RobotG7_project

Généré par Doxygen 1.9.4

Chapitre 1

ESP32_Project_Robot_Manage

1.1 Présentation

Ce projet rentre dans le cadre du cours de robotique. Le cours de robotique est un cours réservé aux étudiants de licence 4 Réseaux et Télécommunications à la faculté des sciences informatiques à UDBL. Il a pour but de mettre en évidence les connaissances acquises dans le cours de robotique, d'électronique ainsi que dans le cours des systèmes embarqués (Cours de Licence 3, prérequis à ce cours). Le projet en question consiste en la mise en place d'un robot ménager qui accomplit la tâche d'un aspirateur. L'idée de ce projet est venue de l'observation quotidienne de notre milieu.

En effet, pour réaliser la tâche ou l'objectif cité de notre robot nous avons mis en œuvre plusieurs notions de la connaissance parmi lesquelles : la maîtrise du langage C++, la souplesse dans conception à l'aide des outils de base ainsi que la logique. Non seulement nos connaissances ont été mises en œuvre, mais aussi nous avons eu en notre possession quelques matériels importants : l'ESP32, Pilote Moteur L298, Capteur à Ultrason, Ventilateur 12 volt... Ce projet se veut être une contribution devant résoudre un problème dans la société en ce qui concerne le ménage en mettant en relief un aspirateur. En fait, Un aspirateur, aussi appelé balayeuse est un appareil électro-ménager muni d'une pompe à air créant une dépression qui provoque l'aspiration de poussière et de petits déchets tombés au sol. Il sert donc au nettoyage des tapis, des parquets, sols et autres surfaces

Dans la réalisation de ce projet des difficultés n'ont pas manqué. Elles concernent particulièrement la disponibilité d'équipements fiables et adéquats. Nous n'allons pas étaler toutes nos difficultés dans cette section, mais nous le ferons un peu plus tard dans ce document. Ainsi, ce document a pour but de présenter notre projet en donnant : ses objectifs, son fonctionnement, sa conception ainsi que ses limites.

1.2 Objectifs

Comme tout projet, notre robot possède des intérêts ou des objectifs. Pour les énumérer, nous allons le classer dans l'ordre suivant :

- Intérêt Académique : Les cours vus sur le banc de l'école sont sanctionnés par différents travaux (Interrogations, travaux pratiques, travaux dirigés...). Ces travaux permettent aux enseignants d'évaluer les étudiants ainsi qu'aux étudiants de s'évaluer eux-mêmes. En réalisant ce projet, nous avons pu évaluer nos compétences et nos capacités tant humaines qu'intellectuelles.
- Intérêt scientifique : Sur le banc de l'école, un projet est intéressant s'il relève aussi un intérêt scientifique. Ainsi l'intérêt scientifique de notre projet se trouve dans le fait que nous avons eu à combiner plusieurs aspects de la connaissance (programmation, électronique, mécanique...)
- Intérêt social : Le but principal d'une formation académique est de faire des étudiants des personnes aptes à résoudre des problèmes tant simples que complexes dans leur environnement tout comme dans leur pays. Le but de l'automatisation est de suppléer l'homme à tout travaux qui exigent beaucoup d'énergie physique. Ainsi notre projet met en œuvre un robot qui facilite une tâche de ménage : le balayage.

1.3 Fonctionnement

Cette section a pour but de décrire le fonctionnement du robot ménager. Etant des ingénieurs en formation, il convient d'utiliser un diagramme pour représenter le principe de fonctionnement du robot en question. Pour arriver à nos fins, nous avons choisi d'utiliser le langage UML. En fait, le Langage de Modélisation Unifié, de l'anglais Unified Modeling Language (UML), est un langage de modélisation graphique à base de pictogrammes conçu comme une méthode normalisée de visualisation dans les domaines du développement logiciel et en conception orientée objet. En fait, notre robot possède trois modes de fonctionnement : le mode start, le mode manuel ainsi que le mode automatique

- Mode Manuel Pour expliquer le fonctionnement manuel, nous allons utiliser le diagramme de séquence qui représente la séquence de messages entre les objets au cours d'une interaction.

- Mode Automatisé Pour expliquer le fonctionnement automatique, nous allons utiliser le diagramme d'activité qui fournit une vue du comportement de notre système en décrivant sa séquence d'actions. La page suivante illustre le diagramme en question.

1.4 Schémas du robot

Dans la documentation d'un système, nous retrouvons des éléments indispensables parmi lesquels nous pouvons citer : la schématisation. Au fait, la schématisation consiste à expliquer, souvent par un dessin, un concept, un objet complexe quitte à le rendre plus simple que ce qu'il n'est en réalité.

- ESP32 ESP32 est une série de microcontrôleurs de type système sur une puce (SoC) d'Espressif Systems, basé sur l'architecture Xtensa LX6 de Tensilica (en), intégrant la gestion du Wi-Fi et du Bluetooth (jusqu'à LE 5.0 et 5.11) en mode double, et un DSP. C'est une évolution d'ESP8266. Le principal outil de développement est ESP-IDF, logiciel libre développé par Espressif, écrit en C et utilisant le système temps réel FreeRTOS. Il intègre un nombre important de bibliothèques et on retrouve dans son écosystème des bibliothèques tierce libres pour différents types de périphériques liés à l'embarqué et au temps réel.
- Pilote Moteur L298 Le L298N est un double pilote à pont complet. C'est un circuit monolithique intégré, un driver pont complet, double, fort courant et tension élevée, conçu pour accepter des niveaux logiques TTL standards et pour piloter des charges inductives telles que des relais, solénoïdes, moteurs DC et pas à pas. Deux entrées d'activation sont disponibles pour activer ou désactiver le composant indépendamment des signaux d'entrée. Les émetteurs des transistors inférieurs de chaque pont sont connectés ensemble et la borne externe correspondante peut être utilisée pour le raccordement d'une résistance de détection externe. Une entrée d'alimentation supplémentaire est prévue pour que la logique fonctionne avec une tension inférieure. Il est utilisé dans des applications telles que les moteurs DC à double balais et les moteurs pas à pas.
- Capteur à Ultrason Un capteur à ultrasons émet à intervalles réguliers de courtes impulsions sonores à haute fréquence. Ces impulsions se propagent dans l'air à la vitesse du son. Lorsqu'elles rencontrent un objet, elles se réfléchissent et reviennent sous forme d'écho au capteur. Celui-ci calcule alors la distance le séparant de la cible sur la base du temps écoulé entre l'émission du signal et la réception de l'écho.
- Ventilateur 12 volt Pour réaliser un aspirateur, nous avons eu principalement besoin d'un ventilateur 12 v. Un ventilateur 12 volts est un dispositif qui fonctionne avec une tension de 12 volts

1.4.1 Aspect mobilité

Dans la conception d'un robot, l'aspect mobilité est très important. Pour rendre mobile notre robot, nous avons utilisé les éléments suivants :

- Moteur a engrenage DC 3-6v pour voiture Arduino
- Les pneus correspondants à ces engrenages

1.5 Dimensionnement électrique

Bilan de puissance Ventilateur : I = 0,88 A ; U = 12 V ; Pneu(moteur) : U = 6 V ; P = 1,2 W ; Capteur Ultrason : I = 2 mA ; U = 5 V Shied motor: U = 5 à 35; I = 36 mA; P = 1,26 W Servomotor: ESP32: 5V Pt = 0,88 x 12 + 1,2 + 2 x 5 x 10 ^ -3 + 1,26 + 5 x 0,2 = 13,23 W Une batterie délivre 3,7V. Nous avons trois batteries, ce qui fera 3,7x3 = 11,1 v et I = 2600mA, Donc la puissance sera : P = 11,1 x 2600 x 10 ^-3 = 28,86 W

1.6 Dependencies

Pour ce qui concerne les dépendances,nous pouvons citer :

- DabbleESP32, téléchargeable sur le lien : https://github.com/STEMpedia/DabbleESP32
- ESP32Servo, téléchargeable sur le lien : https://github.com/madhephaestus/ESP32↩ ServoServer

1.7 LICENSE

The part contains my code is released under BSD 2-Clause License. Regarding other libraries used in this project, please follow the respective Licenses.

Chapitre 2

Index des classes

2.1 Liste des classes

Liste des classes, structures, unions et interfaces avec une brève description :

Chapitre 3

Index des fichiers

3.1 Liste des fichiers

Liste de tous les fichiers avec une brève description :

Chapitre 4

Documentation des classes

4.1 Référence de la classe CtrlWay

La class CtrlWay pour gerer la facon de réagir face à l'obstacle.

`#include <CtrlWay.h>`

Fonctions membres publiques

- CtrlWay (int pinAttached=15)

  CtrlWay(...) : constructeur de la classe.

- void begin ()

  Pour initialiser tout les composant attachés.

