LAMPIRAN C: KODE SUMBER C UNTUK FIRMWARE MOTOR CONTROLLER

Seluruh file ini dapat di akses di URL:

<https://github.com/mekatronik-achmadi/tugas_akhir/tree/master/software/rtos>

Seluruh kode sumber ini dibangun dengan operating system ubuntu atau debian. Kode sumber ini dibangun berdasarkan pustaka ChibiOS/RT yang merupakan pustaka untuk membangun firmware pada mikrokontroller berbasis ARM-Cortex M-series. Kode sumber yang terlampir disini hanyalah kode sumber yang dibuat dalam perancangan ini dan bukan termasuk kode bawaan dari pustaka ChibiOS/RT. File-file tersebut adalah:

1. halconf.h
2. mcuconf.h
3. robot\_gpt.h
4. robot\_led.h
5. robot\_pal.h
6. robot\_shell.h
7. srcconf.h
8. main.c
9. robot\_gpt.c
10. robot\_led.h
11. robot\_pal.c
12. robot\_shell.c
13. Makefile

Untuk dapat mengkompilasi kode sumber ini maka dibutuhkan paket berikut:

1. gcc-arm-none-eabi
2. libnewlib-arm-none-eabi
3. libnewlib-dev
4. build-essential

Untuk file kode sumber selain yang terlampir disini maka file tersebut merupakan bagian dari pustaka ChibiOS/RT yang digunakan tanpa ada perubahan.Untuk mengkompilasinya tinggal memanggil program make.

* + - 1. File halconf.h

/\*\*

\* **@file** templates/halconf.h

\* **@brief** HAL configuration header.

\* @details HAL configuration file, this file allows to enable or disable the

\* various device drivers from your application. You may also use

\* this file in order to override the device drivers default settings.

\*

\* **@addtogroup** HAL\_CONF

\* **@{**

\*/

#ifndef \_HALCONF\_H\_

#define \_HALCONF\_H\_

#include "mcuconf.h"

/\*\*

\* **@brief** Enables the TM subsystem.

\*/

#if !defined(HAL\_USE\_TM) || defined(\_\_DOXYGEN\_\_)

#define HAL\_USE\_TM TRUE

#endif

/\*\*

\* **@brief** Enables the PAL subsystem.

\*/

#if !defined(HAL\_USE\_PAL) || defined(\_\_DOXYGEN\_\_)

#define HAL\_USE\_PAL TRUE

#endif

/\*\*

\* **@brief** Enables the ADC subsystem.

\*/

#if !defined(HAL\_USE\_ADC) || defined(\_\_DOXYGEN\_\_)

#define HAL\_USE\_ADC FALSE

#endif

/\*\*

\* **@brief** Enables the CAN subsystem.

\*/

#if !defined(HAL\_USE\_CAN) || defined(\_\_DOXYGEN\_\_)

#define HAL\_USE\_CAN FALSE

#endif

/\*\*

\* **@brief** Enables the EXT subsystem.

\*/

#if !defined(HAL\_USE\_EXT) || defined(\_\_DOXYGEN\_\_)

#define HAL\_USE\_EXT FALSE

#endif

/\*\*

\* **@brief** Enables the GPT subsystem.

\*/

#if !defined(HAL\_USE\_GPT) || defined(\_\_DOXYGEN\_\_)

#define HAL\_USE\_GPT TRUE

#endif

/\*\*

\* **@brief** Enables the I2C subsystem.

\*/

#if !defined(HAL\_USE\_I2C) || defined(\_\_DOXYGEN\_\_)

#define HAL\_USE\_I2C FALSE

#endif

/\*\*

\* **@brief** Enables the ICU subsystem.

\*/

#if !defined(HAL\_USE\_ICU) || defined(\_\_DOXYGEN\_\_)

#define HAL\_USE\_ICU FALSE

#endif

/\*\*

\* **@brief** Enables the MAC subsystem.

\*/

#if !defined(HAL\_USE\_MAC) || defined(\_\_DOXYGEN\_\_)

#define HAL\_USE\_MAC FALSE

#endif

/\*\*

\* **@brief** Enables the MMC\_SPI subsystem.

\*/

#if !defined(HAL\_USE\_MMC\_SPI) || defined(\_\_DOXYGEN\_\_)

#define HAL\_USE\_MMC\_SPI FALSE

#endif

/\*\*

\* **@brief** Enables the PWM subsystem.

\*/

#if !defined(HAL\_USE\_PWM) || defined(\_\_DOXYGEN\_\_)

#define HAL\_USE\_PWM FALSE

#endif

/\*\*

\* **@brief** Enables the RTC subsystem.

\*/

#if !defined(HAL\_USE\_RTC) || defined(\_\_DOXYGEN\_\_)

#define HAL\_USE\_RTC FALSE

#endif

/\*\*

\* **@brief** Enables the SDC subsystem.

\*/

#if !defined(HAL\_USE\_SDC) || defined(\_\_DOXYGEN\_\_)

#define HAL\_USE\_SDC FALSE

#endif

/\*\*

\* **@brief** Enables the SERIAL subsystem.

\*/

#if !defined(HAL\_USE\_SERIAL) || defined(\_\_DOXYGEN\_\_)

#define HAL\_USE\_SERIAL TRUE

#endif

/\*\*

\* **@brief** Enables the SERIAL over USB subsystem.

\*/

#if !defined(HAL\_USE\_SERIAL\_USB) || defined(\_\_DOXYGEN\_\_)

#define HAL\_USE\_SERIAL\_USB FALSE

#endif

/\*\*

\* **@brief** Enables the SPI subsystem.

