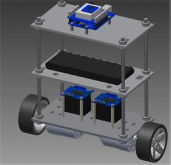
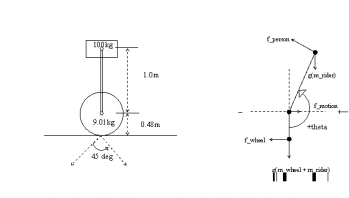
**Projet pratique**

***Présentation :***

Le but du projet est de développer le logiciel de commande d'un robot équilibriste. L'asservissement en équilibre est assuré par un algorithme de commande pilotant la rotation des roues du robot à partir de la connaissance de l'angle du robot avec l'axe vertical. Cet angle est estimé par le biais d'un observateur à partir des mesures issues d'une centrale inertielle (accéléromètre + gyroscope).

Le robot dispose également d'une communication série et d'une IHM via LED.





**Projet pratique**

*L'application :*

*Schéma fonctionnel :*

**Asservissement de l'équilibre :** maintientle robot en équilibre, répète la mêmeséquence en boucle.

* Cycle : Interroger les capteurs Attendre la réponse Exécuter l'algo. De contrôle Appliquer le command
* Répétition : 10ms
* Délai max entre interrogation des capteurs et réception de la donnée : 5ms

**Interface RS232 :** propose un shell de communication à distance, possédant un jeu de 4 commandes.

* Cycle : Attente de l'arrivée d'une commande Prise en compte des commandes/réponse
* Délai max entre requête et envoie de réponse : 50ms
  + **Read** : retourne la dernière valeur de l'angle
  + **Dump** : retourne les 100 dernières valeurs
  + **Stream** : retourne en continu la valeur de l'angle
  + **Help** : affiche un menu d'aide avec les 4 commandes

**Enregistrements** : garder en mémoire les 100 dernières valeurs de l'angle du robot.

**Gestion LED :** clignotement de la LED si angle < 25°, allumage permanent sinon.

- Période de clignotement : 1s ± 50ms (par flash de 100ms)

**Code C**

**La tache StartUART**

La tache StartUART il fait l’initialisation des capteurs et gère l’utilisation du terminal en répondant au différent commande, Read**, Dump, Stream, Help**

**void** **StartUART**(**void** **const** \* argument)

{

/\* USER CODE BEGIN StartDefaultTask \*/

uint8\_t LSM6DS3\_Res = 0;

uint8\_t tempString[100];

uint8\_t message[100];

/\* Init des periphs externes \*/

MotorDriver\_Init();

MESN\_UART\_Init();

**if**(LSM6DS3\_begin(&LSM6DS3\_Res) != *IMU\_SUCCESS*){

MESN\_UART\_PutString\_Poll((uint8\_t\*)"\r\nIMU Error !");

**while**(1);

}

/\* Test des periphs externes \*/

**sprintf**((**char**\*)tempString, "\r\nInit done. LSM6DS3 reg = %02x", LSM6DS3\_Res);

MESN\_UART\_PutString\_Poll(tempString);

MotorDriver\_Move(200);

/\* Test algo autom \*/

**sprintf**((**char**\*)tempString, "\r\nAngle = %ldmDeg", autoAlgo\_angleObs(50,5));

MESN\_UART\_PutString\_Poll(tempString);

osMessagePut ( queue , 8 , 0) ;//apres l'initialisation des capteurs on debloque la tache vTaskAsservissement pour pouvoir utiliser les capteurs

/\* Infinite loop \*/

**for**(;;)

{

MESN\_UART\_GetString(message,osWaitForever);//lecture de commande saisie

**if**(**strcmp**((**char**\*)message,"read")==0){//test est ce que la commande saisie est read

MESN\_UART\_PutString\_Poll((uint8\_t\*)"\r\n \*\*read\*\*");

**sprintf**((**char** \*)message,"\r\n %ld mdeg",ValAngle[indice]);

MESN\_UART\_PutString\_Poll(message);//affichage de la dernier valeur de angle

MESN\_UART\_PutString\_Poll((uint8\_t\*)"\r\n ");

}

**if**(**strcmp**((**char**\*)message,"dump")==0){//test est ce que la commande saisie est dump

MESN\_UART\_PutString\_Poll((uint8\_t\*)"\r\n \*\*dump\*\*");