- WAY theWayToget (GuideSound ∗CaptureObstacle)

  POur choisir quel cote prendre après analyse des deux c\^otés.

4.1.1 Description détaillée

La class CtrlWay pour gerer la facon de réagir face à l'obstacle.

Définition à la ligne 32 du fichier CtrlWay.h.

4.1.2 Documentation des constructeurs et destructeur

4.1.2.1 CtrlWay()

```
CtrlWay::CtrlWay (
            int pinAttached = 15 )
```

CtrlWay(...) : constructeur de la classe.

Paramètres

pinAttached	la pin de control

Définition à la ligne 17 du fichier CtrlWay.cpp.

4.1.3 Documentation des fonctions membres

4.1.3.1 begin()

```
void CtrlWay::begin ( )
```

Pour initialiser tout les composant attachés.

Définition à la ligne 22 du fichier CtrlWay.cpp.

4.1.3.2 theWayToget()

```
WAY CtrlWay::theWayToget (
            GuideSound * CaptureObstacle )
```

POur choisir quel cote prendre après analyse des deux c\^otés.

Paramètres

CaptureObstacle	Un objet de la classe GuideSound implémentant le capteur Ultra son

Renvoie

WAY, le chemin choisit

Définition à la ligne 94 du fichier CtrlWay.cpp.

La documentation de cette classe a été générée à partir du fichier suivant :

- RobotG7/include/CtrlWay/CtrlWay.h
- RobotG7/include/CtrlWay/CtrlWay.cpp

4.2 Référence de la classe GuideSound

GuideSound est la classe qui gere le capteur Ultra Son.

```
#include <GuideSound.h>
```

Fonctions membres publiques

— GuideSound (uint8_t trig=0, uint8_t echo=0)

Guidesound est le constructeur.

— void begin ()

begin initialise la capteur ultra son

— uint32_t getDistance ()

recuperer la distance

— uint32_t getDelayEcho ()

Verifie le delay d'aller et retour du son emi.

4.2.1 Description détaillée

GuideSound est la classe qui gere le capteur Ultra Son.

Définition à la ligne 20 du fichier GuideSound.h.

4.2.2 Documentation des constructeurs et destructeur

4.2.2.1 GuideSound()

```
GuideSound::GuideSound (
            uint8_t trig = 0,
            uint8_t echo = 0 )
```

Guidesound est le constructeur.

Paramètres

trig	parametre de la pin trigger
echo	parametre de la pin echo

Définition à la ligne 20 du fichier GuideSound.cpp.

4.2.3 Documentation des fonctions membres

4.2.3.1 begin()

```
void GuideSound::begin ( )
```

begin initialise la capteur ultra son

Définition à la ligne 27 du fichier GuideSound.cpp.

4.2.3.2 getDelayEcho()

```
uint32_t GuideSound::getDelayEcho ( )  [inline]
```

Verifie le delay d'aller et retour du son emi.

Renvoie

uint32_t la duree du trajet

Définition à la ligne 37 du fichier GuideSound.h.

4.2.3.3 getDistance()

```
uint32_t GuideSound::getDistance ( )  [inline]
```

recuperer la distance

Renvoie

int la distance de l'obstacle

Définition à la ligne 33 du fichier GuideSound.h.

La documentation de cette classe a été générée à partir du fichier suivant :

- RobotG7/include/GuideSound/GuideSound.h
- RobotG7/include/GuideSound/GuideSound.cpp

4.3 Référence de la structure Motor

Structure MOTOR, pour gerer les moteurs des roues.

```
#include <MotorControl.h>
```

Attributs publics

— uint8_t cPin1

Pin control 1.

— uint8_t cPin2

Pin control 2.

— uint8_t enablePin

Pin de perminssion.

— int pwmChannel

Définition du canal PWM.

4.3.1 Description détaillée

Structure MOTOR, pour gerer les moteurs des roues.

Définition à la ligne 29 du fichier MotorControl.h.

4.3.2 Documentation des données membres

4.3.2.1 cPin1

```
uint8_t Motor::cPin1
```

Pin control 1.

Définition à la ligne 32 du fichier MotorControl.h.

4.3.2.2 cPin2

```
uint8_t Motor::cPin2
```

Pin control 2.

Définition à la ligne 35 du fichier MotorControl.h.

4.3.2.3 enablePin

```
uint8_t Motor::enablePin
```

Pin de perminssion.

Définition à la ligne 38 du fichier MotorControl.h.

4.3.2.4 pwmChannel

```
int Motor::pwmChannel
```

Définition du canal PWM.

Définition à la ligne 42 du fichier MotorControl.h.

La documentation de cette structure a été générée à partir du fichier suivant :
— RobotG7/include/MotorControl/MotorControl.h

4.4 Référence de la classe MotorControl

MotorControl pour gerer les moteur des roues.

```
#include <MotorControl.h>
```

Fonctions membres publiques

— MotorControl (MOTOR *roue1=NULL, MOTOR *roue2=NULL)

Le constructeur de la classe MotorControl.

— void begin ()

begin initialise les moteurs

— MOTOR * getRoue1 ()

Recupere la roue 1.

— MOTOR * getRoue2 ()

Recupere la roue 2.

4.4.1 Description détaillée

MotorControl pour gerer les moteur des roues.

Définition à la ligne 48 du fichier MotorControl.h.

4.4.2 Documentation des constructeurs et destructeur

4.4.2.1 MotorControl()

```
MotorControl::MotorControl (
            MOTOR * roue1 = NULL,
            MOTOR * roue2 = NULL )
```

Le constructeur de la classe MotorControl.

Paramètres

roue1	la premiere roue
roue2	la seconde roue

Définition à la ligne 14 du fichier MotorControl.cpp.

4.4.3 Documentation des fonctions membres

4.4.3.1 begin()

```
void MotorControl::begin ( )
```

begin initialise les moteurs

Définition à la ligne 36 du fichier MotorControl.cpp.

4.4.3.2 getRoue1()

```
MOTOR * MotorControl::getRoue1 ( )  [inline]
```

Recupere la roue 1.

Renvoie

MOTOR

Définition à la ligne 63 du fichier MotorControl.h.

4.4.3.3 getRoue2()

```
MOTOR * MotorControl::getRoue2 ( )  [inline]
```

Recupere la roue 2.

Renvoie

MOTOR

Définition à la ligne 67 du fichier MotorControl.h.

La documentation de cette classe a été générée à partir du fichier suivant :

- RobotG7/include/MotorControl/MotorControl.h
- RobotG7/include/MotorControl/MotorControl.cpp

4.5 Référence de la classe RobotG7

La class RobotG7 pour gerer le Système du robot.

```
#include <RobotG7.h>
```

Fonctions membres publiques

- RobotG7 (MOTOR *roue1=NULL, MOTOR *roue2=NULL, int pinCtlMode=5)

  RobotG7.

- void begin ()

  Methode begin() pour initialiser le système.

- uint32_t getDistanceObstacle ()

  Methode getDistanceObstacle() nous retournant la distance entre le robot et l'obstacle.

- int getpwmChannelRoue1 ()

  pour récuperer le canal pwm de la roue 1

- int getpwmChannelRoue2 ()

  pour récuperer le canal pwm de la roue 2

- void deplacementAvant ()

  Déplacement en avant.

- void deplacementArriere ()

  Déplacement en arrière.

- void tourner_a_GaucheAvant ()

Tourner à gauche en avant.

— void tourner_a_DroiteAvant ()

Tourner à droite en avant.

— void tourner_a_GaucheArriere ()

Tourner à gauche en arrière.

— void tourner_a_DroiteArriere ()

Tourner à droite en arrière.

— void stopRobotG7 ()

Stopper le déplacement.

— void writeSpeedOnChannel (int vitesseChannel1, int vitesseChannel2)

Donner la commande de déplacement suivant les vitesses des roues.

— int getVitesseGauche ()

retourner la vitesse du canal 1

— int getVitesseDroite ()

Retourner la vietesse du second canal.

— void setVitesseGauche (int vit)

Insérer la vitesse au canal de gauche.

— void setVitesseDroite (int vit)

Insérer la vitesse au canal de droite.

— void setNavigation (MODE modeNav)

Insérer le mode de navigation.

— WAY getSystemNavDirect ()

Savoir le type de chemin actuel.

— void navigation ()

La navigation su système.

— void setSystemNavDirect (WAY direction)

Insérer le type de chemin à parcourir.

— void systemRunningAuto ()

Ici le système fonctionne en mode automatique.

— void systemRunningManual ()

Ici le système fonction en mode Manuel.

4.5.1 Description détaillée

La class RobotG7 pour gerer le Système du robot.

Définition à la ligne 43 du fichier RobotG7.h.

4.5.2 Documentation des constructeurs et destructeur

4.5.2.1 RobotG7()

```
RobotG7::RobotG7 (
            MOTOR * roue1 = NULL,
            MOTOR * roue2 = NULL,
            int pinCtlMode = 5 )
```

RobotG7.