\*/

#if !defined(HAL\_USE\_SPI) || defined(\_\_DOXYGEN\_\_)

#define HAL\_USE\_SPI FALSE

#endif

/\*\*

\* **@brief** Enables the UART subsystem.

\*/

#if !defined(HAL\_USE\_UART) || defined(\_\_DOXYGEN\_\_)

#define HAL\_USE\_UART FALSE

#endif

/\*\*

\* **@brief** Enables the USB subsystem.

\*/

#if !defined(HAL\_USE\_USB) || defined(\_\_DOXYGEN\_\_)

#define HAL\_USE\_USB FALSE

#endif

/\*===========================================================================\*/

/\* ADC driver related settings. \*/

/\*===========================================================================\*/

/\*\*

\* **@brief** Enables synchronous APIs.

\* **@note** Disabling this option saves both code and data space.

\*/

#if !defined(ADC\_USE\_WAIT) || defined(\_\_DOXYGEN\_\_)

#define ADC\_USE\_WAIT TRUE

#endif

/\*\*

\* **@brief** Enables the **@p** adcAcquireBus() and **@p** adcReleaseBus() APIs.

\* **@note** Disabling this option saves both code and data space.

\*/

#if !defined(ADC\_USE\_MUTUAL\_EXCLUSION) || defined(\_\_DOXYGEN\_\_)

#define ADC\_USE\_MUTUAL\_EXCLUSION TRUE

#endif

/\*===========================================================================\*/

/\* CAN driver related settings. \*/

/\*===========================================================================\*/

/\*\*

\* **@brief** Sleep mode related APIs inclusion switch.

\*/

#if !defined(CAN\_USE\_SLEEP\_MODE) || defined(\_\_DOXYGEN\_\_)

#define CAN\_USE\_SLEEP\_MODE TRUE

#endif

/\*===========================================================================\*/

/\* I2C driver related settings. \*/

/\*===========================================================================\*/

/\*\*

\* **@brief** Enables the mutual exclusion APIs on the I2C bus.

\*/

#if !defined(I2C\_USE\_MUTUAL\_EXCLUSION) || defined(\_\_DOXYGEN\_\_)

#define I2C\_USE\_MUTUAL\_EXCLUSION TRUE

#endif

/\*===========================================================================\*/

/\* MAC driver related settings. \*/

/\*===========================================================================\*/

/\*\*

\* **@brief** Enables an event sources for incoming packets.

\*/

#if !defined(MAC\_USE\_ZERO\_COPY) || defined(\_\_DOXYGEN\_\_)

#define MAC\_USE\_ZERO\_COPY FALSE

#endif

/\*\*

\* **@brief** Enables an event sources for incoming packets.

\*/

#if !defined(MAC\_USE\_EVENTS) || defined(\_\_DOXYGEN\_\_)

#define MAC\_USE\_EVENTS TRUE

#endif

/\*===========================================================================\*/

/\* MMC\_SPI driver related settings. \*/

/\*===========================================================================\*/

/\*\*

\* **@brief** Delays insertions.

\* @details If enabled this options inserts delays into the MMC waiting

\* routines releasing some extra CPU time for the threads with

\* lower priority, this may slow down the driver a bit however.

\* This option is recommended also if the SPI driver does not

\* use a DMA channel and heavily loads the CPU.

\*/

#if !defined(MMC\_NICE\_WAITING) || defined(\_\_DOXYGEN\_\_)

#define MMC\_NICE\_WAITING TRUE

#endif

/\*===========================================================================\*/

/\* SDC driver related settings. \*/

/\*===========================================================================\*/

/\*\*

\* **@brief** Number of initialization attempts before rejecting the card.

\* **@note** Attempts are performed at 10mS intervals.

\*/

#if !defined(SDC\_INIT\_RETRY) || defined(\_\_DOXYGEN\_\_)

#define SDC\_INIT\_RETRY 100

#endif

/\*\*

\* **@brief** Include support for MMC cards.

\* **@note** MMC support is not yet implemented so this option must be kept

\* at **@p** FALSE.

\*/

#if !defined(SDC\_MMC\_SUPPORT) || defined(\_\_DOXYGEN\_\_)

#define SDC\_MMC\_SUPPORT FALSE

#endif

/\*\*

\* **@brief** Delays insertions.

\* @details If enabled this options inserts delays into the MMC waiting

\* routines releasing some extra CPU time for the threads with

\* lower priority, this may slow down the driver a bit however.

\*/

#if !defined(SDC\_NICE\_WAITING) || defined(\_\_DOXYGEN\_\_)

#define SDC\_NICE\_WAITING TRUE

#endif

/\*===========================================================================\*/

/\* SERIAL driver related settings. \*/

/\*===========================================================================\*/

/\*\*

\* **@brief** Default bit rate.

\* @details Configuration parameter, this is the baud rate selected for the

\* default configuration.

\*/

#if !defined(SERIAL\_DEFAULT\_BITRATE) || defined(\_\_DOXYGEN\_\_)

#define SERIAL\_DEFAULT\_BITRATE 38400

#endif

/\*\*

\* **@brief** Serial buffers size.

\* @details Configuration parameter, you can change the depth of the queue

\* buffers depending on the requirements of your application.

\* **@note** The default is 64 bytes for both the transmission and receive

\* buffers.

