gpio.h
      GPIO_InitStruct.Pull = GPIO_PULLUP;
      GPIO_InitStruct.Speed = GPIO_SPEED_MEDIUM;
      GPIO_InitStruct.Alternate =  GPIO_AF1_TIM2 ; // hal_gpio_ex.h

      HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

      GPIO_InitStruct.Pin = GPIO_PIN_0;
      GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
      GPIO_InitStruct.Pull = GPIO_NOPULL;

      HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

      /* Enable and set EXTI Line0 Interrupt to the lowest priority */
      HAL_NVIC_SetPriority(EXTI0_IRQn, EXTI0_IRQ_PRIO, 0);
      HAL_NVIC_EnableIRQ(EXTI0_IRQn);
}

/******************************************************************
//          PWM - TIMER4 COMMANDE MOTEURS
PA6 --> PWM3/1
PC7 --> PWM3/2
PB3 --> ENABLE MOTEUR (actif état Bas)
******************************************************************/
void HAL_PWM_Timer3_MspInit(void)
{
      GPIO_InitTypeDef  GPIO_InitStruct;

      __TIM3_CLK_ENABLE();

      GPIO_InitStruct.Pin = GPIO_PIN_6;
      GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
      GPIO_InitStruct.Pull = GPIO_PULLUP;
      GPIO_InitStruct.Speed = GPIO_SPEED_MEDIUM;
      GPIO_InitStruct.Alternate =  GPIO_AF2_TIM3 ; // hal_gpio_ex.h

      HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

      GPIO_InitStruct.Pin = GPIO_PIN_7;
      GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
      GPIO_InitStruct.Pull = GPIO_PULLUP;
      GPIO_InitStruct.Speed = GPIO_SPEED_MEDIUM;
      GPIO_InitStruct.Alternate =  GPIO_AF2_TIM3 ; // hal_gpio_ex.h
      HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

      // ENABLE MOTEUR : SORTIE LOGIQUE PB3
      GPIO_InitStruct.Pin = GPIO_PIN_7;
      GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
      GPIO_InitStruct.Pull = GPIO_NOPULL;

      HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
      HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, 1);

      GPIO_InitStruct.Pin = GPIO_PIN_3;
      GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
      GPIO_InitStruct.Pull = GPIO_PULLUP;
      GPIO_InitStruct.Speed = GPIO_SPEED_FAST;

      HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
      HAL_GPIO_WritePin(GPIOB, GPIO_PIN_3, 1);




}

/*******************************************************************
            ADC
ADC1_4  --> PA4
ADC1_8  --> PB0
http://stm32f4-discovery.com/2014/04/library-06-ad-converter-on-stm32f4xx/
*******************************************************************/
void HAL_adcir_MspInit(void)
{
      GPIO_InitTypeDef  GPIO_InitStruct;
      /* Peripheral clock enable */
      __ADC1_CLK_ENABLE();

      GPIO_InitStruct.Pin = GPIO_PIN_4 ;
      GPIO_InitStruct.Mode = GPIO_MODE_ANALOG;
      GPIO_InitStruct.Pull = GPIO_NOPULL;

      HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);


      GPIO_InitStruct.Pin = GPIO_PIN_0 ;
      GPIO_InitStruct.Mode = GPIO_MODE_ANALOG;
      GPIO_InitStruct.Pull = GPIO_NOPULL;

      HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

}


/*******************************************************************
* @brief I2C MSP Initialization
* This function configures the hardware resources used in this example
* @param hi2c: I2C handle pointer
* @retval None
*******************************************************************/
void HAL_I2C_MspInit(I2C_HandleTypeDef* hi2c)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
  if(hi2c->Instance==I2C1)
  {
    __HAL_RCC_GPIOB_CLK_ENABLE();
    GPIO_InitStruct.Pin = GPIO_PIN_8|GPIO_PIN_9;
    GPIO_InitStruct.Mode = GPIO_MODE_AF_OD;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
    GPIO_InitStruct.Alternate = GPIO_AF4_I2C1;
    HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

    /* Peripheral clock enable */
    __HAL_RCC_I2C1_CLK_ENABLE();

      HAL_NVIC_SetPriority(I2C1_ER_IRQn, I2C1_ER_IRQ_PRIO, 0);
      HAL_NVIC_EnableIRQ(I2C1_ER_IRQn);
      HAL_NVIC_SetPriority(I2C1_EV_IRQn, I2C1_EV_IRQ_PRIO, 0);
      HAL_NVIC_EnableIRQ(I2C1_EV_IRQn);



  }

}

/*******************************************************************
* @brief I2C MSP De-Initialization
* This function freeze the hardware resources used in this example
* @param hi2c: I2C handle pointer
* @retval None
*******************************************************************/
void HAL_I2C_MspDeInit(I2C_HandleTypeDef* hi2c)
{
  if(hi2c->Instance==I2C1)
  {
  /* USER CODE BEGIN I2C1_MspDeInit 0 */

  /* USER CODE END I2C1_MspDeInit 0 */
    /* Peripheral clock disable */
    __HAL_RCC_I2C1_CLK_DISABLE();

    HAL_GPIO_DeInit(GPIOB, GPIO_PIN_6);

    HAL_GPIO_DeInit(GPIOB, GPIO_PIN_7);

  /* USER CODE BEGIN I2C1_MspDeInit 1 */

  /* USER CODE END I2C1_MspDeInit 1 */
  }

}

/*******************************************************************
* @brief UART MSP Initialization
* This function configures the hardware resources used in this example
* @param huart: UART handle pointer
* @retval None
*******************************************************************/
void HAL_UART_MspInit(UART_HandleTypeDef* huart)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
  if(huart->Instance==USART1)
  {
  /* USER CODE BEGIN USART1_MspInit 0 */

  /* USER CODE END USART1_MspInit 0 */
    /* Peripheral clock enable */
    __HAL_RCC_USART1_CLK_ENABLE();