Paramètres

roue1	Pour lier le robot à la prémière roue du système
roue2	Pour lier le robot à la seconde roue du système

Définition à la ligne 24 du fichier RobotG7.cpp.

4.5.3 Documentation des fonctions membres

4.5.3.1 begin()

```
void RobotG7::begin ( )
```

Methode begin() pour initialiser le système.

Définition à la ligne 45 du fichier RobotG7.cpp.

4.5.3.2 deplacementArriere()

```
void RobotG7::deplacementArriere ( )
```

Déplacement en arrière.

Définition à la ligne 79 du fichier RobotG7.cpp.

4.5.3.3 deplacementAvant()

```
void RobotG7::deplacementAvant ( )
```

Déplacement en avant.

Définition à la ligne 65 du fichier RobotG7.cpp.

4.5.3.4 getDistanceObstacle()

```
uint32_t RobotG7::getDistanceObstacle ( )  [inline]
```

Methode getDistanceObstacle() nous retournant la distance entre le robot et l'obstacle.

Renvoie

uint32_t (int)

Définition à la ligne 58 du fichier RobotG7.h.

4.5.3.5 getpwmChannelRoue1()

```
int RobotG7::getpwmChannelRoue1 ( )  [inline]
```

pour récuperer le canal pwm de la roue 1

Renvoie

int, le numero du canal pwm

Définition à la ligne 61 du fichier RobotG7.h.

4.5.3.6 getpwmChannelRoue2()

```
int RobotG7::getpwmChannelRoue2 ( )  [inline]
```

pour récuperer le canal pwm de la roue 2

Renvoie

int, le numéro du canal pwm

Définition à la ligne 64 du fichier RobotG7.h.

4.5.3.7 getSystemNavDirect()

```
WAY RobotG7::getSystemNavDirect ( )  [inline]
```

Savoir le type de chemin actuel.

Renvoie

WAY, le chemin de parcours

Définition à la ligne 108 du fichier RobotG7.h.

4.5.3.8 getVitesseDroite()

```
int RobotG7::getVitesseDroite ( )  [inline]
```

Retourner la vietesse du second canal.

Renvoie

int, la vitesse du second canal

Définition à la ligne 90 du fichier RobotG7.h.

4.5.3.9 getVitesseGauche()

```
int RobotG7::getVitesseGauche ( )  [inline]
```

retourner la vitesse du canal 1

Renvoie

int, la vitesse du canal 1

Définition à la ligne 87 du fichier RobotG7.h.

4.5.3.10 navigation()

```
void RobotG7::navigation ( )
```

La navigation su système.

Définition à la ligne 341 du fichier RobotG7.cpp.

4.5.3.11 setNavigation()

```
void RobotG7::setNavigation (
            MODE modeNav )  [inline]
```

Insérer le mode de navigation.

Paramètres

modeNav,le	type de navigation

Définition à la ligne 102 du fichier RobotG7.h.

4.5.3.12 setSystemNavDirect()

```
void RobotG7::setSystemNavDirect (
            WAY direction )  [inline]
```

Insérer le type de chemin à parcourir.

Paramètres

direction,la	direction du chemin

Définition à la ligne 115 du fichier RobotG7.h.

4.5.3.13 setVitesseDroite()

```
void RobotG7::setVitesseDroite (
            int vit )  [inline]
```

Insérer la vitesse au canal de droite.

Paramètres

vit,la	vitesse à insérer

Définition à la ligne 96 du fichier RobotG7.h.

4.5.3.14 setVitesseGauche()

```
void RobotG7::setVitesseGauche (
            int vit )  [inline]
```

Insérer la vitesse au canal de gauche.

Paramètres

vit,la	vitesse à insérer

Définition à la ligne 93 du fichier RobotG7.h.

4.5.3.15 stopRobotG7()

```
void RobotG7::stopRobotG7 ( )
```

Stopper le déplacement.

Définition à la ligne 144 du fichier RobotG7.cpp.

4.5.3.16 systemRunningAuto()

```
void RobotG7::systemRunningAuto ( )
```

Ici le système fonctionne en mode automatique.

Définition à la ligne 162 du fichier RobotG7.cpp.

4.5.3.17 systemRunningManual()

```
void RobotG7::systemRunningManual ( )
```

Ici le système fonction en mode Manuel.

Définition à la ligne 311 du fichier RobotG7.cpp.

4.5.3.18 tourner_a_DroiteArriere()

```
void RobotG7::tourner_a_DroiteArriere ( )
```

Tourner à droite en arrière.

Définition à la ligne 131 du fichier RobotG7.cpp.

4.5.3.19 tourner_a_DroiteAvant()

```
void RobotG7::tourner_a_DroiteAvant ( )
```

Tourner à droite en avant.

Définition à la ligne 105 du fichier RobotG7.cpp.

4.5.3.20 tourner_a_GaucheArriere()

```
void RobotG7::tourner_a_GaucheArriere ( )
```

Tourner à gauche en arrière.

Définition à la ligne 119 du fichier RobotG7.cpp.

4.5.3.21 tourner_a_GaucheAvant()

```
void RobotG7::tourner_a_GaucheAvant ( )
```

Tourner à gauche en avant.

Définition à la ligne 93 du fichier RobotG7.cpp.

4.5.3.22 writeSpeedOnChannel()

```
void RobotG7::writeSpeedOnChannel (
            int vitesseChannel1,
            int vitesseChannel2 )
```

Donner la commande de déplacement suivant les vitesses des roues.

Paramètres

vitesseChannel1,la	vitesse au prémier canal
vitesseChannel2,la	vitesse au second canal

Définition à la ligne 56 du fichier RobotG7.cpp.

La documentation de cette classe a été générée à partir du fichier suivant :

- RobotG7/RobotG7.h
- RobotG7/RobotG7.cpp

Chapitre 5

Documentation des fichiers

5.1 Référence du fichier RobotG7/include/CtrlWay/CtrlWay.cpp

```
#include "CtrlWay.h"
```

Macros

— #define INIT_POS_MOTOR 90

— #define TAILLE_TAB_DIST 5

Variables

— unsigned long previouMillis

Gestion du temps.

5.1.1 Documentation des macros

5.1.1.1 INIT_POS_MOTOR

```
#define INIT_POS_MOTOR 90
```

Définition à la ligne 13 du fichier CtrlWay.cpp.

5.1.1.2 TAILLE_TAB_DIST

```
#define TAILLE_TAB_DIST 5
```

Définition à la ligne 14 du fichier CtrlWay.cpp.

5.1.2 Documentation des variables

5.1.2.1 previouMillis

```
unsigned long previouMillis  [extern]
```

Gestion du temps.

Définition à la ligne 45 du fichier main.cpp.

5.2 CtrlWay.cpp

Aller à la documentation de ce fichier.

```
00001 /*****************************************************************************
00002 ******************************************************************************
00003 ** @author : André MUTOKE MUSULE & Japhet TSHIMANGA BWENDE                    **
00004 ** @date : Mardi 29 Mai 2024                                                 **
00005 ** @name : CtrlWay.cpp                                                       **
00006 ** @details : Ce fichier implémente la prise de décison du chemin            **
00007 * ******************************************************************************
00008 * ******************************************************************************
00009 */
00010
00011 #include "CtrlWay.h"
00012
00013 #define INIT_POS_MOTOR 90
00014 #define TAILLE_TAB_DIST 5
00015 extern unsigned long previouMillis;
00016
00017 CtrlWay::CtrlWay(int pinAttached)
00018 {
00019     this->m_pin_to_attach = pinAttached;
00020     this->posMotor = INIT_POS_MOTOR;
00021 }
00022 void CtrlWay::begin()
00023 {
00024     this->m_pServo.attach(this->m_pin_to_attach);
00025     this->m_pServo.write(90);
00026
00027
00028 }
00029 int CtrlWay::seeToLeft(GuideSound *CaptureObstacle)
00030 {
00031     int tabDistance[TAILLE_TAB_DIST]; // Pour capturer les valeurs des distance
00032     // On tourne a gauche
00033     unsigned long currentTimefor = millis();
00034     Serial.println("On cherche a verifier a gauche");
00035
00036     delay(500);
00037     this->m_pServo.write(0);
00038     delay(500);
00039
00040     Serial.println("On a verifier a gauche");
00041     // On observe la distance des obstacles a gauches
00042     previouMillis = millis();
00043
00044         for(int i=0; i<TAILLE_TAB_DIST; ++i)
00045         {
00046             tabDistance[i] = CaptureObstacle->getDistance();
```

```
00047         }
00048     delay(500);
00049     this->m_pServo.write(this->posMotor);
00050     delay(500);
00051 
00052     Serial.print("La valeur a gauche est : ");
00053     Serial.println(this->MoyenneDistances(tabDistance));
00054     return this->MoyenneDistances(tabDistance);
00055 }
00056 
00057 int CtrlWay::seeToRight(GuideSound *CaptureObstacle)
00058 {
00059     int tabDistance[TAILLE_TAB_DIST]; // Pour capturer les valeurs des distance
00060     // On tourne a gauche
00061     unsigned long currentTimefor = millis();
00062     delay(500);
00063     this->m_pServo.write(INIT_POS_MOTOR + 90);
00064     delay(500);
00065 
00066     // On observe la distance des obstacles a gauches
00067     previouMillis = millis();
00068 
00069     if(currentTimefor - previouMillis >= 2000)
00070     {
00071         for(int i=0; i<TAILLE_TAB_DIST; ++i)
00072         {
00073             tabDistance[i] = CaptureObstacle->getDistance();
00074         }
00075     }
00076     delay(500);
00077     this->m_pServo.write(this->posMotor);
00078     delay(500);
00079 
00080     return this->MoyenneDistances(tabDistance);
00081 }
00082 
00083 int CtrlWay::MoyenneDistances(int tab[])
00084 {
00085     int moyenne = 0;
00086     for(int i = 0; i< TAILLE_TAB_DIST; ++i)
00087     {
00088         moyenne = moyenne + tab[i];
00089     }
00090     moyenne = moyenne / TAILLE_TAB_DIST;
00091     return moyenne;
00092 }
00093 
00094 WAY CtrlWay::theWayToget(GuideSound *CaptureObstacle)
00095 {
00096     int way1 = this->seeToLeft(CaptureObstacle);
00097     int way2 = this->seeToRight(CaptureObstacle);
00098 
00099     if(way1 > way2) {
00100         Serial.println("On chisit la gauche");
00101         return GAUCHE;
00102     }
00103     if(way1 < way2) {
00104         Serial.println("On chisit la droite");
00105         return DROITE;
00106     }
00107     return GAUCHE;
00108 
00109 }
```