\*/

#if !defined(SERIAL\_BUFFERS\_SIZE) || defined(\_\_DOXYGEN\_\_)

#define SERIAL\_BUFFERS\_SIZE 32

#endif

/\*===========================================================================\*/

/\* SPI driver related settings. \*/

/\*===========================================================================\*/

/\*\*

\* **@brief** Enables synchronous APIs.

\* **@note** Disabling this option saves both code and data space.

\*/

#if !defined(SPI\_USE\_WAIT) || defined(\_\_DOXYGEN\_\_)

#define SPI\_USE\_WAIT TRUE

#endif

/\*\*

\* **@brief** Enables the **@p** spiAcquireBus() and **@p** spiReleaseBus() APIs.

\* **@note** Disabling this option saves both code and data space.

\*/

#if !defined(SPI\_USE\_MUTUAL\_EXCLUSION) || defined(\_\_DOXYGEN\_\_)

#define SPI\_USE\_MUTUAL\_EXCLUSION TRUE

#endif

#endif /\* \_HALCONF\_H\_ \*/

/\*\* **@}** \*/

* + - 1. File mcuconf.h

#define STM32F103\_MCUCONF

/\*

\* STM32F103 drivers configuration.

\* The following settings override the default settings present in

\* the various device driver implementation headers.

\* Note that the settings for each driver only have effect if the whole

\* driver is enabled in halconf.h.

\*

\* IRQ priorities:

\* 15...0 Lowest...Highest.

\*

\* DMA priorities:

\* 0...3 Lowest...Highest.

\*/

/\*

\* HAL driver system settings.

\*/

#define STM32\_NO\_INIT FALSE

#define STM32\_HSI\_ENABLED TRUE

#define STM32\_LSI\_ENABLED FALSE

#define STM32\_HSE\_ENABLED TRUE

#define STM32\_LSE\_ENABLED FALSE

#define STM32\_SW STM32\_SW\_PLL

#define STM32\_PLLSRC STM32\_PLLSRC\_HSE

#define STM32\_PLLXTPRE STM32\_PLLXTPRE\_DIV1

#define STM32\_PLLMUL\_VALUE 9

#define STM32\_HPRE STM32\_HPRE\_DIV1

#define STM32\_PPRE1 STM32\_PPRE1\_DIV2

#define STM32\_PPRE2 STM32\_PPRE2\_DIV2

#define STM32\_ADCPRE STM32\_ADCPRE\_DIV4

#define STM32\_USB\_CLOCK\_REQUIRED TRUE

#define STM32\_USBPRE STM32\_USBPRE\_DIV1P5

#define STM32\_MCOSEL STM32\_MCOSEL\_NOCLOCK

#define STM32\_RTCSEL STM32\_RTCSEL\_HSEDIV

#define STM32\_PVD\_ENABLE FALSE

#define STM32\_PLS STM32\_PLS\_LEV0

/\*

\* ADC driver system settings.

\*/

#define STM32\_ADC\_USE\_ADC1 FALSE

#define STM32\_ADC\_ADC1\_DMA\_PRIORITY 2

#define STM32\_ADC\_ADC1\_IRQ\_PRIORITY 6

/\*

\* CAN driver system settings.

\*/

#define STM32\_CAN\_USE\_CAN1 FALSE

#define STM32\_CAN\_CAN1\_IRQ\_PRIORITY 11

/\*

\* EXT driver system settings.

\*/

#define STM32\_EXT\_EXTI0\_IRQ\_PRIORITY 6

#define STM32\_EXT\_EXTI1\_IRQ\_PRIORITY 6

#define STM32\_EXT\_EXTI2\_IRQ\_PRIORITY 6

#define STM32\_EXT\_EXTI3\_IRQ\_PRIORITY 6

#define STM32\_EXT\_EXTI4\_IRQ\_PRIORITY 6

#define STM32\_EXT\_EXTI5\_9\_IRQ\_PRIORITY 6

#define STM32\_EXT\_EXTI10\_15\_IRQ\_PRIORITY 6

#define STM32\_EXT\_EXTI16\_IRQ\_PRIORITY 6

#define STM32\_EXT\_EXTI17\_IRQ\_PRIORITY 6

#define STM32\_EXT\_EXTI18\_IRQ\_PRIORITY 6

#define STM32\_EXT\_EXTI19\_IRQ\_PRIORITY 6

/\*

\* GPT driver system settings.

\*/

#define STM32\_GPT\_USE\_TIM1 FALSE

#define STM32\_GPT\_USE\_TIM2 FALSE

#define STM32\_GPT\_USE\_TIM3 TRUE

#define STM32\_GPT\_USE\_TIM4 FALSE

#define STM32\_GPT\_USE\_TIM5 FALSE

#define STM32\_GPT\_USE\_TIM8 FALSE

#define STM32\_GPT\_TIM1\_IRQ\_PRIORITY 7

#define STM32\_GPT\_TIM2\_IRQ\_PRIORITY 7

#define STM32\_GPT\_TIM3\_IRQ\_PRIORITY 7

#define STM32\_GPT\_TIM4\_IRQ\_PRIORITY 7

#define STM32\_GPT\_TIM5\_IRQ\_PRIORITY 7

#define STM32\_GPT\_TIM8\_IRQ\_PRIORITY 7

/\*

\* I2C driver system settings.

\*/

#define STM32\_I2C\_USE\_I2C1 FALSE

#define STM32\_I2C\_USE\_I2C2 FALSE

#define STM32\_I2C\_I2C1\_IRQ\_PRIORITY 5

#define STM32\_I2C\_I2C2\_IRQ\_PRIORITY 5

#define STM32\_I2C\_I2C1\_DMA\_PRIORITY 3

#define STM32\_I2C\_I2C2\_DMA\_PRIORITY 3

#define STM32\_I2C\_I2C1\_DMA\_ERROR\_HOOK() chSysHalt()

#define STM32\_I2C\_I2C2\_DMA\_ERROR\_HOOK() chSysHalt()

/\*

\* ICU driver system settings.