    __HAL_RCC_GPIOA_CLK_ENABLE();
    GPIO_InitStruct.Pin = GPIO_PIN_10;
    GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
    GPIO_InitStruct.Alternate = GPIO_AF7_USART1;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

    GPIO_InitStruct.Pin = GPIO_PIN_6;
    GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
    GPIO_InitStruct.Alternate = GPIO_AF7_USART1;
    HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

    /* USART1 DMA Init */
    /* USART1_RX Init */
    hdma_usart1_rx.Instance = DMA2_Stream2;
    hdma_usart1_rx.Init.Channel = DMA_CHANNEL_4;
    hdma_usart1_rx.Init.Direction = DMA_PERIPH_TO_MEMORY;
    hdma_usart1_rx.Init.PeriphInc = DMA_PINC_DISABLE;
    hdma_usart1_rx.Init.MemInc = DMA_MINC_ENABLE;
    hdma_usart1_rx.Init.PeriphDataAlignment = DMA_PDATAALIGN_BYTE;
    hdma_usart1_rx.Init.MemDataAlignment = DMA_MDATAALIGN_BYTE;
    hdma_usart1_rx.Init.Mode = DMA_CIRCULAR;
    hdma_usart1_rx.Init.Priority = DMA_PRIORITY_VERY_HIGH;
    hdma_usart1_rx.Init.FIFOMode = DMA_FIFOMODE_DISABLE;
    if (HAL_DMA_Init(&hdma_usart1_rx) != HAL_OK)
    {
      Error_Handler();
    }

    __HAL_LINKDMA(huart,hdmarx,hdma_usart1_rx);

    /* USART1_TX Init */
    hdma_usart1_tx.Instance = DMA2_Stream7;
    hdma_usart1_tx.Init.Channel = DMA_CHANNEL_4;
    hdma_usart1_tx.Init.Direction = DMA_MEMORY_TO_PERIPH;
    hdma_usart1_tx.Init.PeriphInc = DMA_PINC_DISABLE;
    hdma_usart1_tx.Init.MemInc = DMA_MINC_ENABLE;
    hdma_usart1_tx.Init.PeriphDataAlignment = DMA_PDATAALIGN_BYTE;
    hdma_usart1_tx.Init.MemDataAlignment = DMA_MDATAALIGN_BYTE;
    hdma_usart1_tx.Init.Mode = DMA_NORMAL;
    hdma_usart1_tx.Init.Priority = DMA_PRIORITY_VERY_HIGH;
    hdma_usart1_tx.Init.FIFOMode = DMA_FIFOMODE_DISABLE;
```

```c
    if (HAL_DMA_Init(&hdma_usart1_tx) != HAL_OK)
    {
      Error_Handler();
    }

    __HAL_LINKDMA(huart,hdmatx,hdma_usart1_tx);

    /* USART1 interrupt Init */
    HAL_NVIC_SetPriority(USART1_IRQn, 5, 0);
    HAL_NVIC_EnableIRQ(USART1_IRQn);
  /* USER CODE BEGIN USART1_MspInit 1 */

  /* USER CODE END USART1_MspInit 1 */
  }
  else if(huart->Instance==USART2)
  {
  /* USER CODE BEGIN USART2_MspInit 0 */

  /* USER CODE END USART2_MspInit 0 */
    /* Peripheral clock enable */
    __HAL_RCC_USART2_CLK_ENABLE();

    __HAL_RCC_GPIOA_CLK_ENABLE();
    GPIO_InitStruct.Pin = USART_TX_Pin|USART_RX_Pin;
    GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
    GPIO_InitStruct.Alternate = GPIO_AF7_USART2;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

    /* USART2 DMA Init */
    /* USART2_RX Init */
    hdma_usart2_rx.Instance = DMA1_Stream5;
    hdma_usart2_rx.Init.Channel = DMA_CHANNEL_4;
    hdma_usart2_rx.Init.Direction = DMA_PERIPH_TO_MEMORY;
    hdma_usart2_rx.Init.PeriphInc = DMA_PINC_DISABLE;
    hdma_usart2_rx.Init.MemInc = DMA_MINC_ENABLE;
    hdma_usart2_rx.Init.PeriphDataAlignment = DMA_PDATAALIGN_BYTE;
    hdma_usart2_rx.Init.MemDataAlignment = DMA_MDATAALIGN_BYTE;
    hdma_usart2_rx.Init.Mode = DMA_CIRCULAR;
    hdma_usart2_rx.Init.Priority = DMA_PRIORITY_VERY_HIGH;
    hdma_usart2_rx.Init.FIFOMode = DMA_FIFOMODE_DISABLE;
    if (HAL_DMA_Init(&hdma_usart2_rx) != HAL_OK)
    {
      Error_Handler();
    }

    __HAL_LINKDMA(huart,hdmarx,hdma_usart2_rx);

    /* USART2_TX Init */
    hdma_usart2_tx.Instance = DMA1_Stream6;
    hdma_usart2_tx.Init.Channel = DMA_CHANNEL_4;
    hdma_usart2_tx.Init.Direction = DMA_MEMORY_TO_PERIPH;
    hdma_usart2_tx.Init.PeriphInc = DMA_PINC_DISABLE;
    hdma_usart2_tx.Init.MemInc = DMA_MINC_ENABLE;
    hdma_usart2_tx.Init.PeriphDataAlignment = DMA_PDATAALIGN_BYTE;
    hdma_usart2_tx.Init.MemDataAlignment = DMA_MDATAALIGN_BYTE;
    hdma_usart2_tx.Init.Mode = DMA_NORMAL;
    hdma_usart2_tx.Init.Priority = DMA_PRIORITY_VERY_HIGH;
    hdma_usart2_tx.Init.FIFOMode = DMA_FIFOMODE_DISABLE;
    if (HAL_DMA_Init(&hdma_usart2_tx) != HAL_OK)
    {
      Error_Handler();
    }

