Nauteff Conception du matériel et du logiciel

Emmanuel Gautier

12 mars 2021

Résumé

Nauteff P-1 est une maquette de pilote automatique pour navires. Il est destiné à développer et tester son matériel et ses algorithmes. Ce document contient les informations relatives à la conception de cette maquette.

Historique du document

Version	Date	Description
	15 avril 2020	Début de rédaction

Table des matières

Table des figures

Chapitre 1

Buts du document

1.1 But

Ce document contient les informations de la conception du Nauteff P-1 et vise à en permettre un développement et une maintenance efficace. Il est principalement destiné au développeurs.

1.2 Guide de lecture

La documentation nécessaire à la maîtrise du STM32 est volumineuse et répartie dans plusieurs documents mentionnés dans le chapitre suivant.

Chapitre 2

Documents applicables et de référence

2.1 Documents applicables

[1] Nauteff, éd. *Algorithmes de Nauteff*. 2020.

[2] Nauteff, éd. *Dossier de spécification de Nauteff*. 2020.

[3] Eric S. Raymond. *NMEA revealed*. Anglais. Version 2.23. 2019. url : `https://gpsd.gitlab.io/gpsd/NMEA.html`.

[4] ST Microelectronic, éd. *iNEMO inertial module 3D accelerometer and 3D gyroscope*. 2015.

Dossier de spécification de Nauteff NAUTEFF-SPEC [4] [2] [1] [3]
Algorithmes de calcul de Nauteff NAUTEFF-CALC

2.2 Documents de référence

NMEA revealed https ://gpsd.gitlab.io/gpsd/NMEA.html
RM0316 Reference manual STM32F303 ST Microelectronics
STM32F303x6/x8 ST Microelectronics
ARM Cortex-M4 Processor, Technical Reference Manual
Revision : r0p1
100166_0001_00_en
ARM
ARM Cortex-M4 Devices, Generic User Guide ARM
LIS3MDL ST Microelectronics Digital output magnetic sensor : ultra-low power, high performance, 3-axis magnetometer. DocID024204 Rev 4 mai 2015
LSM6DS33 ST Microelectronics iNEMO inertial module : always-on 3D accelerometer and 3D gyroscope DicID024423 Rev 4 october 2015

Chapitre 3

Terminologie

MEMS Micro Electro Mechanicals Systems. 9

Chapitre 4

Matériel

4.1 STM32

4.1.1 Horlogerie

Nauteff utilise l'horloge interne HSI à 8Mhz.

4.1.2 Interruptions

Liste

Interruption	n°	Usage
Exceptions du système		
Reset		appel de la fct main()
NMI		
HardFault		
MemManage		
BusFault		
UsageFault		
SVCall		vPortSVCHandler (fct de FreeRTOS)
PendSV		xPortPendSVHandler (fct de FreeRTOS)
SysTick		xPortSysTickHandler (fct de FreeRTOS)
Événements extérieurs		
EXTI0		Bouton Marche
EXTI1		Bouton Veille
EXTI2		Touche +1
EXTI3		Touche -1
EXTI4		Touche +10
EXTI10_15		Touche -10
I2C1_Event		Évènement I2C1
I2C1_Error		Erreur I2C1
USART1_Event		Événement USART 1

USART

I2C1

Le bus I2C1 sert à la communication avec les capteurs MEMS.

4.1.3 Affectation des broches du STM32

Le STM32F303K8 comporte 32 broches.