\*/

#define STM32\_ICU\_USE\_TIM1 FALSE

#define STM32\_ICU\_USE\_TIM2 FALSE

#define STM32\_ICU\_USE\_TIM3 FALSE

#define STM32\_ICU\_USE\_TIM4 FALSE

#define STM32\_ICU\_USE\_TIM5 FALSE

#define STM32\_ICU\_USE\_TIM8 FALSE

#define STM32\_ICU\_TIM1\_IRQ\_PRIORITY 7

#define STM32\_ICU\_TIM2\_IRQ\_PRIORITY 7

#define STM32\_ICU\_TIM3\_IRQ\_PRIORITY 7

#define STM32\_ICU\_TIM4\_IRQ\_PRIORITY 7

#define STM32\_ICU\_TIM5\_IRQ\_PRIORITY 7

#define STM32\_ICU\_TIM8\_IRQ\_PRIORITY 7

/\*

\* PWM driver system settings.

\*/

#define STM32\_PWM\_USE\_ADVANCED FALSE

#define STM32\_PWM\_USE\_TIM1 FALSE

#define STM32\_PWM\_USE\_TIM2 FALSE

#define STM32\_PWM\_USE\_TIM3 FALSE

#define STM32\_PWM\_USE\_TIM4 FALSE

#define STM32\_PWM\_USE\_TIM5 FALSE

#define STM32\_PWM\_USE\_TIM8 FALSE

#define STM32\_PWM\_TIM1\_IRQ\_PRIORITY 7

#define STM32\_PWM\_TIM2\_IRQ\_PRIORITY 7

#define STM32\_PWM\_TIM3\_IRQ\_PRIORITY 7

#define STM32\_PWM\_TIM4\_IRQ\_PRIORITY 7

#define STM32\_PWM\_TIM5\_IRQ\_PRIORITY 7

#define STM32\_PWM\_TIM8\_IRQ\_PRIORITY 7

/\*

\* RTC driver system settings.

\*/

#define STM32\_RTC\_IRQ\_PRIORITY 15

/\*

\* SERIAL driver system settings.

\*/

#define STM32\_SERIAL\_USE\_USART1 TRUE

#define STM32\_SERIAL\_USE\_USART2 FALSE

#define STM32\_SERIAL\_USE\_USART3 FALSE

#define STM32\_SERIAL\_USE\_UART4 FALSE

#define STM32\_SERIAL\_USE\_UART5 FALSE

#define STM32\_SERIAL\_USART1\_PRIORITY 12

#define STM32\_SERIAL\_USART2\_PRIORITY 12

#define STM32\_SERIAL\_USART3\_PRIORITY 12

#define STM32\_SERIAL\_UART4\_PRIORITY 12

#define STM32\_SERIAL\_UART5\_PRIORITY 12

/\*

\* SPI driver system settings.

\*/

#define STM32\_SPI\_USE\_SPI1 FALSE

#define STM32\_SPI\_USE\_SPI2 FALSE

#define STM32\_SPI\_USE\_SPI3 FALSE

#define STM32\_SPI\_SPI1\_DMA\_PRIORITY 1

#define STM32\_SPI\_SPI2\_DMA\_PRIORITY 1

#define STM32\_SPI\_SPI3\_DMA\_PRIORITY 1

#define STM32\_SPI\_SPI1\_IRQ\_PRIORITY 10

#define STM32\_SPI\_SPI2\_IRQ\_PRIORITY 10

#define STM32\_SPI\_SPI3\_IRQ\_PRIORITY 10

#define STM32\_SPI\_DMA\_ERROR\_HOOK(spip) chSysHalt()

/\*

\* UART driver system settings.

\*/

#define STM32\_UART\_USE\_USART1 FALSE

#define STM32\_UART\_USE\_USART2 FALSE

#define STM32\_UART\_USE\_USART3 FALSE

#define STM32\_UART\_USART1\_IRQ\_PRIORITY 12

#define STM32\_UART\_USART2\_IRQ\_PRIORITY 12

#define STM32\_UART\_USART3\_IRQ\_PRIORITY 12

#define STM32\_UART\_USART1\_DMA\_PRIORITY 0

#define STM32\_UART\_USART2\_DMA\_PRIORITY 0

#define STM32\_UART\_USART3\_DMA\_PRIORITY 0

#define STM32\_UART\_DMA\_ERROR\_HOOK(uartp) chSysHalt()

/\*

\* USB driver system settings.