    __HAL_LINKDMA(huart,hdmatx,hdma_usart2_tx);

    /* USART2 interrupt Init */
    HAL_NVIC_SetPriority(USART2_IRQn, 5, 0);
    HAL_NVIC_EnableIRQ(USART2_IRQn);
  /* USER CODE BEGIN USART2_MspInit 1 */

  /* USER CODE END USART2_MspInit 1 */
  }

}

/**********************************************************************
* @brief UART MSP De-Initialization
* This function freeze the hardware resources used in this example
* @param huart: UART handle pointer
* @retval None
**********************************************************************/
void HAL_UART_MspDeInit(UART_HandleTypeDef* huart)
{
  if(huart->Instance==USART1)
  {
```

```
  /* USER CODE BEGIN USART1_MspDeInit 0 */

  /* USER CODE END USART1_MspDeInit 0 */
    /* Peripheral clock disable */
    __HAL_RCC_USART1_CLK_DISABLE();

    HAL_GPIO_DeInit(GPIOA, GPIO_PIN_9|GPIO_PIN_10);

    /* USART1 DMA DeInit */
    HAL_DMA_DeInit(huart->hdmarx);
    HAL_DMA_DeInit(huart->hdmatx);

    /* USART1 interrupt DeInit */
    HAL_NVIC_DisableIRQ(USART1_IRQn);
  /* USER CODE BEGIN USART1_MspDeInit 1 */

  /* USER CODE END USART1_MspDeInit 1 */
  }
  else if(huart->Instance==USART2)
  {
  /* USER CODE BEGIN USART2_MspDeInit 0 */

  /* USER CODE END USART2_MspDeInit 0 */
    /* Peripheral clock disable */
    __HAL_RCC_USART2_CLK_DISABLE();

    HAL_GPIO_DeInit(GPIOA, USART_TX_Pin|USART_RX_Pin);

    /* USART2 DMA DeInit */
    HAL_DMA_DeInit(huart->hdmarx);
    HAL_DMA_DeInit(huart->hdmatx);

    /* USART2 interrupt DeInit */
    HAL_NVIC_DisableIRQ(USART2_IRQn);
  /* USER CODE BEGIN USART2_MspDeInit 1 */

  /* USER CODE END USART2_MspDeInit 1 */
  }

}

/* USER CODE BEGIN 1 */

/* USER CODE END 1 */
```

5.37 C:/Users/Administrator/Documents/GitHub/ENIB_Robot_Mobile_Ros2/WORKSPACE_F411_uROS6/base_robot/Core/Src/stm32f4xx_hal_timebase_tim.c File Reference

HAL time base based on the hardware TIM.

```
#include "stm32f4xx_hal.h"
#include "stm32f4xx_hal_tim.h"
```

Functions

- HAL_StatusTypeDef HAL_InitTick (uint32_t TickPriority)

  This function configures the TIM1 as a time base source. The time source is configured to have 1ms time base with a dedicated Tick interrupt priority.
- void HAL_SuspendTick (void)

  Suspend Tick increment.
- void HAL_ResumeTick (void)

  Resume Tick increment.

Variables

- TIM_HandleTypeDef htim4

5.37.1 Detailed Description

HAL time base based on the hardware TIM.

Attention

Definition in file stm32f4xx_hal_timebase_tim.c.

5.37.2 Function Documentation

5.37.2.1 HAL_InitTick()

```
HAL_StatusTypeDef HAL_InitTick (
            uint32_t TickPriority )
```

This function configures the TIM1 as a time base source. The time source is configured to have 1ms time base with a dedicated Tick interrupt priority.

Note

> This function is called automatically at the beginning of program after reset by HAL_Init() or at any time when clock is configured, by HAL_RCC_ClockConfig().

Parameters

TickPriority	Tick interrupt priority.

Return values

HAL	status

Definition at line 41 of file stm32f4xx_hal_timebase_tim.c.

5.37.2.2 HAL_ResumeTick()

```
void HAL_ResumeTick (
            void )
```

Resume Tick increment.

Note

> Enable the tick increment by Enabling TIM1 update interrupt.

Parameters

None	

Return values

None	

Definition at line 122 of file stm32f4xx_hal_timebase_tim.c.

5.37.2.3 HAL_SuspendTick()

```
void HAL_SuspendTick (
             void  )
```

Suspend Tick increment.

Note

> Disable the tick increment by disabling TIM1 update interrupt.

Parameters

None	

Return values

None	

Definition at line 110 of file stm32f4xx_hal_timebase_tim.c.

5.37.3 Variable Documentation

5.37.3.1 htim4

```
TIM_HandleTypeDef htim4
```

Definition at line 28 of file stm32f4xx_hal_timebase_tim.c.

5.38 stm32f4xx_hal_timebase_tim.c

Go to the documentation of this file.

```
00001 /* USER CODE BEGIN Header */
00018 /* USER CODE END Header */
00019
00020 /* Includes ------------------------------------------------------------------*/
00021 #include "stm32f4xx_hal.h"
```

```
#include "stm32f4xx_hal_tim.h"

/* Private typedef -----------------------------------------------------------*/
/* Private define ------------------------------------------------------------*/
/* Private macro -------------------------------------------------------------*/
/* Private variables ---------------------------------------------------------*/
TIM_HandleTypeDef        htim4;
/* Private function prototypes -----------------------------------------------*/
/* Private functions ---------------------------------------------------------*/

HAL_StatusTypeDef HAL_InitTick(uint32_t TickPriority)
{
  RCC_ClkInitTypeDef    clkconfig;
  uint32_t              uwTimclock = 0U;

  uint32_t              uwPrescalerValue = 0U;
  uint32_t              pFLatency;
  HAL_StatusTypeDef     status;