STM	Nucleo	Fonction	Périphérique	Usage
PA0	A0	Entrée GPIO, PU	EXTI0	Bouton marche
PA1	A1	Entrée GPIO, PU	EXTI1	Bouton veille
PA2	A7	Entrée GPIO, PU	EXTI2	Touche +1
PA3	A2	Entrée GPIO, PU	EXTI3	Touche -1
PA4	A3	Entrée GPIO, PU	EXTI4	Touche +10
PA5	A4	Entrée GPIO	EXTI5	Inutilisable sur nucléo
PA6	A5	Entrée GPIO	EXTI6	Inutilisable sur nucléo
PA7	A6	Entrée analogique	ADC2_IN4	Courant moteur
PA8	D9	Alt Fct n° 0	RCC_MCO	Sortie horloge système
PA9	D1	Alt Fct n° 7	USART1 TX	Sortie série
PA10	D0	Alt Fct n° 7	USART1 RX	Entrée série
PA11	D10	Entrée GPIO PU	EXTI10_15	Touche -10
PB0	D3	Sortie GPIO	GPIO B0	Commande moteur
PB1	D6	Entrée analogique	ADC1_IN2	Tension alim.
PB3	D13	Sortie GPIO	GPIO B3	LED Verte et embrayage
PB4	D12	Sortie GPIO	GPIO B4	Sens pont A (INA)
PB5	D11	Sortie GPIO	GPIO B5	Sens pont B (INB)
PB6	D5	Alt Fct n° 4	I2C1 SCL	Vers capteurs MEMs
PB7	D4	Alt Fct n° 4	I2C1 SDA	Vers capteurs MEMs

La broche PA8 sert à observer l'horloge système à l'oscilloscope. Elle pourra servir plus tard à une autre fonction.

4.1.4 Organisation de la mémoireSTM32

Mémoire	Adresse	taille		
FLASH	0x0800 0000	64K		Permanente, plus lente
RAM	0x2000 0000	12K		
CCM RAM	0x1000 0000	4K		Core coupled memory

4.2 Capteurs MEMS

Les capteurs MEMS sont un montés sur une carte d'essai de Polulu AltIMU-10 v5. Cette carte contient les circtuits de ST Microelectronics suivants :

— un compas LIS3MDL ;
— un gyromètre et accéléromètre LSMDS33 ;
— un baromètre LPS25H, non utilisé.

4.3 Commande de l'actionneur

La commande de l'actionneur est réalisée par un circuit VNH 5019 de ST Microelectronics monté sur une plaque d'essai de Pololu. Ce circuit comprend un pont en H pouvant délivrer un courant de 12A en permanence avec des protections contre les surcharges et court-circuits, des signaux de diagnostic et une tension proportionnelle au courant. Ce circuit a un partie logique et contrôle alimentée avec la carte Nucleo et une partie de puissance alimentée par une source de 12V.

Les signaux suivants sont utilisés :
— INA ,sens pont A) relié à B4 ;
— INB (sens pont B) relié à B5 ;
— CS relié PA7 (ADC2_IN4) ;
— PWM relié à PB0.

L'actuel STM32 F303K8 n'a pas suffisamment de broches pour y raccorder ENA/DIAG et ENB/DIAG.

Chapitre 5

Architecture générale de l'application

5.1 Architecture statique

5.1.1 Système

5.1.2 Pilotes de périphériques

5.1.3 Tâches

5.1.4 Calculs

5.1.5 Traitements des trames NMEA0183

5.2 Relations dynamiques

5.3 Justification des choix d'architecture

Chapitre 6

Contexte de la conception

6.1 présentation de l'application

Nauteff est un pilote automatique pour navires.

6.2 Principales exigences applicables

6.2.1 Exigences liées à la sûreté de fonctionnement

Le navigateur, confiant la timonerie de son navire à Nauteff, attend de ce dernier un fonctionnement fiable et sûr. Il doit être conçu pour éviter toute panne ou comportement mettant en danger le navire et son équipage. Ainsi la programmation suit le principe KISS ("Keep It Simple Stupid") et est défensive. Les principes suivants sont adoptés :

— commentaires précis et détaillés dans le code ;
— Absence d'allocation dynamique après le démarrage ;
— Usage très réduit de variables locales dans les fonctions.

6.2.2 Architecture matérielle opérationnelle

Nauteff utilise une carte Nucleo dotée d'un STM32 F303K8 et d'une sonde ST-Link accessible par prise USB.

Le STM32 F303K8 ne comporte que 32 broches, le prochain Nauteff devrait utiliser un STM32 avec au moins 48 broches.

6.2.3 Logiciels imposés

Néant.

6.2.4 Interface avec d'autres applications

Interfaces NMEA 0183 Cette interface est aussi utilisée pour la sortie de messages pendant le développement.

6.3 Contraintes de développement

6.3.1 Méthode et formalisme

La programmation est essentiellement fonctionnelle. Le formalisme est celui d'UML pour les schémas. Les concepts de la programmation objets ne sont pas utilisés.