5.3 Référence du fichier RobotG7/include/CtrlWay/CtrlWay.h

```
#include <Arduino.h>
#include <ESP32Servo.h>
#include "../GuideSound/GuideSound.h"
```

Classes

— class CtrlWay

La class CtrlWay pour gerer la facon de réagir face à l'obstacle.

Macros

— #define CTRL_WAY

Définitions de type

— typedef enum Way WAY

Énumérations

— enum Way { DEVANT , DERRIERE , GAUCHE , DROITE }

La structure Way contient les différents chemins de navigation Nous avons DEVANT, DERRIERE, GAUCHE et DROITE.

5.3.1 Documentation des macros

5.3.1.1 CTRL_WAY

```
#define CTRL_WAY
```

Définition à la ligne 13 du fichier CtrlWay.h.

5.3.2 Documentation des définitions de type

5.3.2.1 WAY

```
typedef enum Way WAY
```

Définition à la ligne 29 du fichier CtrlWay.h.

5.3.3 Documentation du type de l'énumération

5.3.3.1 Way

```
enum Way
```

La structure Way contient les différents chemins de navigation Nous avons DEVANT, DERRIERE, GAUCHE et DROITE.

Valeurs énumérées

DEVANT	
DERRIERE	
GAUCHE	
DROITE	

Définition à la ligne 22 du fichier CtrlWay.h.

5.4 CtrlWay.h

Aller à la documentation de ce fichier.

```
00001 /*****************************************************************************
00002 ******************************************************************************
00003 ** @author : André MUTOKE MUSULE & Japhet TSHIMANGA BWENDE                    **
00004 ** @date : Mardi 29 Mai 2024                                                 **
00005 ** @name : CtrlWay.h                                                         **
00006 ** @details : Ce fichier implémente la prise de décison du chemin            **
00007 * ******************************************************************************
00008 * ******************************************************************************
00009 */
00010
00011 #pragma once
00012 #ifndef CTRL_WAY
00013 #define CTRL_WAY
00014
00015 #include <Arduino.h>
00016 #include <ESP32Servo.h>
00017
00018 #include "../GuideSound/GuideSound.h"
00019
00022 enum Way
00023 {
00024     DEVANT,
00025     DERRIERE,
00026     GAUCHE,
00027     DROITE
00028 };
00029 typedef enum Way WAY;
00030
00032 class CtrlWay
00033 {
00034 public :
00037     CtrlWay(int pinAttached = 15);
00039     void begin();
00043     WAY theWayToget(GuideSound *CaptureObstacle);
00044
00045 private :
00047     Servo m_pServo;
00048
00049     uint8_t m_pin_to_attach;
00050     int posMotor;
00054     int seeToLeft(GuideSound *CaptureObstacle);
00058     int seeToRight(GuideSound *CaptureObstacle);
00062     int MoyenneDistances(int tab[]);
00063
00064 };
00065
00066 #endif // CTRL_WAY
```

5.5 Référence du fichier RobotG7/include/GuideSound/GuideSound.cpp

```
#include "GuideSound.h"
```

Macros

- #define __PULSE_DOWN__ 2
- #define __PULSE_UP__ 10
- #define __PIN_OK__ 1
- #define __PIN_NOT_OK__ 0

5.5.1 Documentation des macros

5.5.1.1 __PIN_NOT_OK__

```
#define __PIN_NOT_OK__ 0
```

Définition à la ligne 18 du fichier GuideSound.cpp.

5.5.1.2 __PIN_OK__

```
#define __PIN_OK__ 1
```

Définition à la ligne 17 du fichier GuideSound.cpp.

5.5.1.3 __PULSE_DOWN__

```
#define __PULSE_DOWN__ 2
```

Définition à la ligne 14 du fichier GuideSound.cpp.

5.5.1.4 __PULSE_UP__

```
#define __PULSE_UP__ 10
```

Définition à la ligne 15 du fichier GuideSound.cpp.

5.6 GuideSound.cpp

Aller à la documentation de ce fichier.

```
00001 /*****************************************************************************************
00002 ******************************************************************************************
00003 ** @author : André MUTOKE MUSULE & Japhet TSHIMANGA BWENDE                                **
00004 ** @date : Mardi 29 Mai 2024                                                              **
00005 ** @name : GuideSound.cpp                                                                 **
00006 ** @details : Ce fichier implémente le capteur ultra son                                  **
00007 * ******************************************************************************************
00008 * ******************************************************************************************
00009 */
00010
00011
00012 #include "GuideSound.h"
00013
00014 #define __PULSE_DOWN__ 2
00015 #define __PULSE_UP__ 10
00016
00017 #define __PIN_OK__ 1
00018 #define __PIN_NOT_OK__ 0
00019
00020 GuideSound::GuideSound(uint8_t trig, uint8_t echo)
00021 {
00022     this->m_trig_pin = trig;
00023     this->m_echo_pin= echo;
00024 }
00025
00026
00027 void GuideSound::begin()
00028 {
00029
00030     pinMode(this->m_trig_pin, OUTPUT);
00031     pinMode(this->m_echo_pin, INPUT);
00032
00033     digitalWrite(this->m_trig_pin, LOW);
00034 }
00035
00036 int GuideSound::pulseEmit()
00037 {
00038     if(this->m_trig_pin != 0)
00039     {
00040         digitalWrite(this->m_trig_pin, LOW); delayMicroseconds(__PULSE_DOWN__);
00041         digitalWrite(this->m_trig_pin, HIGH); delayMicroseconds(__PULSE_UP__);
00042         digitalWrite(this->m_trig_pin, LOW);
00043
00044         return __PIN_OK__;
00045
00046     }
00047     return __PIN_NOT_OK__;
00048
00049 }
00050
00051 void GuideSound::captureDistance_and_Delay()
00052 {
00053     if(this->pulseEmit())
00054     {
00055         if(this->m_echo_pin)
00056         {
00057             this->m_duree = pulseIn(this->m_echo_pin, HIGH);
00058             this->m_distance = this->m_duree / 58.2;
00059         }
00060
00061     }
00062 }
```

5.7 Référence du fichier RobotG7/include/GuideSound/GuideSound.h

```
#include <Arduino.h>
#include <Wire.h>
```

Classes

— class GuideSound

GuideSound est la classe qui gere le capteur Ultra Son.

Macros

— #define GUIDE_SOUND_H

5.7.1 Documentation des macros

5.7.1.1 GUIDE_SOUND_H

```
#define GUIDE_SOUND_H
```

Définition à la ligne 14 du fichier GuideSound.h.