  /* Enable TIM1 clock */
  __HAL_RCC_TIM4_CLK_ENABLE();

  /* Get clock configuration */
  HAL_RCC_GetClockConfig(&clkconfig, &pFLatency);

  /* Compute TIM1 clock */
  uwTimclock = 2*HAL_RCC_GetPCLK2Freq();

  /* Compute the prescaler value to have TIM1 counter clock equal to 1MHz */
  uwPrescalerValue = (uint32_t) ((uwTimclock / 1000000U) - 1U);

  /* Initialize TIM1 */
  htim4.Instance = TIM4;

  /* Initialize TIMx peripheral as follow:
  + Period = [(TIM1CLK/1000) - 1]. to have a (1/1000) s time base.
  + Prescaler = (uwTimclock/1000000 - 1) to have a 1MHz counter clock.
  + ClockDivision = 0
  + Counter direction = Up
  */
  htim4.Init.Period = (1000000U / 1000U) - 1U;
  htim4.Init.Prescaler = uwPrescalerValue;
  htim4.Init.ClockDivision = 0;
  htim4.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim4.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;

  status = HAL_TIM_Base_Init(&htim4);
  if (status == HAL_OK)
  {
    /* Start the TIM time Base generation in interrupt mode */
    status = HAL_TIM_Base_Start_IT(&htim4);
    if (status == HAL_OK)
    {
    /* Enable the TIM1 global Interrupt */
        HAL_NVIC_EnableIRQ(TIM4_IRQn);
      /* Configure the SysTick IRQ priority */
      if (TickPriority < (1UL « __NVIC_PRIO_BITS))
      {
        /* Configure the TIM IRQ priority */
        HAL_NVIC_SetPriority(TIM4_IRQn, TickPriority, 0U);
        uwTickPrio = TickPriority;
      }
      else
      {
        status = HAL_ERROR;
      }
    }
  }

 /* Return function status */
  return status;
}

void HAL_SuspendTick(void)
{
  /* Disable TIM1 update Interrupt */
  __HAL_TIM_DISABLE_IT(&htim4, TIM_IT_UPDATE);
}

void HAL_ResumeTick(void)
{
  /* Enable TIM1 Update interrupt */
  __HAL_TIM_ENABLE_IT(&htim4, TIM_IT_UPDATE);
}

```

5.39 stm32f4xx_it.c

```

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

extern DMA_HandleTypeDef hdma_usart1_rx;
extern DMA_HandleTypeDef hdma_usart1_tx;
extern DMA_HandleTypeDef hdma_usart2_rx;
extern DMA_HandleTypeDef hdma_usart2_tx;
extern UART_HandleTypeDef huart1;
extern UART_HandleTypeDef huart2;
extern TIM_HandleTypeDef htim4;
extern I2C_HandleTypeDef hi2c1;


void NMI_Handler(void)
{
  while (1)
  {
  }
}

void HardFault_Handler(void)
{

  while (1)
  {
  }
}

void MemManage_Handler(void)
{
  while (1)
  {
  }
}


void BusFault_Handler(void)
{
  while (1)
  {
  }
}

void UsageFault_Handler(void)
{
  while (1)
  {
  }
}

void DebugMon_Handler(void)
{
}

/******************************************************************************/
/* STM32F4xx Peripheral Interrupt Handlers                                    */
/* Add here the Interrupt Handlers for the used peripherals.                  */
/* For the available peripheral interrupt handler names,                      */
/* please refer to the startup file (startup_stm32f4xx.s).                   */
/******************************************************************************/

void DMA1_Stream5_IRQHandler(void)
{
  HAL_DMA_IRQHandler(&hdma_usart2_rx);
}


void DMA1_Stream6_IRQHandler(void)
{
  HAL_DMA_IRQHandler(&hdma_usart2_tx);
}

/*void TIM1_UP_TIM10_IRQHandler(void)
{
  HAL_TIM_IRQHandler(&htim1);
}*/

void TIM4_IRQHandler(void)
{
  HAL_TIM_IRQHandler(&htim4);
}

void USART1_IRQHandler(void)
{
  HAL_UART_IRQHandler(&huart1);
}

void USART2_IRQHandler(void)
{
  HAL_UART_IRQHandler(&huart2);
}

void DMA2_Stream2_IRQHandler(void)
{
  HAL_DMA_IRQHandler(&hdma_usart1_rx);
}

void DMA2_Stream7_IRQHandler(void)
{
  HAL_DMA_IRQHandler(&hdma_usart1_tx);
}




//=======================================================
//      ENCODER INDEX LEFT
//=======================================================
void EXTI15_10_IRQHandler(void)
{
  HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_10);
}
//=======================================================
//      ENCODER INDEX RIGHT
//=======================================================

void EXTI0_IRQHandler(void)
{
  HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_0);
}


void I2C1_EV_IRQHandler(void)
{
      HAL_I2C_EV_IRQHandler(&hi2c1);
}

void I2C1_ER_IRQHandler(void)
{
      HAL_I2C_ER_IRQHandler(&hi2c1);
}

```

5.40 C:/Users/Administrator/Documents/GitHub/ENIB_Robot_Mobile_Ros2/WORKSPACE_F411_uROS6/base_robot/Core/Src/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>
```

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

5.40.1 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

Definition in file syscalls.c.

5.40.2 Function Documentation

5.40.2.1 __attribute__()

```
__attribute__ (
              (weak)  )
```

Definition at line 65 of file syscalls.c.

5.40.2.2 __io_getchar()

```
int __io_getchar (
             void )  [extern]
```

Definition at line 36 of file syscalls.c.

5.40.2.3 _close()

```
int _close (
             int file )
```

Definition at line 88 of file syscalls.c.

5.40.2.4 _execve()

```
int _execve (
             char * name,
             char ** argv,
             char ** env )
```

Definition at line 151 of file syscalls.c.