6.3.2 Langage de programmation

La programmation est réalisée entièrement avec le langage C11. Il permet d'accéder très facilement à la mémoire et à tous les registres. Le recours à l'assembleur n'est pas envisagé.

6.3.3 Logiciels extérieurs

FreeRTOS V 10.0.1 fournit le noyau temps réel, les horloges, les sémaphores et les canaux de communication. Nauteff P-1 utilise la librairie standard libc et la librairie mathématiques lm.

6.3.4 Outils

EDI	Eclipse C/C++	eclipse.org
Compilation	arm-none-eabi-gcc	chaîne du GNU, compilation croisée
sonde	ST-Link	intégrée sur carte Carte Nucleo
Accès sonde	openocd 0.10.0	
Documentation	Latex	avec TeXstudio

6.3.5 Environnement matériel de développement

Le développement au bureau est réalisé avec les matériels suivant :
— un PC permettant de faire fonctionner eclipse ;
— un oscilloscope pour certaines mise au point ;
— occasionnellement un voltmètre ;
— petit outillage : pinces tournevis, dénudeuse,... ;
Lors des essais en mer du prototype un petit PC consommant peu mais ne permettant pas d'utiliser eclipse permet de faire des modifications du logiciel et, en particulier, des ajustements de paramètres.Il embarque la chaîne de compilation et openocd. La compilation est alors réalisée avec la commande make.

Chapitre 7

Architecture du logiciel

7.1 Noyau temps réel : FreeRTOS

7.2 Tâches

Tâche	Priorité	Nom
Gestion du clavier	6	`TaskKeyboard`
Contrôle du moteur	5	`TaskMotor`
Gestion de l'état	3	`TaskCore`
Gestion des capteurs MEMs	3	`TaskMEMs`

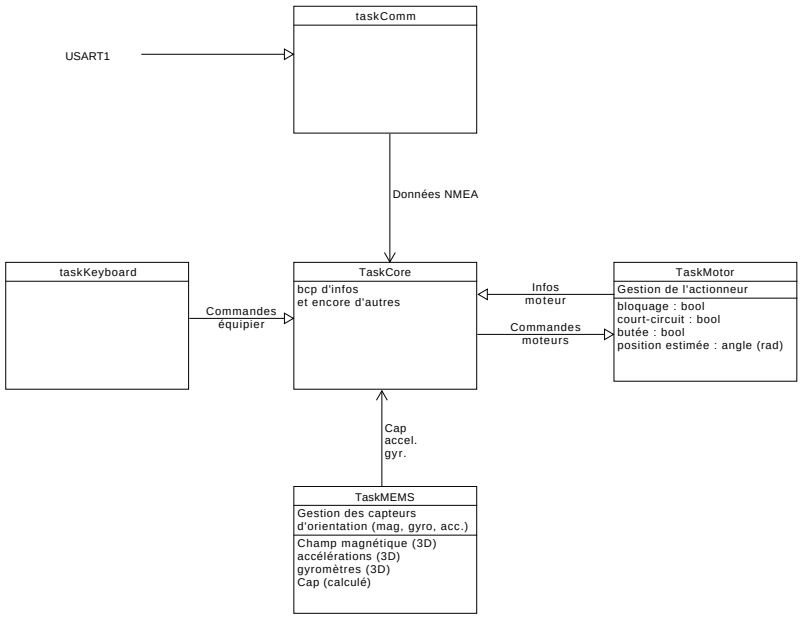

FIGURE 7.1 – Flots de données entre les tâches

7.2.1 ScrutationClavier

La tâche utilise la fonction nommée TaskKeyboard qui n'utilise aucun argument. Elle fonctionne par scrutation périodique des entrées sur lesquelles sont branchées les touches. La scrutation périodique est utilisée car les touches provoquaient de très nombreux rebonds parasites. Elle envoie les ordres à la tâche de gestion de l'état par l'intermédiaire d'un `MessageBuffer_t` nommé buffclav.