\*/

#define STM32\_USB\_USE\_USB1 FALSE

#define STM32\_USB\_LOW\_POWER\_ON\_SUSPEND FALSE

#define STM32\_USB\_USB1\_HP\_IRQ\_PRIORITY 13

#define STM32\_USB\_USB1\_LP\_IRQ\_PRIORITY 14

* + - 1. File robot\_gpt.h

#ifndef ROBOT\_GPT

#define ROBOT\_GPT

#define F\_GPT 100000

void Gpt\_Setup**(**void**);**

#endif

* + - 1. File robot\_led.h

#ifndef ROBOT\_LED

#define ROBOT\_LED

#define L1 0

#define L2 1

#define L3 2

#define L4 3

#define L5 4

#define L6 5

#define L7 6

#define L8 7

#define L0 8

#define PORT\_LED GPIOA

void Led\_Setup**(**void**);**

#endif

1. File robot\_pal.h

#ifndef ROBOT\_PAL

#define ROBOT\_PAL

#define O1 0

#define O2 1

#define O3 2

#define O4 3

#define O5 4

#define O6 5

#define O7 6

#define O8 7

#define O0 8

#define PORT\_PAL GPIOC

#define LEFT\_A O1

#define LEFT\_B O2

#define LEFT\_I L1

#define RIGHT\_A O3

#define RIGHT\_B O4

#define RIGHT\_I L2

void Pal\_Setup**(**void**);**

void Left\_fow**(**void**);**

void Left\_back**(**void**);**

void Right\_fow**(**void**);**

void Right\_back**(**void**);**

void Pal\_clear**(**void**);**

#endif

1. File robot\_shell.h

#ifndef ROBOT\_SHELL

#define ROBOT\_SHELL

#define SHELL\_WA\_SIZE THD\_WA\_SIZE(2048)

#define TEST\_WA\_SIZE THD\_WA\_SIZE(256)

#define STOP 0

#define FOWARD 1

#define BACKWARD 2

#define RIGHT 3

#define LEFT 4

void Serial\_Setup**(**void**);**

void Shell\_Setup**(**void**);**

#endif

1. File srcconf.h

#ifndef SRC\_CONF

#define SRC\_CONF

#include <stdio.h>

#include <string.h>

#include <stdint.h>

#include <stdarg.h>

#include <stdlib.h>

#include <math.h>

#include "ch.h"

#include "hal.h"

#include "memstreams.h"

#include "shell.h"

#include "chprintf.h"

#include "evtimer.h"

#include "robot\_led.h"

#include "robot\_shell.h"

#include "robot\_pal.h"

#include "robot\_gpt.h"

#define assert\_param(expr) ((void)0)

#endif

1. File main.c

#include "srcconf.h"

/\*

\* Application entry point.

\*/

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

/\*

\* System initializations.

\* - HAL initialization, this also initializes the configured device drivers

\* and performs the board-specific initializations.

\* - Kernel initialization, the main() function becomes a thread and the

\* RTOS is active.

\*/

halInit**();**

chSysInit**();**

/\*

\* Project defined HAL Setup

\*/

Gpt\_Setup**();**

Pal\_Setup**();**

Led\_Setup**();**

Serial\_Setup**();**

**while** **(**TRUE**)** **{**

Shell\_Setup**();** /\* create shell \*/

chThdSleepMilliseconds**(**500**);**

**};**

**}**

1. File robot\_gpt.c

#include "srcconf.h"

uint16\_t loop**;**

uint16\_t value**=**10**;**

extern uint16\_t dir**;**

static void gpt3cb**(**GPTDriver **\***gptp**)** **{**

**(**void**)**gptp**;**

loop**++;**

**if(**loop**==**value**){**

Pal\_clear**();**

**}**

**if(**loop**==**100**){**

loop**=**0**;**

**if(**dir**==**FOWARD**){**

Right\_fow**();**

Left\_fow**();**

palClearPad**(**PORT\_LED**,**RIGHT\_I**);**

palClearPad**(**PORT\_LED**,**LEFT\_I**);**

**}**

**else** **if(**dir**==**BACKWARD**){**

Right\_back**();**

Left\_back**();**

palClearPad**(**PORT\_LED**,**RIGHT\_I**);**

palClearPad**(**PORT\_LED**,**LEFT\_I**);**

**}**

**else** **if(**dir**==**LEFT**){**

Right\_back**();**

Left\_fow**();**

palClearPad**(**PORT\_LED**,**RIGHT\_I**);**

palSetPad**(**PORT\_LED**,**LEFT\_I**);**

**}**

**else** **if(**dir**==**RIGHT**){**

Right\_fow**();**

Left\_back**();**

palSetPad**(**PORT\_LED**,**RIGHT\_I**);**

palClearPad**(**PORT\_LED**,**LEFT\_I**);**

**}**

**else** **if(**dir**==**STOP**){**

Pal\_clear**();**

palSetPad**(**PORT\_LED**,**RIGHT\_I**);**

palSetPad**(**PORT\_LED**,**LEFT\_I**);**

**}**

**}**

chSysLockFromIsr**();**

gptStartOneShotI**(&**GPTD3**,** 10**);**

chSysUnlockFromIsr**();**

**}**

static const GPTConfig gpt3cfg **=** **{**

F\_GPT**,** /\* 10kHz timer clock.\*/

gpt3cb**,** /\* Timer callback.\*/

0

**};**

void Gpt\_Setup**(**void**){**

gptStart**(&**GPTD3**,** **&**gpt3cfg**);**

gptStartOneShot**(&**GPTD3**,** 10**);**

**}**

1. File robot\_led.c

#include "srcconf.h"