5.8 GuideSound.h

Aller à la documentation de ce fichier.

```
00001 /*****************************************************************************************
00002 ******************************************************************************************
00003 ** @author : André MUTOKE MUSULE & Japhet TSHIMANGA BWENDE                              **
00004 ** @date : Mardi 29 Mai 2024                                                           **
00005 ** @name : GuideSound.h                                                                **
00006 ** @details : Ce fichier implémente le capteur ultra son                              **
00007 * ******************************************************************************************
00008 * ******************************************************************************************
00009 */
00010
00011
00012 #pragma once
00013 #ifndef GUIDE_SOUND_H
00014 #define GUIDE_SOUND_H
00015
00016 #include <Arduino.h>
00017 #include <Wire.h>
00018
00020 class GuideSound
00021 {
00022 public :
00023
00027     GuideSound(uint8_t trig = 0, uint8_t echo = 0);
00029     void begin(); // Initialiser le programme;
00030
00033     uint32_t getDistance(){ captureDistance_and_Delay(); return this->m_distance;}
00034
00037     uint32_t getDelayEcho(){ captureDistance_and_Delay(); return this->m_duree;} // POur nous
      retourner le temps de voyage capturé
00038 private:
00039     uint8_t m_echo_pin; // Le pin pour recevoir les echos retours
00040     uint8_t m_trig_pin; // Le pin pour Emettre le son
00041
00042     uint32_t m_distance; // Pour calculer et concerver la distance
00043     uint32_t m_duree;
00044
00045
00047     void captureDistance_and_Delay();
00050     int pulseEmit();
00051
00052 };
00053
00054 #endif // GUIDE_SOUND_H
```

5.9 Référence du fichier RobotG7/include/MotorControl/MotorControl.cpp

```
#include "MotorControl.h"
```

5.10 MotorControl.cpp

Aller à la documentation de ce fichier.

```
00001 /*****************************************************************************
00002 ******************************************************************************
00003 ** @author : André MUTOKE MUSULE & Japhet TSHIMANGA BWENDE                    **
00004 ** @date : Mardi 29 Mai 2024                                                 **
00005 ** @name : MotorControl.cpp                                                  **
00006 ** @details : Ce fichier implémente les engrenages moteurs pour les roues    **
00007 * ******************************************************************************
00008 * ******************************************************************************
00009 */
00010
00011
00012 #include "MotorControl.h"
00013
00014 MotorControl::MotorControl(MOTOR *roue1, MOTOR *roue2)
00015 {
00016     this->m_roue1 = roue1;
00017     this->m_roue2 = roue2;
00018 }
00019
00020 void MotorControl::configurationMotor()
00021 {
00022     pinMode(this->m_roue1->cPin1, OUTPUT);
00023     pinMode(this->m_roue1->cPin2, OUTPUT);
00024     pinMode(this->m_roue1->enablePin, OUTPUT);
00025
00026     pinMode(this->m_roue2->cPin1, OUTPUT);
00027     pinMode(this->m_roue2->cPin2, OUTPUT);
00028     pinMode(this->m_roue2->enablePin, OUTPUT);
00029
00030     ledcSetup(PWMCHANNEL, FREQENCE_CONTROL, RESOLUTION);
00031
00032     ledcAttachPin(this->m_roue1->enablePin, this->m_roue1->pwmChannel);
00033     ledcAttachPin(this->m_roue2->enablePin, this->m_roue2->pwmChannel);
00034 }
00035
00036 void MotorControl::begin()
00037 {
00038     configurationMotor();
00039 }
```

5.11 Référence du fichier RobotG7/include/MotorControl/MotorControl.h

```
#include <Arduino.h>
```

Classes

— struct Motor

Structure MOTOR, pour gerer les moteurs des roues.

— class MotorControl

MotorControl pour gerer les moteur des roues.

Macros

— #define MOTOR_CONTROL_H

— #define FREQENCE_CONTROL 30000

— #define PWMCHANNEL 0

— #define PWMCHANNEL2 1

— #define RESOLUTION 8

— #define DUTY_CYCLE 170

Définitions de type

— typedef struct Motor MOTOR

5.11.1 Documentation des macros

5.11.1.1 DUTY_CYCLE

```
#define DUTY_CYCLE 170
```

Définition à la ligne 25 du fichier MotorControl.h.

5.11.1.2 FREQENCE_CONTROL

```
#define FREQENCE_CONTROL 30000
```

PROPRIETIES PWM

Définition à la ligne 21 du fichier MotorControl.h.

5.11.1.3 MOTOR_CONTROL_H

```
#define MOTOR_CONTROL_H
```

Définition à la ligne 13 du fichier MotorControl.h.

5.11.1.4 PWMCHANNEL

```
#define PWMCHANNEL 0
```

Définition à la ligne 22 du fichier MotorControl.h.

5.11.1.5 PWMCHANNEL2

```
#define PWMCHANNEL2 1
```

Définition à la ligne 23 du fichier MotorControl.h.

5.11.1.6 RESOLUTION

```
#define RESOLUTION 8
```

Définition à la ligne 24 du fichier MotorControl.h.

5.11.2 Documentation des définitions de type

5.11.2.1 MOTOR

```
typedef struct Motor MOTOR
```

Définition à la ligne 45 du fichier MotorControl.h.

5.12 MotorControl.h

Aller à la documentation de ce fichier.

```
00001 /*****************************************************************************
00002 ******************************************************************************
00003 ** @author : André MUTOKE MUSULE & Japhet TSHIMANGA BWENDE                    **
00004 ** @date : Mardi 29 Mai 2024                                                 **
00005 ** @name : MotorControl.h                                                    **
00006 ** @details : Ce fichier implémente les engrenages moteurs pour les roues    **
00007 * ******************************************************************************
00008 * ******************************************************************************
00009 */
00010
00011 #pragma once
00012 #ifndef MOTOR_CONTROL_H
00013 #define MOTOR_CONTROL_H
00014
00015 #include <Arduino.h>
00016
00017
00021 #define FREQENCE_CONTROL 30000
00022 #define PWMCHANNEL 0
00023 #define PWMCHANNEL2 1
```

```
00024 #define RESOLUTION 8
00025 #define DUTY_CYCLE 170
00026
00028
00029 struct Motor
00030 {
00032     uint8_t cPin1; // Pin Control 1
00033
00035     uint8_t cPin2; // Pin Control 2
00036
00038     uint8_t enablePin; // Pin four enabling
00039
00040
00042     int pwmChannel; // PWMChannel
00043 };
00044
00045 typedef struct Motor MOTOR;
00046
00048 class MotorControl
00049 {
00050 public :
00051
00055     MotorControl(MOTOR *roue1 = NULL,
00056                  MOTOR *roue2  = NULL
00057                  );
00059     void begin();
00060
00063     MOTOR *getRoue1(){return this->m_roue1;}
00064
00067     MOTOR *getRoue2(){return this->m_roue2;}
00068 private:
00069     MOTOR *m_roue1; // Moteur de la roue 1
00070     MOTOR *m_roue2; // Moteur de la roue 2
00071
00073     void configurationMotor();
00074 };
00075
00076 #endif // MOTOR_CONTROL_H
```

5.13 Référence du fichier RobotG7/RobotG7.cpp

```
#include "RobotG7.h"
```

Macros

- #define DEFAULT_TRIG_PIN 4
- #define DEFAULT_ECHO_PIN 2
- #define PIN_CLEANNIG 18
- #define LIMIT_GET_OBSTACLE 10

Variables

- unsigned long previouMillis

  Gestion du temps.

- int tempctrl

5.13.1 Documentation des macros

5.13.1.1 DEFAULT_ECHO_PIN

```
#define DEFAULT_ECHO_PIN 2
```

Définition à la ligne 16 du fichier RobotG7.cpp.

5.13.1.2 DEFAULT_TRIG_PIN

```
#define DEFAULT_TRIG_PIN 4
```

Définition à la ligne 15 du fichier RobotG7.cpp.

5.13.1.3 LIMIT_GET_OBSTACLE

```
#define LIMIT_GET_OBSTACLE 10
```

Définition à la ligne 19 du fichier RobotG7.cpp.

5.13.1.4 PIN_CLEANNIG

```
#define PIN_CLEANNIG 18
```

Définition à la ligne 17 du fichier RobotG7.cpp.

5.13.2 Documentation des variables

5.13.2.1 previouMillis

```
unsigned long previouMillis  [extern]
```

Gestion du temps.

Définition à la ligne 45 du fichier main.cpp.

5.13.2.2 tempctrl

```
int tempctrl  [extern]
```

Définition à la ligne 42 du fichier main.cpp.