5.40.2.5 _exit()

```
void _exit (
             int status )
```

Definition at line 59 of file syscalls.c.

5.40.2.6 _fork()

```
int _fork (
             void )
```

Definition at line 145 of file syscalls.c.

5.40.2.7 _fstat()

```
int _fstat (
             int file,
             struct stat * st )
```

Definition at line 94 of file syscalls.c.

5.40.2.8 _getpid()

```
int _getpid (
             void   )
```

Definition at line 48 of file syscalls.c.

5.40.2.9 _isatty()

```
int _isatty (
             int file )
```

Definition at line 100 of file syscalls.c.

5.40.2.10 _kill()

```
int _kill (
             int pid,
             int sig )
```

Definition at line 53 of file syscalls.c.

5.40.2.11 _link()

```
int _link (
             char * old,
             char * new )
```

Definition at line 139 of file syscalls.c.

5.40.2.12 _lseek()

```
int _lseek (
             int file,
             int ptr,
             int dir )
```

Definition at line 105 of file syscalls.c.

5.40.2.13 _open()

```
int _open (
             char * path,
             int flags,
              ...   )
```

Definition at line 110 of file syscalls.c.

5.40.2.14 _stat()

```
int _stat (
            char * file,
            struct stat * st )
```

Definition at line 133 of file syscalls.c.

5.40.2.15 _times()

```
int _times (
            struct tms * buf )
```

Definition at line 128 of file syscalls.c.

5.40.2.16 _unlink()

```
int _unlink (
            char * name )
```

Definition at line 122 of file syscalls.c.

5.40.2.17 _wait()

```
int _wait (
            int * status )
```

Definition at line 116 of file syscalls.c.

5.40.2.18 initialise_monitor_handles()

```
void initialise_monitor_handles ( )
```

Definition at line 44 of file syscalls.c.

5.40.3 Variable Documentation

5.40.3.1 environ

```
char** environ = __env
```

Definition at line 40 of file syscalls.c.

5.41 syscalls.c

Go to the documentation of this file.

```

/* Includes */
#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>


/* Variables */
extern int __io_putchar(int ch) __attribute__((weak));
extern int __io_getchar(void) __attribute__((weak));


char *__env[1] = { 0 };
char **environ = __env;


/* Functions */
void initialise_monitor_handles()
{
}

int _getpid(void)
{
    return 1;
}

int _kill(int pid, int sig)
{
    errno = EINVAL;
    return -1;
}

void _exit (int status)
{
    _kill(status, -1);
    while (1) {}        /* Make sure we hang here */
}

__attribute__((weak)) int _read(int file, char *ptr, int len)
{
    int DataIdx;

    for (DataIdx = 0; DataIdx < len; DataIdx++)
    {
        *ptr++ = __io_getchar();
    }

return len;
}

__attribute__((weak)) int _write(int file, char *ptr, int len)
{
    int DataIdx;

    for (DataIdx = 0; DataIdx < len; DataIdx++)
    {
        __io_putchar(*ptr++);
    }
    return len;
}

int _close(int file)
{
    return -1;
}


int _fstat(int file, struct stat *st)
{
    st->st_mode = S_IFCHR;
    return 0;
}

int _isatty(int file)
{
    return 1;
}
```

```

int _lseek(int file, int ptr, int dir)
{
    return 0;
}

int _open(char *path, int flags, ...)
{
    /* Pretend like we always fail */
    return -1;
}

int _wait(int *status)
{
    errno = ECHILD;
    return -1;
}

int _unlink(char *name)
{
    errno = ENOENT;
    return -1;
}

int _times(struct tms *buf)
{
    return -1;
}

int _stat(char *file, struct stat *st)
{
    st->st_mode = S_IFCHR;
    return 0;
}

int _link(char *old, char *new)
{
    errno = EMLINK;
    return -1;
}

int _fork(void)
{
    errno = EAGAIN;
    return -1;
}

int _execve(char *name, char **argv, char **env)
{
    errno = ENOMEM;
    return -1;
}
```

5.42 C:/Users/Administrator/Documents/GitHub/ENIB_Robot_Mobile_Ros2/WORKSPACE_F411_uROS6/base_robot/Core/Src/sysmem.c File Reference

STM32CubeIDE System Memory calls file.

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

Functions

- void ∗ _sbrk (ptrdiff_t incr)

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

Variables

- static uint8_t ∗ __sbrk_heap_end = NULL

5.42.1 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

Definition in file sysmem.c.

5.42.2 Function Documentation

5.42.2.1 _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

Definition at line 53 of file sysmem.c.

5.42.3 Variable Documentation

5.42.3.1 __sbrk_heap_end

```
uint8_t* __sbrk_heap_end = NULL  [static]
```

Pointer to the current high watermark of the heap usage

Definition at line 30 of file sysmem.c.

5.43 sysmem.c

Go to the documentation of this file.

```
00001
00023 /* Includes */
00024 #include <errno.h>
00025 #include <stdint.h>
00026
00030 static uint8_t *__sbrk_heap_end = NULL;
00031
00053 void *_sbrk(ptrdiff_t incr)
00054 {
00055   extern uint8_t _end; /* Symbol defined in the linker script */
00056   extern uint8_t _estack; /* Symbol defined in the linker script */
00057   extern uint32_t _Min_Stack_Size; /* Symbol defined in the linker script */
00058   const uint32_t stack_limit = (uint32_t)&_estack - (uint32_t)&_Min_Stack_Size;
00059   const uint8_t *max_heap = (uint8_t *)stack_limit;
00060   uint8_t *prev_heap_end;
00061
00062   /* Initialize heap end at first call */
00063   if (NULL == __sbrk_heap_end)
00064   {
00065     __sbrk_heap_end = &_end;
00066   }
00067
00068   /* Protect heap from growing into the reserved MSP stack */
00069   if (__sbrk_heap_end + incr > max_heap)
00070   {
00071     errno = ENOMEM;
00072     return (void *)-1;
00073   }
00074
00075   prev_heap_end = __sbrk_heap_end;
00076   __sbrk_heap_end += incr;
00077
00078   return (void *)prev_heap_end;
00079 }
```