static WORKING\_AREA**(**wa\_ledThread**,** 128**);**

static msg\_t ledThread**(**void **\***arg**)** **{**

**(**void**)**arg**;**

**while(**TRUE**){**

chThdSleepMilliseconds**(**500**);**

palTogglePad**(**PORT\_LED**,**L0**);**

**}**

**return** 0**;**

**}**

static WORKING\_AREA**(**wa\_ledTestThread**,** 128**);**

static msg\_t ledTestThread**(**void **\***arg**)** **{**

**(**void**)**arg**;**

**while(**TRUE**){**

chThdSleepMilliseconds**(**1000**);**

palSetPad**(**PORT\_LED**,**L1**);**

palSetPad**(**PORT\_LED**,**L2**);**

palSetPad**(**PORT\_LED**,**L3**);**

palSetPad**(**PORT\_LED**,**L4**);**

palSetPad**(**PORT\_LED**,**L5**);**

palSetPad**(**PORT\_LED**,**L6**);**

palSetPad**(**PORT\_LED**,**L7**);**

palSetPad**(**PORT\_LED**,**L8**);**

chThdSleepMilliseconds**(**1000**);**

palClearPad**(**PORT\_LED**,**L1**);**

palClearPad**(**PORT\_LED**,**L2**);**

palClearPad**(**PORT\_LED**,**L3**);**

palClearPad**(**PORT\_LED**,**L4**);**

palClearPad**(**PORT\_LED**,**L5**);**

palClearPad**(**PORT\_LED**,**L6**);**

palClearPad**(**PORT\_LED**,**L7**);**

palClearPad**(**PORT\_LED**,**L8**);**

**}**

**return** 0**;**

**}**

void Led\_Setup**(**void**){**

palSetPadMode**(**PORT\_LED**,**L0**,**PAL\_MODE\_OUTPUT\_PUSHPULL**);**

palSetPadMode**(**PORT\_LED**,**L1**,**PAL\_MODE\_OUTPUT\_PUSHPULL**);**

palSetPadMode**(**PORT\_LED**,**L2**,**PAL\_MODE\_OUTPUT\_PUSHPULL**);**

palSetPadMode**(**PORT\_LED**,**L3**,**PAL\_MODE\_OUTPUT\_PUSHPULL**);**

palSetPadMode**(**PORT\_LED**,**L4**,**PAL\_MODE\_OUTPUT\_PUSHPULL**);**

palSetPadMode**(**PORT\_LED**,**L5**,**PAL\_MODE\_OUTPUT\_PUSHPULL**);**

palSetPadMode**(**PORT\_LED**,**L6**,**PAL\_MODE\_OUTPUT\_PUSHPULL**);**

palSetPadMode**(**PORT\_LED**,**L7**,**PAL\_MODE\_OUTPUT\_PUSHPULL**);**

palSetPadMode**(**PORT\_LED**,**L8**,**PAL\_MODE\_OUTPUT\_PUSHPULL**);**

palSetPad**(**PORT\_LED**,**L0**);**

palSetPad**(**PORT\_LED**,**L1**);**

palSetPad**(**PORT\_LED**,**L2**);**

palSetPad**(**PORT\_LED**,**L3**);**

palSetPad**(**PORT\_LED**,**L4**);**

palSetPad**(**PORT\_LED**,**L5**);**

palSetPad**(**PORT\_LED**,**L6**);**

palSetPad**(**PORT\_LED**,**L7**);**

palSetPad**(**PORT\_LED**,**L8**);**

chThdCreateStatic**(**wa\_ledThread**,** **sizeof(**wa\_ledThread**),** NORMALPRIO**,** ledThread**,** **NULL);**

//chThdCreateStatic(wa\_ledTestThread, sizeof(wa\_ledTestThread), NORMALPRIO, ledTestThread, NULL);

**}**

1. File robot\_pal.c

#include "srcconf.h"