5.14 RobotG7.cpp

Aller à la documentation de ce fichier.

```
00001 /*****************************************************************************************
00002 *****************************************************************************************
00003 ** @author : André MUTOKE MUSULE & Japhet TSHIMANGA BWENDE                                **
00004 ** @date : Mardi 29 Mai 2024                                                             **
00005 ** @name : RobotG7.cpp                                                                   **
00006 ** @details : Ce fichier implémente les élément composant le robot du système           **
00007 * *****************************************************************************************
00008 * *****************************************************************************************
00009 */
00010
00011 #include "RobotG7.h"
00012
00013
00014
00015 #define DEFAULT_TRIG_PIN 4
00016 #define DEFAULT_ECHO_PIN 2
00017 #define PIN_CLEANNIG 18
00018
00019 #define LIMIT_GET_OBSTACLE 10 // 10 cm
00020 extern unsigned long previouMillis;
00021
00022 extern int tempctrl;
00023
00024 RobotG7::RobotG7(MOTOR *roue1, MOTOR *roue2, int pinCtrlMode)
00025 {
00026     this->m_gdSound = new GuideSound(DEFAULT_TRIG_PIN, DEFAULT_ECHO_PIN); // (triger, echo)
00027     this->m_rouesRobots = new MotorControl(roue1, roue2);
00028     this->m_vitesseDroite = 190;
00029     this->m_vitesseGauche = 190;
00030
00031     this->cheminAprendre = new CtrlWay(); // Par defaut à la pin 15
00032
00033     this->systemNav = DEVANT;
00034
00035     this->m_modeNav = MODE_START;
00036     this->modeManualnotPrepared = ROBOTG7_TRUE;
00037
00038     this->enableCleanning = ROBOTG7_FALSE;
00039
00040     // this->m_modeNav = MODE_AUTOMATIQUE;
00041
00042     // this->ctrlMode = pinCtrlMode;
00043 }
00044
00045 void RobotG7::begin()
00046 {
00047     // pinMode(this->ctrlMode, INPUT_PULLUP);
00048     pinMode(PIN_CLEANNIG, OUTPUT);
00049
00050     this->m_gdSound->begin();
00051     this->m_rouesRobots->begin();
00052     this->cheminAprendre->begin();
00053     digitalWrite(PIN_CLEANNIG, HIGH);
00054 }
00055
00056 void RobotG7::writeSpeedOnChannel(int vitesseChannel1, int vitesseChannel2)
00057 {
00058     this->m_vitesseDroite = vitesseChannel2;
00059     this->m_vitesseGauche = vitesseChannel1;
00060
00061     ledcWrite(this->m_rouesRobots->getRoue1()->pwmChannel, this->m_vitesseGauche);
00062     ledcWrite(this->m_rouesRobots->getRoue2()->pwmChannel, this->m_vitesseDroite);
00063 }
00064
00065 void RobotG7::deplacementAvant()
00066 {
00067     digitalWrite(this->m_rouesRobots->getRoue1()->enablePin, HIGH);
00068     digitalWrite(this->m_rouesRobots->getRoue2()->enablePin, HIGH);
00069
00070     digitalWrite(this->m_rouesRobots->getRoue1()->cPin1, LOW);
00071     digitalWrite(this->m_rouesRobots->getRoue1()->cPin2, HIGH);
00072     digitalWrite(this->m_rouesRobots->getRoue2()->cPin1, HIGH);
00073     digitalWrite(this->m_rouesRobots->getRoue2()->cPin2, LOW);
00074
00075
00076     this->writeSpeedOnChannel(200, 200);
00077
00078 }
00079 void RobotG7::deplacementArriere()
00080 {
00081     digitalWrite(this->m_rouesRobots->getRoue1()->enablePin, HIGH);
00082     digitalWrite(this->m_rouesRobots->getRoue2()->enablePin, HIGH);
```

```
00083
00084     digitalWrite(this->m_rouesRobots->getRoue1()->cPin1, HIGH);
00085     digitalWrite(this->m_rouesRobots->getRoue1()->cPin2, LOW);
00086     digitalWrite(this->m_rouesRobots->getRoue2()->cPin1, LOW);
00087     digitalWrite(this->m_rouesRobots->getRoue2()->cPin2, HIGH);
00088
00089     this->writeSpeedOnChannel(190, 190);
00090
00091 }
00092
00093 void RobotG7::tourner_a_GaucheAvant()
00094 {
00095     digitalWrite(this->m_rouesRobots->getRoue1()->enablePin, HIGH);
00096     digitalWrite(this->m_rouesRobots->getRoue2()->enablePin, LOW);
00097
00098     digitalWrite(this->m_rouesRobots->getRoue1()->cPin1, LOW);
00099     digitalWrite(this->m_rouesRobots->getRoue1()->cPin2, HIGH);
00100     // digitalWrite(this->m_rouesRobots->getRoue2()->cPin1, LOW);
00101     // digitalWrite(this->m_rouesRobots->getRoue2()->cPin1, HIGH);
00102
00103     this->writeSpeedOnChannel(190, 0);
00104 }
00105 void RobotG7::tourner_a_DroiteAvant()
00106 {
00107     digitalWrite(this->m_rouesRobots->getRoue1()->enablePin, LOW);
00108     digitalWrite(this->m_rouesRobots->getRoue2()->enablePin, HIGH);
00109
00110     // digitalWrite(this->m_rouesRobots->getRoue1()->cPin1, LOW);
00111     // digitalWrite(this->m_rouesRobots->getRoue1()->cPin2, HIGH);
00112     digitalWrite(this->m_rouesRobots->getRoue2()->cPin1, HIGH);
00113     digitalWrite(this->m_rouesRobots->getRoue2()->cPin2, LOW);
00114
00115     this->writeSpeedOnChannel(0, 190);
00116 }
00117
00118
00119 void RobotG7::tourner_a_GaucheArriere()
00120 {
00121     digitalWrite(this->m_rouesRobots->getRoue1()->enablePin, LOW);
00122     digitalWrite(this->m_rouesRobots->getRoue2()->enablePin, HIGH);
00123
00124     // digitalWrite(this->m_rouesRobots->getRoue1()->cPin1, LOW);
00125     // digitalWrite(this->m_rouesRobots->getRoue1()->cPin2, HIGH);
00126     digitalWrite(this->m_rouesRobots->getRoue2()->cPin1, LOW);
00127     digitalWrite(this->m_rouesRobots->getRoue2()->cPin2, HIGH);
00128
00129     this->writeSpeedOnChannel(190, 0);
00130 }
00131 void RobotG7::tourner_a_DroiteArriere()
00132 {
00133     digitalWrite(this->m_rouesRobots->getRoue1()->enablePin, HIGH);
00134     digitalWrite(this->m_rouesRobots->getRoue2()->enablePin, LOW);
00135
00136     digitalWrite(this->m_rouesRobots->getRoue1()->cPin1, HIGH);
00137     digitalWrite(this->m_rouesRobots->getRoue1()->cPin2, LOW);
00138     // digitalWrite(this->m_rouesRobots->getRoue2()->cPin1, LOW);
00139     // digitalWrite(this->m_rouesRobots->getRoue2()->cPin1, HIGH);
00140
00141     this->writeSpeedOnChannel(0, 190);
00142 }
00143
00144 void RobotG7::stopRobotG7()
00145 {
00146     digitalWrite(this->m_rouesRobots->getRoue1()->enablePin, LOW);
00147     digitalWrite(this->m_rouesRobots->getRoue2()->enablePin, LOW);
00148
00149     this->writeSpeedOnChannel(0, 0);
00150
00151 }
00152
00153 void RobotG7::gamePadAutoGoStart()
00154 {
00155      if(GamePad.isStartPressed())
00156     {
00157         tempctrl = 1;
00158     }
00159
00160 }
00161
00162 void RobotG7::systemRunningAuto()
00163 {
00164     this->prepareRunningManual();
00165     this->inputProcess();
00166
00167     switch(this->systemNav)
00168     {
00169     case DEVANT :
```

```
00170         Serial.println("On avance !!!"); this->gamePadAutoGoStart();
00171         if(this->getDistanceObstacle() >= LIMIT_GET_OBSTACLE)
00172         {
00173             this->enableCleanning = ROBOTG7_TRUE;this->gamePadAutoGoStart();
00174             Serial.println("Permit d'avancer !!!");
00175             this->deplacementAvant();
00176         }
00177         else
00178         {
00179             Serial.println("Non Permit d'avancer !!!");
00180             this->stopRobotG7();this->gamePadAutoGoStart();
00181             this->enableCleanning = ROBOTG7_FALSE;
00182             this->systemNav = this->cheminAprendre->theWayToget(this->m_gdSound);
00183         }
00184         break;
00185     case DERRIERE :
00186         this->enableCleanning = ROBOTG7_TRUE;this->gamePadAutoGoStart();
00187         this->deplacementArriere();
00188         break;
00189     case GAUCHE :
00190         this->enableCleanning = ROBOTG7_TRUE;this->gamePadAutoGoStart();
00191         this->tourner_a_GaucheArriere();
00192         this->systemNav = DEVANT;
00193         break;
00194     case DROITE :
00195         this->enableCleanning = ROBOTG7_TRUE;this->gamePadAutoGoStart();
00196         this->tourner_a_DroiteArriere();
00197         this->systemNav = DEVANT;
00198         break;
00199     }
00200 
00201     if(this->enableCleanning)
00202     {
00203         digitalWrite(PIN_CLEANNIG, HIGH);this->gamePadAutoGoStart();
00204     }
00205     else
00206     {
00207         digitalWrite(PIN_CLEANNIG, LOW);this->gamePadAutoGoStart();
00208     }
00209     this->gamePadAutoGoStart();
00210 }
00211 
00212 void RobotG7::inputProcess()
00213 {
00214     Dabble.processInput();
00215 }
00216 
00217 void RobotG7::obtenirVitesseJoystick()
00218 {
00219     float xval = GamePad.getXaxisData();
00220   float yval = GamePad.getYaxisData();
00221 
00222   //Pour la calibration
00223   if (xval < -6) {
00224     xval = -6;
00225     this->gamePadGestionCommand();
00226   } else if (xval > 6) {
00227     xval = 6;
00228     this->gamePadGestionCommand();
00229   }
00230   if (yval < -6) {
00231     yval = -6;this->gamePadGestionCommand();
00232   } else if (yval > 6) {
00233     yval = 6;this->gamePadGestionCommand();
00234   }
00235 
00236   int yMapped = 0;
00237   if (yval < 0 )
00238   {
00239     this->deplacementArriere();this->gamePadGestionCommand();
00240     yMapped = map(yval, -6, 0, 250, 0);
00241   }
00242   else
00243   {
00244     this->deplacementAvant();this->gamePadGestionCommand();
00245     yMapped = map(yval, 0, 6, 0, 250);
00246   }
00247 
00248   int xMapped;
00249   if (xval < 0 ){
00250     xMapped = map(xval, -6, 0, 250, 0);this->gamePadGestionCommand();
00251   }
00252   else {
00253     xMapped = map(xval, 0, 6, 0, 250);this->gamePadGestionCommand();
00254   }
00255 
00256   if (xval < 0 ) {
```

```
00257     this->m_vitesseGauche = yMapped + xMapped;this->gamePadGestionCommand();
00258     if (this->m_vitesseGauche  > 250) {
00259       this->m_vitesseGauche  = 250;this->gamePadGestionCommand();
00260     }
00261 
00262     this->m_vitesseDroite = yMapped - xMapped;this->gamePadGestionCommand();
00263     if (this->m_vitesseDroite < 0) {this->gamePadGestionCommand();
00264       this->m_vitesseDroite = 0;
00265     }
00266 
00267   } else {
00268     this->m_vitesseGauche = yMapped - xMapped;this->gamePadGestionCommand();
00269     if (this->m_vitesseGauche < 0) {this->gamePadGestionCommand();
00270       this->m_vitesseGauche = 0;
00271     }// pwmchannel
00272 
00273     this->m_vitesseDroite = yMapped + xMapped;this->gamePadGestionCommand();
00274     if (this->m_vitesseDroite > 250) {this->gamePadGestionCommand();
00275       this->m_vitesseDroite = 250;
00276     }
00277   }
00278 
00279 
00280 }
00281 
00282 void RobotG7::gamePadGestionCommand()
00283 {
00284     if(GamePad.isCrossPressed())
00285     {
00286         Serial.println("On nettoie");
00287         this->enableCleanning = ROBOTG7_TRUE;
00288     }
00289     else
00290     {
00291         Serial.println("On ne nettoie pas");
00292         this->enableCleanning = ROBOTG7_FALSE;
00293     }
00294     if(this->enableCleanning)
00295     {
00296         Serial.println("On nettoie ici");
00297         digitalWrite(PIN_CLEANNIG, HIGH);
00298     }
00299     else
00300     {
00301         Serial.println("On ne nettoie pas ici");
00302         digitalWrite(PIN_CLEANNIG, LOW);
00303     }
00304 
00305     if(GamePad.isSelectPressed())
00306     {
00307         tempctrl = 3;
00308     }
00309 }
00310 
00311 void RobotG7::systemRunningManual()
00312 {
00313     this->prepareRunningManual();
00314     this->inputProcess();
00315     this->obtenirVitesseJoystick();
00316     this->writeSpeedOnChannel(this->m_vitesseGauche, this->m_vitesseDroite);
00317 }
00318 
00319 // void RobotG7::runningAutoMode(uint32_t PreviousTime)
00320 // {
00321 
00322 // }
00323 
00324 // void RobotG7::systemRunning()
00325 // {
00326 //     switch (this->m_modeNav)
00327 //     {
00328 //     case MODE_AUTOMATIQUE:
00329 //         /* code */
00330 //         break;
00331 //     case MODE_MANUEL:
00332 //         /* code */
00333 //         break;
00334 
00335 //     default:
00336 //         break;
00337 //     }
00338 // }
00339 
00340 // ++++++++++++++++++++++++ mode de navigation ++++++++++++++++++++++++
00341 void RobotG7::navigation()
00342 {
00343     switch(this->m_modeNav)
```

```
00344     {
00345     case MODE_START :
00346         Serial.println("Mode start !!!");
00347 
00348         this->modeManualnotPrepared = ROBOTG7_TRUE;
00349 
00350         if(this->enableCleanning)
00351         {
00352             digitalWrite(PIN_CLEANNIG, HIGH);
00353         }
00354         else
00355         {
00356             digitalWrite(PIN_CLEANNIG, LOW);
00357         }
00358         this->stopRobotG7();
00359         break;
00360     case  MODE_AUTOMATIQUE :
00361         Serial.println("Mode automatique !!!");
00362         this->systemRunningAuto();
00363         break;
00364 
00365     case MODE_MANUEL:
00366         Serial.println("Mode manuel !!!");
00367         // this->stopRobotG7();
00368         this->systemRunningManual();
00369         break;
00370 
00371     }
00372 }
```