5.44 systemclock.c

```
/*
 * systemclock.c
 *
 *  Created on: Mar 13, 2023
 *      Author: kerhoas
 */


#include "main.h"

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

  __HAL_RCC_PWR_CLK_ENABLE();
  __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);

  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
  RCC_OscInitStruct.HSEState = RCC_HSE_BYPASS;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
  RCC_OscInitStruct.PLL.PLLM = 8;
  RCC_OscInitStruct.PLL.PLLN = 432;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV6;
  RCC_OscInitStruct.PLL.PLLQ = 4;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }

  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_DIV4;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)
  {
    Error_Handler();
  }
}
```

5.45 util.c

```
#include "util.h"

//===================================================================
void num2str(char *s, unsigned int number, unsigned int base, unsigned int size, int sp)
{
        static char  hexChars[] = "0123456789ABCDEF";

        char *p=s;
        unsigned int cnt;
        unsigned int i;
        char tmp;

        // get digits
        do {
                *s++=hexChars[number % base];
        } while (number /= base);
        *s='\0';

        // reverse string
        cnt=s-p;
        for (i=0;i<cnt/2;i++) {
                tmp=p[i]; p[i] = p[cnt-i-1]; p[cnt-i-1]=tmp;
        }

        // add extra space
        if (cnt<size) {
                for (i=cnt;i==0;i--)
                        {p[i+size-cnt]=p[i];}
                if (sp) tmp=' '; else tmp='0';
                for (i=0;i<size-cnt;i++) p[i]=tmp;
        }
}

//===================================================================
unsigned int str2num(char *s, unsigned base)
{
    unsigned int u=0, d;
    char ch=*s++;
    while (ch) {
        if ((ch>='0') && (ch<='9')) d=ch-'0';
        else if ((base==16) && (ch>='A') && (ch<='F')) d=ch-'A'+10;
        else if ((base==16) && (ch>='a') && (ch<='f')) d=ch-'a'+10;
        else break;
        u=d+base*u;
        ch=*s++;
    }
    return u;
}

//===================================================================
void reverse(char *str, int len)
{
    int i=0, j=len-1, temp;
    while (i<j)
    {
        temp = str[i];
        str[i] = str[j];
        str[j] = temp;
        i++; j--;
    }
}

//==================================================================
int intToStr(int x, char str[], int d)
{
    int i = 0;
    while (x)
    {
        str[i++] = (x%10) + '0';
        x = x/10;
    }

    // If number of digits required is more, then
    // add 0s at the beginning
    while (i < d)
        str[i++] = '0';

    reverse(str, i);
    str[i] = '\0';
    return i;
}
//==================================================================
void float2str( char *res, float n, int afterpoint)
{
    // Extract integer part
    int ipart = (int)n;

    // Extract floating part
    float fpart = n - (float)ipart;

    // convert integer part to string
    int i = intToStr(ipart, res, 0);

    // check for display option after point
    if (afterpoint != 0)
    {
        res[i] = '.';  // add dot

        // Get the value of fraction part upto given no.
        // of points after dot. The third parameter is needed
        // to handle cases like 233.007
        fpart = fpart * (float)myPow(10.0, afterpoint);

        intToStr((int)fpart, res + i + 1, afterpoint);
    }
}
//==================================================================
double myPow(double x, int n) {
    unsigned int p = abs(n);
    double result = 1;
    while(p > 0)
    {
        if(p & 1) // if bit is set
        {
            result = result * x;
        }
        p = p » 1;
        x = x * x;
    }

    if(n < 0)
    {
        return 1/result;
    }
    return result;
}

//==================================================================
void flush_ch(char* ch, int ch_size)
{
    int i=0;
    for (i=0 ; i<ch_size ; i++)
    {
        ch[i]=0;

    }
}
//==================================================================

int size_ch(char* ch, int ch_size_max)
{

    int i=0;
    for (i=0 ; i<ch_size_max ; i++)
    {
        if (ch[i]==0)
```

```
            break;
    }

    return i;
}

//==================================================================
```