void Right\_fow**(**void**){**

palSetPad**(**PORT\_PAL**,**RIGHT\_B**);**

palClearPad**(**PORT\_PAL**,**RIGHT\_A**);**

palClearPad**(**PORT\_LED**,**RIGHT\_I**);**

**}**

void Right\_back**(**void**){**

palSetPad**(**PORT\_PAL**,**RIGHT\_A**);**

palClearPad**(**PORT\_PAL**,**RIGHT\_B**);**

palSetPad**(**PORT\_LED**,**RIGHT\_I**);**

**}**

void Left\_fow**(**void**){**

palSetPad**(**PORT\_PAL**,**LEFT\_B**);**

palClearPad**(**PORT\_PAL**,**LEFT\_A**);**

palClearPad**(**PORT\_LED**,**LEFT\_I**);**

**}**

void Left\_back**(**void**){**

palSetPad**(**PORT\_PAL**,**LEFT\_A**);**

palClearPad**(**PORT\_PAL**,**LEFT\_B**);**

palSetPad**(**PORT\_LED**,**LEFT\_I**);**

**}**

void Pal\_clear**(**void**){**

palClearPad**(**PORT\_PAL**,**LEFT\_A**);**

palClearPad**(**PORT\_PAL**,**LEFT\_B**);**

palClearPad**(**PORT\_PAL**,**RIGHT\_A**);**

palClearPad**(**PORT\_PAL**,**RIGHT\_B**);**

**}**

static WORKING\_AREA**(**wa\_palTestThread**,** 128**);**

static msg\_t palTestThread**(**void **\***arg**)** **{**

**(**void**)**arg**;**

**while(**TRUE**){**

Left\_back**();**

Right\_fow**();**

chThdSleepMilliseconds**(**1000**);**

Left\_fow**();**

Right\_back**();**

chThdSleepMilliseconds**(**1000**);**

**}**

**return** 0**;**

**}**

void Pal\_Setup**(**void**){**

palSetPadMode**(**PORT\_PAL**,**O1**,**PAL\_MODE\_OUTPUT\_PUSHPULL**);**

palSetPadMode**(**PORT\_PAL**,**O2**,**PAL\_MODE\_OUTPUT\_PUSHPULL**);**

palSetPadMode**(**PORT\_PAL**,**O3**,**PAL\_MODE\_OUTPUT\_PUSHPULL**);**

palSetPadMode**(**PORT\_PAL**,**O4**,**PAL\_MODE\_OUTPUT\_PUSHPULL**);**

palSetPadMode**(**PORT\_PAL**,**O5**,**PAL\_MODE\_OUTPUT\_PUSHPULL**);**

palSetPadMode**(**PORT\_PAL**,**O6**,**PAL\_MODE\_OUTPUT\_PUSHPULL**);**

palSetPadMode**(**PORT\_PAL**,**O7**,**PAL\_MODE\_OUTPUT\_PUSHPULL**);**

palSetPadMode**(**PORT\_PAL**,**O8**,**PAL\_MODE\_OUTPUT\_PUSHPULL**);**

//chThdCreateStatic(wa\_palTestThread, sizeof(wa\_palTestThread), NORMALPRIO, palTestThread, NULL);

**}**

1. File robot\_shell.c

#include "srcconf.h"

uint16\_t dir**;**

Thread **\***shelltp **=** **NULL;**

static void cmd\_test**(**BaseSequentialStream **\***chp**,** int argc**,** char **\***argv**[])** **{**

**(**void**)**argv**;**

**if(**argc**>**0**){**

chprintf**(**chp**,**"bad commands\n\r"**);**

**return;**

**}**

chprintf**(**chp**,**"serial ok !!\n\r"**);**

**return;**

**}**

static void cmd\_foward**(**BaseSequentialStream **\***chp**,** int argc**,** char **\***argv**[])** **{**

**(**void**)**argv**;**

**if(**argc**>**0**){**

chprintf**(**chp**,**"bad commands\n\r"**);**

**return;**

**}**

dir**=**FOWARD**;**

chThdSleepMilliseconds**(**250**);**

dir**=**STOP**;**

**return;**

**}**

static void cmd\_backward**(**BaseSequentialStream **\***chp**,** int argc**,** char **\***argv**[])** **{**

**(**void**)**argv**;**

**if(**argc**>**0**){**

chprintf**(**chp**,**"bad commands\n\r"**);**

**return;**

**}**

dir**=**BACKWARD**;**

chThdSleepMilliseconds**(**250**);**

dir**=**STOP**;**

**return;**

**}**

static void cmd\_right**(**BaseSequentialStream **\***chp**,** int argc**,** char **\***argv**[])** **{**

**(**void**)**argv**;**

**if(**argc**>**0**){**

chprintf**(**chp**,**"bad commands\n\r"**);**

**return;**

**}**

dir**=**RIGHT**;**

chThdSleepMilliseconds**(**50**);**

dir**=**STOP**;**

**return;**

**}**

static void cmd\_left**(**BaseSequentialStream **\***chp**,** int argc**,** char **\***argv**[])** **{**

**(**void**)**argv**;**

**if(**argc**>**0**){**

chprintf**(**chp**,**"bad commands\n\r"**);**

**return;**

**}**

dir**=**LEFT**;**

chThdSleepMilliseconds**(**50**);**

dir**=**STOP**;**

**return;**

**}**

static void cmd\_stop**(**BaseSequentialStream **\***chp**,** int argc**,** char **\***argv**[])** **{**

**(**void**)**argv**;**

**if(**argc**>**0**){**

chprintf**(**chp**,**"bad commands\n\r"**);**

**return;**

**}**

dir**=**STOP**;**

**return;**

**}**

static void cmd\_booted**(**BaseSequentialStream **\***chp**,** int argc**,** char **\***argv**[])** **{**

**(**void**)**argv**;**

**if(**argc**>**0**){**

chprintf**(**chp**,**"bad commands\n\r"**);**

**return;**

**}**

palTogglePad**(**PORT\_LED**,**L3**);**

**return;**

**}**

static const ShellCommand commands**[]** **=** **{**

**{**"test"**,**cmd\_test**},**

**{**"foward"**,**cmd\_foward**},**

**{**"backward"**,**cmd\_backward**},**

**{**"right"**,**cmd\_right**},**

**{**"left"**,**cmd\_left**},**

**{**"stop"**,**cmd\_stop**},**

**{**"booted"**,**cmd\_booted**},**

**{NULL,** **NULL}**

**};**

static const ShellConfig shell\_cfg1 **=** **{**

**(**BaseSequentialStream **\*)&**SD1**,**

commands

**};**

void Serial\_Setup**(**void**){**

palSetPadMode**(**GPIOA**,**9**,**16**);**

palSetPadMode**(**GPIOA**,**10**,**2**);**

sdStart**(&**SD1**,NULL);**

shellInit**();**

**}**

void Shell\_Setup**(**void**){**

**if** **(!**shelltp**){**

shelltp **=** shellCreate**(&**shell\_cfg1**,** SHELL\_WA\_SIZE**,** NORMALPRIO**);}** /\* create shell tread \*/

**else** **if** **(**chThdTerminated**(**shelltp**))** **{**

chThdRelease**(**shelltp**);** /\* Recovers memory of the previous shell. \*/

shelltp **=** **NULL;** /\* Triggers spawning of a new shell. \*/

**}**

**}**

1. File Makefile

ifeq ($(USE\_OPT),)

USE\_OPT **=** -O2 -ggdb -fomit-frame-pointer -falign-functions=16

endif

# C specific options here (added to USE\_OPT).

ifeq ($(USE\_COPT),)

USE\_COPT **=**

endif

# C++ specific options here (added to USE\_OPT).

ifeq ($(USE\_CPPOPT),)