5.15 Référence du fichier RobotG7/RobotG7.h

```
#include <Arduino.h>
#include <DabbleESP32.h>
#include "include/GuideSound/GuideSound.h"
#include "include/MotorControl/MotorControl.h"
#include "include/CtrlWay/CtrlWay.h"
```

Classes

— class RobotG7

La class RobotG7 pour gerer le Système du robot.

Macros

— #define CUSTOM_SETTINGS

Dabble bluetooth////.

— #define INCLUDE_GAMEPAD_MODULEpowerChannel

— #define ROBOTG7_FALSE 0

FIN Dabble bluetooth////.

— #define ROBOTG7_TRUE 1

Définitions de type

— typedef enum Navigation MODE

Énumérations

— enum Navigation { MODE_START , MODE_AUTOMATIQUE , MODE_MANUEL }

5.15.1 Documentation des macros

5.15.1.1 CUSTOM_SETTINGS

```
#define CUSTOM_SETTINGS
```

Dabble bluetooth////.

Définition à la ligne 21 du fichier RobotG7.h.

5.15.1.2 INCLUDE_GAMEPAD_MODULEpowerChannel

```
#define INCLUDE_GAMEPAD_MODULEpowerChannel
```

Définition à la ligne 22 du fichier RobotG7.h.

5.15.1.3 ROBOTG7_FALSE

```
#define ROBOTG7_FALSE 0
```

FIN Dabble bluetooth////.

Définition à la ligne 30 du fichier RobotG7.h.

5.15.1.4 ROBOTG7_TRUE

```
#define ROBOTG7_TRUE 1
```

Définition à la ligne 31 du fichier RobotG7.h.

5.15.2 Documentation des définitions de type

5.15.2.1 MODE

```
typedef enum Navigation MODE
```

Définition à la ligne 40 du fichier RobotG7.h.

5.15.3 Documentation du type de l'énumération

5.15.3.1 Navigation

```
enum Navigation
```

Valeurs énumérées

MODE_START	
MODE_AUTOMATIQUE	
MODE_MANUEL	

Définition à la ligne 34 du fichier RobotG7.h.