5.46 VL53L0X.c

```

#include "main.h"
#include <unistd.h>
    // Most of the functionality of this library is based on the VL53L0X API
// provided by ST (STSW-IMG005), and some of the explanatory comments are quoted
// or paraphrased from the API source code, API user manual (UM2039), and the
// VL53L0X datasheet.

#include <stdint.h>
#include "VL53L0X.h"
#include "drv_i2c.h"

//-----------------------------------------------------------
// Local variables within this file (private)
//-----------------------------------------------------------
uint8_t g_i2cAddr = ADDRESS_DEFAULT;
uint8_t g_stopVariable; // read by init and used when starting measurement; is StopVariable field of VL53L0X_DevData_t structure in API


//-----------------------------------------------------------
// Locally used functions (private)
//-----------------------------------------------------------
uint8_t performSingleRefCalibration(uint8_t vhv_init_byte);
//-----------------------------------------------------------
// I2C communication Functions
//-----------------------------------------------------------
// Write an 8-bit register
void writeReg(uint8_t reg, uint8_t value) {
    i2c1_WriteRegBuffer(0x53,reg,&value,1);

}

// Write a 16-bit register
void writeReg16Bit(uint8_t reg, uint16_t value){
    uint8_t tab[2];
    tab[0]= ((value » 8));
    tab[1] = ((value ) & 0xFF);
    i2c1_WriteRegBuffer(0x53,reg,tab,2);

}

// Write a 32-bit register
void writeReg32Bit(uint8_t reg, uint32_t value){
    uint8_t tab[4];
        tab[3]= ((value » 24) & 0xFF);
        tab[2]= ((value » 16) & 0xFF);
        tab[1]= ((value » 8) & 0xFF);
        tab[0] = ((value ) & 0xFF);
        i2c1_WriteRegBuffer(0x53,reg,tab,4);
}

// Read an 8-bit register
uint8_t readReg(uint8_t reg) {
    uint8_t value=0;
    i2c1_ReadRegBuffer(0x53,reg,&value,1);
    return value;
}

// Read a 16-bit register
uint16_t readReg16Bit(uint8_t reg) {
    uint8_t tab[2];
    i2c1_ReadRegBuffer(0x53,reg,tab,2);
    uint16_t value= ((uint16_t)tab[0] « 8) | (uint16_t)tab[1];
    return value;
}

// Read a 32-bit register
uint32_t readReg32Bit(uint8_t reg) {
  uint8_t tab[4];
  i2c1_ReadRegBuffer(0x53,reg,tab,4);
  uint32_t value= (tab[3] « 24) | (tab[2] « 16 ) | (tab[1] « 8) | tab[0];
  return value;
}

// Write an arbitrary number of bytes from the given array to the sensor,
// starting at the given register
void writeMulti(uint8_t reg, uint8_t const *src, uint8_t count){

  while ( count-- > 0 ) {
    i2c1_WriteRegBuffer(0x53,reg,(uint8_t *)src,1);
  }
}



// Public Methods //////////////////////////////////////////////////////////////

void setAddress(uint8_t new_addr) {
  writeReg( I2C_SLAVE_DEVICE_ADDRESS, (new_addr»1) & 0x7F );
  g_i2cAddr = new_addr;
}

uint8_t getAddress() {
  return g_i2cAddr;
}

// Initialize sensor using sequence based on VL53L0X_DataInit(),
// VL53L0X_StaticInit(), and VL53L0X_PerformRefCalibration().
// This function does not perform reference SPAD calibration
// (VL53L0X_PerformRefSpadManagement()), since the API user manual says that it
// is performed by ST on the bare modules; it seems like that should work well
// enough unless a cover glass is added.
// If io_2v8 (optional) is true or not given, the sensor is configured for 2V8
// mode.
uint8_t initVL53L0X( ){
  // VL53L0X_DataInit() begin

  // "Set I2C standard mode"
  writeReg(0x88, 0x00);

  writeReg(0x80, 0x01);
  writeReg(0xFF, 0x01);
  writeReg(0x00, 0x00);
  g_stopVariable = readReg(0x91);
  writeReg(0x00, 0x01);
  writeReg(0xFF, 0x00);
  writeReg(0x80, 0x00);

  // disable SIGNAL_RATE_MSRC (bit 1) and SIGNAL_RATE_PRE_RANGE (bit 4) limit checks
  writeReg(MSRC_CONFIG_CONTROL, readReg(MSRC_CONFIG_CONTROL) | 0x12);

  // set final range signal rate limit to 0.25 MCPS (million counts per second)
  setSignalRateLimit(0.25);

  writeReg(SYSTEM_SEQUENCE_CONFIG, 0xFF);

  // VL53L0X_DataInit() end

  // VL53L0X_StaticInit() begin

  // The SPAD map (RefGoodSpadMap) is read by VL53L0X_get_info_from_device() in
  // the API, but the same data seems to be more easily readable from
  // GLOBAL_CONFIG_SPAD_ENABLES_REF_0 through _6, so read it from there

  // -- VL53L0X_set_reference_spads() begin (assume NVM values are valid)

  writeReg(0xFF, 0x01);
  writeReg(DYNAMIC_SPAD_REF_EN_START_OFFSET, 0x00);
  writeReg(DYNAMIC_SPAD_NUM_REQUESTED_REF_SPAD, 0x2C);
  writeReg(0xFF, 0x00);
  writeReg(GLOBAL_CONFIG_REF_EN_START_SELECT, 0xB4);


  // -- VL53L0X_set_reference_spads() end

  // -- VL53L0X_load_tuning_settings() begin
  // DefaultTuningSettings from vl53l0x_tuning.h

  writeReg(0xFF, 0x01);
  writeReg(0x00, 0x00);

  writeReg(0xFF, 0x00);
  writeReg(0x09, 0x00);
  writeReg(0x10, 0x00);
  writeReg(0x11, 0x00);

  writeReg(0x24, 0x01);
  writeReg(0x25, 0xFF);
  writeReg(0x75, 0x00);

  writeReg(0xFF, 0x01);
  writeReg(0x4E, 0x2C);
  writeReg(0x48, 0x00);
  writeReg(0x30, 0x20);

  writeReg(0xFF, 0x00);
  writeReg(0x30, 0x09);