USE\_CPPOPT **=** -fno-rtti

endif

# Enable this if you want the linker to remove unused code and data

ifeq ($(USE\_LINK\_GC),)

USE\_LINK\_GC **=** yes

endif

# If enabled, this option allows to compile the application in THUMB mode.

ifeq ($(USE\_THUMB),)

USE\_THUMB **=** yes

endif

# Enable this if you want to see the full log while compiling.

ifeq ($(USE\_VERBOSE\_COMPILE),)

USE\_VERBOSE\_COMPILE **=** no

endif

#

# Build global options

##############################################################################

##############################################################################

# Architecture or project specific options

#

# Enable this if you really want to use the STM FWLib.

ifeq ($(USE\_FWLIB),)

USE\_FWLIB **=** no

endif

#

# Architecture or project specific options

##############################################################################

##############################################################################

# Project, sources and paths

#

# Define project name here

PROJECT **=** robot

# Imported source files and paths

CHIBIOS **=** ..

include $(CHIBIOS)/boards/STM32/board.mk

include $(CHIBIOS)/os/hal/platforms/STM32F1xx/platform.mk

include $(CHIBIOS)/os/hal/hal.mk

include $(CHIBIOS)/os/ports/GCC/ARMCMx/STM32F1xx/port.mk

include $(CHIBIOS)/os/kernel/kernel.mk

# Define linker script file here

LDSCRIPT**=** $(PORTLD)/STM32F103xB.ld

# C sources that can be compiled in ARM or THUMB mode depending on the global

# setting.

CSRC **=** $(PORTSRC) \

$(KERNSRC) \

$(TESTSRC) \

$(HALSRC) \

$(PLATFORMSRC) \

$(BOARDSRC) \

$(CHIBIOS)/os/various/syscalls.c \

$(CHIBIOS)/os/various/evtimer.c \

$(CHIBIOS)/os/various/shell.c \

$(CHIBIOS)/os/various/chprintf.c \

$(CHIBIOS)/os/various/memstreams.c \

main.c robot\_led.c robot\_pal.c robot\_gpt.c robot\_shell.c

# C++ sources that can be compiled in ARM or THUMB mode depending on the global

# setting.

CPPSRC **=**

# C sources to be compiled in ARM mode regardless of the global setting.

# NOTE: Mixing ARM and THUMB mode enables the -mthumb-interwork compiler

# option that results in lower performance and larger code size.

ACSRC **=**

# C++ sources to be compiled in ARM mode regardless of the global setting.

# NOTE: Mixing ARM and THUMB mode enables the -mthumb-interwork compiler

# option that results in lower performance and larger code size.

ACPPSRC **=**

# C sources to be compiled in THUMB mode regardless of the global setting.

# NOTE: Mixing ARM and THUMB mode enables the -mthumb-interwork compiler

# option that results in lower performance and larger code size.

TCSRC **=**

# C sources to be compiled in THUMB mode regardless of the global setting.

# NOTE: Mixing ARM and THUMB mode enables the -mthumb-interwork compiler

# option that results in lower performance and larger code size.

TCPPSRC **=**

# List ASM source files here

ASMSRC **=** $(PORTASM)

INCDIR **=** $(PORTINC) $(KERNINC) $(TESTINC) \

$(HALINC) $(PLATFORMINC) $(BOARDINC) \

$(CHIBIOS)/os/various

#

# Project, sources and paths

##############################################################################

##############################################################################

# Compiler settings

#

MCU **=** cortex-m3

#TRGT = arm-elf-

TRGT **=** arm-none-eabi-

CC **=** $(TRGT)gcc

CPPC **=** $(TRGT)g++

# Enable loading with g++ only if you need C++ runtime support.

# NOTE: You can use C++ even without C++ support if you are careful. C++

# runtime support makes code size explode.

LD **=** $(TRGT)gcc

#LD = $(TRGT)g++

CP **=** $(TRGT)objcopy

AS **=** $(TRGT)gcc -x assembler-with-cpp

OD **=** $(TRGT)objdump

HEX **=** $(CP) -O ihex

BIN **=** $(CP) -O binary

# ARM-specific options here

AOPT **=**

# THUMB-specific options here

TOPT **=** -mthumb -DTHUMB

# Define C warning options here

CWARN **=** -Wall -Wextra -Wstrict-prototypes

# Define C++ warning options here

CPPWARN **=** -Wall -Wextra

#

# Compiler settings

##############################################################################

##############################################################################

# Start of default section

#

# List all default C defines here, like -D\_DEBUG=1

DDEFS **=**

# List all default ASM defines here, like -D\_DEBUG=1

DADEFS **=**

# List all default directories to look for include files here

DINCDIR **=**

# List the default directory to look for the libraries here

DLIBDIR **=**

# List all default libraries here

DLIBS **=**

#

# End of default section

##############################################################################

##############################################################################

# Start of user section

#

# List all user C define here, like -D\_DEBUG=1

UDEFS **=**

# Define ASM defines here

UADEFS **=**

# List all user directories here

UINCDIR **=**

# List the user directory to look for the libraries here

ULIBDIR **=**

# List all user libraries here

ULIBS **=**

#

# End of user defines

##############################################################################

ifeq ($(USE\_FWLIB),yes)

include $(CHIBIOS)/ext/stm32lib/stm32lib.mk

CSRC +**=** $(STM32SRC)

INCDIR +**=** $(STM32INC)

USE\_OPT +**=** -DUSE\_STDPERIPH\_DRIVER

endif

include $(CHIBIOS)/os/ports/GCC/ARMCMx/rules.mk