7.2.2 Contrôle du moteur

Cette tâche reçoit les ordres de la tâche de gestion, elle commande l'embrayage du moteur et son fonctionnement : sens et durée. Elle corrige la durée de fonctionnement en fonction de l'effort du moteur. Lorsque le moteur tourne, elle mesure le courant du moteur, vérifie que le moteur n'est pas bloqué ou débranché et renvoie une estimation de l'effort du moteur. Elle informe la tâche principale lorsque le moteur a fini un mouvement demandé.

Entrées

— Ordres de commande du moteur et de l'embrayage en provenance de la tâche principale ;
— valeurs du courant et de la tension en provenance de l'ADC ;
— horloge (à définir).

Sorties

— ordres vers le moteur mise en marche et sens ;
— ordres vers l'embrayage : enclenchement et désenclenchement ;
— informations vers la tâche principale sur le fonctionnement du moteur estimation de l'effort, de la position, blocage ou court-circuit.

7.2.3 Gestion des capteurs MEMS

Cette tâche lit les informations des capteurs MEMs, les filtre, calcule le cap et le lacet et envoie les informations traitées à la tâche principale. Les documents de ré

Le signal d'interruption n'étant pas disponible sur la carte de Pololu la tâche fonctionne par scrutation.

Sorties

La tâche envoie les données vers la tâche principale.

7.2.4 Tâche principale

Entrées

— Commandes en provenance du clavier ;
— Informations du moteur ;
— Informations des capteurs MEMS ;
— Informations en provenance des interfaces NMEA
— horloge (à définir).

Sorties

— Commandes moteurs ;
— alarmes ;

7.3 Canaux de communications

Canal	Nom	Type	Taille
Messages vers le gestionnaire principal	`bufferCore`	`MessageBuffer_t`	10
Ordres vers le moteur	`bufferMotor`	`MessageBuffer_t`	5

7.3.1 Message gestionnaire principal

Ces messages sont envoyés par les tâches TaskKeyboard, TaskMotor et TaskMEMs. Il sont stockés dans une structure dont les deux premiers octets comportent un code indiquant la nature du message et, en option, d'autres informations.

Ordre	Code et valeurs optionnelles
Ordres en provenance de la tâche du clavier vers la tâche principale	
Mise en mode automatique	MSG_KBD_AUTO
Mise en veille	MSG_KBD_STDBY
+1	MSG_KBD_STARBOARD_ONE
-1	MSG_KBD_PORT_ONE
+10	MSG_KBD_STARBOARD_TEN
-10	MSG_KBD_PORT_TEN
+1 relachée	MSG_KBD_STARBOARD_ONE_END
-1 relachée	MSG_KBD_PORT_ONE_END
Informations en provenance de la tâche de gestion du moteur	
Position estimée du moteur	MSG_MOT_POSITION_ESTIMEE
Effort moteur	MSG_MOT_EFFORT + float effort
Moteur bloqué ou en butée	MSG_MOT_BLOCKED
Court-circuit sortie moteur	MSG_MOT_SHORT_CIRCUIT
Fin de déplacement actuateur	MSG_MOT_END_MOVE + float : effort
Informations des capteurs MEMs	
Info Capteurs MEMs	MSG_MEMS_VALUES + 2 floats : cap et lacet
Info MEMs indisponible	MSG_MEMS_FAILURE

7.3.2 Commandes envoyées à la tâche de gestion du moteur

Ces messages sont envoyés par la tâche principale ou un timer à la tâche de gestion du moteur.

Ordre	Code et valeurs optionnelles
Commande de la tâche principale	
Enclenchement de l'embrayage	MSG_MOT_EMBRAYE
Désenclenchement de l'embrayage	MSG_MOT_DEBRAYE
Mise en marche vers tribord	MSG_MOT_STARBOARD
Mise en marche vers babord	MSG_MOT_PORT
Mouvement vers tribord	MSG_MOT_MOVE_STARBOARD + float : valeur
Mouvement vers babord	MSG_MOT_MOVE_PORT + float : valeur
Arrêt du moteur	MSG_MOT_STOP
Demande de mesure d'effort	MSG_MOT_DMD_EFFORT
commande du timer	
Commande du TIMER	MSG_MOT_MOVING_CONTROL

La tâche de gestion du moteur effectue les déplacements dont l'amplitude doit être proportionnelle à la valeur en paramètre, elle adapte la durée au courant du moteur.