  writeReg(0x54, 0x00);
  writeReg(0x31, 0x04);
  writeReg(0x32, 0x03);
  writeReg(0x40, 0x83);
  writeReg(0x46, 0x25);
  writeReg(0x60, 0x00);
  writeReg(0x27, 0x00);
  writeReg(0x50, 0x06);
  writeReg(0x51, 0x00);
  writeReg(0x52, 0x96);
  writeReg(0x56, 0x08);
  writeReg(0x57, 0x30);
  writeReg(0x61, 0x00);
  writeReg(0x62, 0x00);
  writeReg(0x64, 0x00);
  writeReg(0x65, 0x00);
  writeReg(0x66, 0xA0);

  writeReg(0xFF, 0x01);
  writeReg(0x22, 0x32);
  writeReg(0x47, 0x14);
  writeReg(0x49, 0xFF);
  writeReg(0x4A, 0x00);

  writeReg(0xFF, 0x00);
  writeReg(0x7A, 0x0A);
  writeReg(0x7B, 0x00);
  writeReg(0x78, 0x21);

  writeReg(0xFF, 0x01);
  writeReg(0x23, 0x34);
  writeReg(0x42, 0x00);
  writeReg(0x44, 0xFF);
  writeReg(0x45, 0x26);
  writeReg(0x46, 0x05);
  writeReg(0x40, 0x40);
  writeReg(0x0E, 0x06);
  writeReg(0x20, 0x1A);
  writeReg(0x43, 0x40);

  writeReg(0xFF, 0x00);
  writeReg(0x34, 0x03);
  writeReg(0x35, 0x44);

  writeReg(0xFF, 0x01);
  writeReg(0x31, 0x04);
  writeReg(0x4B, 0x09);
  writeReg(0x4C, 0x05);
  writeReg(0x4D, 0x04);

  writeReg(0xFF, 0x00);
  writeReg(0x44, 0x00);
  writeReg(0x45, 0x20);
  writeReg(0x47, 0x08);
  writeReg(0x48, 0x28);
  writeReg(0x67, 0x00);
  writeReg(0x70, 0x04);
  writeReg(0x71, 0x01);
  writeReg(0x72, 0xFE);
  writeReg(0x76, 0x00);
  writeReg(0x77, 0x00);

  writeReg(0xFF, 0x01);
  writeReg(0x0D, 0x01);

  writeReg(0xFF, 0x00);
  writeReg(0x80, 0x01);
  writeReg(0x01, 0xF8);

  writeReg(0xFF, 0x01);
  writeReg(0x8E, 0x01);
  writeReg(0x00, 0x01);
  writeReg(0xFF, 0x00);
  writeReg(0x80, 0x00);

  // -- VL53L0X_load_tuning_settings() end

  // "Set interrupt config to new sample ready"
  // -- VL53L0X_SetGpioConfig() begin

  writeReg(SYSTEM_INTERRUPT_CONFIG_GPIO, 0x04);
  writeReg(GPIO_HV_MUX_ACTIVE_HIGH, readReg(GPIO_HV_MUX_ACTIVE_HIGH) & ~0x10); // active low
  writeReg(SYSTEM_INTERRUPT_CLEAR, 0x01);

  // -- VL53L0X_SetGpioConfig() end



  // "Disable MSRC and TCC by default"
  // MSRC = Minimum Signal Rate Check
  // TCC = Target CentreCheck
  // -- VL53L0X_SetSequenceStepEnable() begin

  writeReg(SYSTEM_SEQUENCE_CONFIG, 0xE8);

  // -- VL53L0X_SetSequenceStepEnable() end



  // VL53L0X_StaticInit() end

  // VL53L0X_PerformRefCalibration() begin (VL53L0X_perform_ref_calibration())

  // -- VL53L0X_perform_vhv_calibration() begin

  writeReg(SYSTEM_SEQUENCE_CONFIG, 0x01);
  if (performSingleRefCalibration(0x40)) { return 1; }

  // -- VL53L0X_perform_vhv_calibration() end

  // -- VL53L0X_perform_phase_calibration() begin

  writeReg(SYSTEM_SEQUENCE_CONFIG, 0x02);
  if (performSingleRefCalibration(0x00)) { return 1; }

  // -- VL53L0X_perform_phase_calibration() end

  // "restore the previous Sequence Config"
  writeReg(SYSTEM_SEQUENCE_CONFIG, 0xE8);

  // VL53L0X_PerformRefCalibration() end

  return 0;
}

// Set the return signal rate limit check value in units of MCPS (mega counts
// per second). "This represents the amplitude of the signal reflected from the
// target and detected by the device"; setting this limit presumably determines
// the minimum measurement necessary for the sensor to report a valid reading.
// Setting a lower limit increases the potential range of the sensor but also
// seems to increase the likelihood of getting an inaccurate reading because of
// unwanted reflections from objects other than the intended target.
// Defaults to 0.25 MCPS as initialized by the ST API and this library.
uint8_t setSignalRateLimit(float limit_Mcps)
{
  if (limit_Mcps < 0 || limit_Mcps > 511.99) { return false; }

  // Q9.7 fixed point format (9 integer bits, 7 fractional bits)
  writeReg16Bit(FINAL_RANGE_CONFIG_MIN_COUNT_RATE_RTN_LIMIT, limit_Mcps * (1 « 7));
  return 0;
}

// Get the return signal rate limit check value in MCPS
float getSignalRateLimit(void)
{
  return (float)readReg16Bit(FINAL_RANGE_CONFIG_MIN_COUNT_RATE_RTN_LIMIT) / (1 « 7);
}





// Performs a single-shot range measurement and returns the reading in
// millimeters
// based on VL53L0X_PerformSingleRangingMeasurement()
// extraStats provides additional info for this measurment. Set to 0 if not needed.
uint16_t readRangeSingleMillimeters( /*statInfo_t *extraStats */) {
  writeReg(0x80, 0x01);
  writeReg(0xFF, 0x01);
  writeReg(0x00, 0x00);
  writeReg(0x91, g_stopVariable);
  writeReg(0x00, 0x01);
  writeReg(0xFF, 0x00);
  writeReg(0x80, 0x00);
  writeReg(SYSRANGE_START, 0x01);

  uint16_t temp;

    // assumptions: Linearity Corrective Gain is 1000 (default);
    // fractional ranging is not enabled
      temp = readReg16Bit(RESULT_RANGE_STATUS + 10);

      temp+=0;

  writeReg(SYSTEM_INTERRUPT_CLEAR, 0x01);
  return temp;
}


// based on VL53L0X_perform_single_ref_calibration()
uint8_t performSingleRefCalibration(uint8_t vhv_init_byte)
{
  writeReg(SYSRANGE_START, 0x01 | vhv_init_byte); // VL53L0X_REG_SYSRANGE_MODE_START_STOP



  writeReg(SYSTEM_INTERRUPT_CLEAR, 0x01);

  writeReg(SYSRANGE_START, 0x00);

  return 0;
}
```

Index