5.16 RobotG7.h

Aller à la documentation de ce fichier.

```
00001 /*****************************************************************************************
00002 *****************************************************************************************
00003 ** @author : André MUTOKE MUSULE & Japhet TSHIMANGA BWENDE                              **
00004 ** @date : Mardi 29 Mai 2024                                                           **
00005 ** @name : RobotG7.h                                                                   **
00006 ** @details : Ce fichier implémente les élément composant le robot du système         **
00007 * *****************************************************************************************
00008 * *****************************************************************************************
00009 */
00010
00011
00012
00013
00014 #ifndef ROBOT_GROUPE_7_L4_TLC
00015 #define ROBOT_GROUPE_7_L4_TLC
00016
00017 #pragma once
00018 #include <Arduino.h>
00019
00021 #define CUSTOM_SETTINGS
00022 #define INCLUDE_GAMEPAD_MODULEpowerChannel
00023 #include <DabbleESP32.h>
00025
00026 #include "include/GuideSound/GuideSound.h"
00027 #include "include/MotorControl/MotorControl.h"
00028 #include "include/CtrlWay/CtrlWay.h"
00029
00030 #define ROBOTG7_FALSE 0
00031 #define ROBOTG7_TRUE 1
00032
00033
00034 enum Navigation
00035 {
00036     MODE_START,
00037     MODE_AUTOMATIQUE,
00038     MODE_MANUEL
00039 };
00040 typedef enum Navigation MODE;
00041
00043 class RobotG7
00044 {
00045 public :
00046
00050     RobotG7(MOTOR *roue1 = NULL,
00051             MOTOR *roue2  = NULL,
00052             int pinCtlMode = 5
00053             );
00055     void begin();
00058     uint32_t getDistanceObstacle(){return this->m_gdSound->getDistance();}
00061     int getpwmChannelRoue1(){return this->m_rouesRobots->getRoue1()->pwmChannel;}
00064     int getpwmChannelRoue2(){return this->m_rouesRobots->getRoue2()->pwmChannel;}
00066     void deplacementAvant();
00068     void deplacementArriere();
00070     void tourner_a_GaucheAvant();
00072     void tourner_a_DroiteAvant();
00074     void tourner_a_GaucheArriere();
00076     void tourner_a_DroiteArriere();
00078     void stopRobotG7();
00082     void writeSpeedOnChannel(int vitesseChannel1, int vitesseChannel2);
00083
00084     // ++++++++++++++++++++++++++ Control VITESSE ++++++++++++++++++++++++++++++++
00087     int getVitesseGauche(){return this->m_vitesseGauche;}
00090     int getVitesseDroite(){return this->m_vitesseDroite;}
00093     void setVitesseGauche(int vit){this->m_vitesseGauche = vit;}
00096     void setVitesseDroite(int vit){this->m_vitesseDroite = vit;}
```

```
00097
00098     // ++++++++++++++++++ Control Navigation ++++++++++++++++++
00099
00102     void setNavigation(MODE modeNav){this->m_modeNav = modeNav;}
00103
00104     // ++++++++++++++++ to control movement ++++++++++++++++++++++++++
00105
00108     WAY getSystemNavDirect(){return this->systemNav;}
00109
00111     void navigation();
00112
00115     void setSystemNavDirect(WAY direction){ this->systemNav = direction;}
00116     // ++++++++++++++++++ Control Chemin +++++++++++++++++++++++++++++++
00117
00119     void systemRunningAuto();
00121     void systemRunningManual();
00122 private :
00123
00124     GuideSound *m_gdSound;
00125     MotorControl *m_rouesRobots;
00126
00127     CtrlWay *cheminAprendre;
00128
00129     MODE m_modeNav;
00130
00131     WAY systemNav;
00132
00133     int m_vitesseGauche;
00134     int m_vitesseDroite;
00136     void inputProcess();
00138     void obtenirVitesseJoystick();
00139
00141     void gamePadGestionCommand();
00142     void gamePadAutoGoStart();
00143
00145     void prepareRunningManual(){
00146         if(this->modeManualnotPrepared)
00147         {
00148             Dabble.begin("RobotG7");
00149             this->modeManualnotPrepared = ROBOTG7_FALSE; //
00150         }
00151
00152     }
00153
00154     int modeManualnotPrepared;
00155
00156     int enableCleanning;
00157
00158
00159     // int ctrlMode;
00160
00161     // void runningAutoMode(uint32_t PreviousTime );
00162
00163     // MODE m_modeNCaptureObstacleav;
00164 };
00165
00166 #endif // ROBOT_GROUPE_7_L4_TLC
```

5.17 Référence du fichier /home/andremutoke/Documents/PlatformIO/↩ Projects/ESP32_Project_Robot_Manage/README.md

5.18 Référence du fichier /home/andremutoke/Documents/PlatformIO/↩ Projects/ESP32_Project_Robot_Manage/src/main.cpp

```
#include <Arduino.h>
#include <DabbleESP32.h>
#include <ESP32Servo.h>
#include <RobotG7.h>
```

Macros

— #define INTERVAL_TIME 1000

Le système est implémenté dans ce fichier.

Fonctions

— void setup ()

Fonction de parametrage.

— void loop ()

Fonction du déroulement du système.

Variables

— MOTOR roue1 = {27, 26, 14, PWMCHANNEL}

Definitions des roues.

— MOTOR roue2 = {25, 33, 32, PWMCHANNEL2}

— RobotG7 * myrobot = new RobotG7(&roue1, &roue2)

Instantiation du robot.

— int PIN_MODE_CTRL = 5

— int tempctrl

— unsigned long previouMillis

Gestion du temps.

5.18.1 Documentation des macros

5.18.1.1 INTERVAL_TIME

```
#define INTERVAL_TIME 1000
```

Le système est implémenté dans ce fichier.

Définition à la ligne 23 du fichier main.cpp.

5.18.2 Documentation des fonctions

5.18.2.1 loop()

```
void loop ( )
```

Fonction du déroulement du système.

Définition à la ligne 60 du fichier main.cpp.

5.18.2.2 setup()

```
void setup ( )
```

Fonction de parametrage.

Définition à la ligne 48 du fichier main.cpp.

5.18.3 Documentation des variables

5.18.3.1 myrobot

```
RobotG7* myrobot = new RobotG7(&roue1, &roue2)
```

Instantiation du robot.

Définition à la ligne 39 du fichier main.cpp.

5.18.3.2 PIN_MODE_CTRL

```
int PIN_MODE_CTRL = 5
```

Définition à la ligne 41 du fichier main.cpp.

5.18.3.3 previouMillis

```
unsigned long previouMillis
```

Gestion du temps.

Définition à la ligne 45 du fichier main.cpp.

5.18.3.4 roue1

```
MOTOR roue1 = {27, 26, 14, PWMCHANNEL}
```

Definitions des roues.

Put to PIN 2 the ECHO PIN of U-S Put to PIN 4 the TRIG PIN of U-S

Définition à la ligne 34 du fichier main.cpp.

5.18.3.5 roue2

```
MOTOR roue2 = {25, 33, 32, PWMCHANNEL2}
```

Définition à la ligne 35 du fichier main.cpp.

5.18.3.6 tempctrl

```
int tempctrl
```

Définition à la ligne 42 du fichier main.cpp.

5.19 main.cpp

Aller à la documentation de ce fichier.

```
00001 /*****************************************************************
00002 ******************************************************************
00003 ** @author : André MUTOKE MUSULE & Japhet TSHIMANGA BWENDE       **
00004 ** @date : Mardi 29 Mai 2024                                     **
00005 ** @name : main.cpp                                              **
00006 ** @details : Ce fichier implémente le déroulement du système    **
00007 * ****************************************************************
00008 * ****************************************************************
00009 */
00010
00011
00012 #pragma once
00013
00015
00016 #include <Arduino.h>
00017
00018 #include <DabbleESP32.h>
00019 #include <ESP32Servo.h>
00020
00021 #include <RobotG7.h>
00022
00023 #define INTERVAL_TIME 1000
00024
00025
00026
00034 MOTOR roue1 = {27, 26, 14, PWMCHANNEL};  // Premiere Roue ( cPin1, cPin2, enable)
00035 MOTOR roue2 = {25, 33, 32, PWMCHANNEL2}; //  Seconde Roue
00036
00037
00039 RobotG7 *myrobot = new RobotG7(&roue1, &roue2); // Notre Robot
00040
00041 int PIN_MODE_CTRL = 5;
00042 int tempctrl;
00043
00045 unsigned long previouMillis;
```

```
00046
00048 void setup()
00049 {
00050     tempctrl = 1;
00051     previouMillis = 0;
00052     Serial.begin(115200);
00053     myrobot->begin();
00054
00055     pinMode(PIN_MODE_CTRL, INPUT_PULLUP);
00056
00057 }
00058
00060 void loop()
00061 {
00062     unsigned long currentMillis = millis();
00063     // Serial.print("On veut lire le bp :  ");
00064     int temp = digitalRead(PIN_MODE_CTRL); delay(100);
00065     // Serial.println(temp);
00066
00067
00068     if(temp == 0)
00069     {
00070         delay(1000);
00071
00072         tempctrl++;
00073         if(tempctrl > 3) {tempctrl = 1;}
00074
00075     }
00076
00077     switch (tempctrl)
00078     {
00079     case 1:
00080         myrobot->setNavigation(MODE_START);
00081         break;
00082     case 2:
00083         myrobot->setNavigation(MODE_MANUEL);
00084         break;
00085     case 3:
00086         myrobot->setNavigation(MODE_AUTOMATIQUE);
00087         break;
00088
00089     default:
00090         break;
00091     }
00092
00093     if(currentMillis - previouMillis >= INTERVAL_TIME)
00094     {
00095         Serial.println(myrobot->getDistanceObstacle());
00096         myrobot->navigation();
00097
00098         previouMillis = currentMillis;
00099
00100     }
00101
00102 }
```