**int** etat=1,init=indice;

**while**(etat){

**if**(++indice==100)indice=0;//tableau cyclique

**sprintf**((**char** \*)message,"\r\n %ld mdeg",ValAngle[indice]);

MESN\_UART\_PutString\_Poll(message);//affichage de 100 valeur de angles en commancant par la plus ancienne valeur qui se trouve a init + 1 jusqu'a init (tableau cyclique)

MESN\_UART\_GetString(message,0);

**if**(indice==init)etat=0;

}

MESN\_UART\_PutString\_Poll((uint8\_t\*)"\r\n ");

}

**if**(**strcmp**((**char**\*)message,"stream")==0){//test est ce que la commande saisie est stream

MESN\_UART\_PutString\_Poll((uint8\_t\*)"\r\n \*\*stream\*\* \r\n");

**int** etat=1;

osEvent evt ;

**while**(etat){

evt = osMessageGet ( queueAngle , 50 ); // wait for message

**if** ( evt.status == *osEventMessage* ) {

**sprintf**((**char** \*)message,"\r %6ld mdeg",evt.value.v);

MESN\_UART\_PutString\_Poll(message);//affichage de la dernier valeur de angle jusqu'a appui sur un botton

}

**if**(MESN\_UART\_GetString(message,0)==*USER\_OK*)etat=0;//sortie en cas d'appui sur un botton

}

MESN\_UART\_PutString\_Poll((uint8\_t\*)"\r\n ");

}

**if**(**strcmp**((**char**\*)message,"help")==0){//test est ce que la commande saisie est help

MESN\_UART\_PutString\_Poll((uint8\_t\*)"\r\n \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*help\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

MESN\_UART\_PutString\_Poll((uint8\_t\*)"\r\n read : return last measured angle value");

MESN\_UART\_PutString\_Poll((uint8\_t\*)"\r\n dump : return last hundred angle values");

MESN\_UART\_PutString\_Poll((uint8\_t\*)"\r\n stream : continiously return last measured angle value and update display press ENTER to quit stream mode");

MESN\_UART\_PutString\_Poll((uint8\_t\*)"\r\n help : print this menu");

MESN\_UART\_PutString\_Poll((uint8\_t\*)"\r\n \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*end\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

MESN\_UART\_PutString\_Poll((uint8\_t\*)"\r\n ");

}

}

**La tache vTaskAsservissement**

La tache vTaskAsservissement il détermine l’angle du robot et en utilise l’algorithme de l’automatique pour calculer la vitesse de commande du moteur

**void** **vTaskAsservissement**(**void** **const** \* argument){

int32\_t GyroY,AccelX,Angle,Vitesse;

osMessageGet ( queue , osWaitForever );

uint32\_t PrevieusTick = osKernelSysTick() ;

**while**(1){

osDelayUntil(&PrevieusTick,10); // synchronisation de 10 pour l'asservissement

LSM6DS3\_readMgAccelX( &AccelX );//recuperation de la valeur d'acceleration par rapport a x

LSM6DS3\_readMdpsGyroY(&GyroY );//recuperation de la valeur de geroscope par rapport a y

Angle = autoAlgo\_angleObs(AccelX,GyroY);//calcul de l'angle

Vitesse=autoAlgo\_commandLaw (Angle,GyroY);//calcul du commande de l'asservissement

MotorDriver\_Move(Vitesse);//commande le moteur

osMessagePut ( queueAngle,Angle , 0 );//la tache de l'asservissemnt ne doit jamais de bloquer

}

}

**La tache Enregistrement**

La tache Enregistrement il récupère la valeur de nouvelle angle a partir de la queue et l’enregistre dans un tableau globale apres test si l’angle inferieur 25 pour clignoter la LED

**void** **Enregistrement**(**void** **const** \* argument){

int32\_t NewAngle=0;

osEvent evt ;

**while**(1){

evt = osMessageGet ( queueAngle , osWaitForever ); // wait for message

**if** ( evt.status == *osEventMessage* ) {

NewAngle=evt.value.v;

**if**(++indice==100)indice=0;//tableau cyclique

ValAngle[indice]=NewAngle;//enregistrement de la nouvelle valeur de l'angle

**if**(indice==100)indice=0;//tableau cyclique

}

osMessagePut ( queueStreamAngle,NewAngle , 0 );//envoie dans la queue pour la commande stream

**if**(NewAngle<25){

osMutexWait(mutexGPIO, osWaitForever);

HAL\_GPIO\_TogglePin(GPIOA, GPIO\_PIN\_5);//clignotement de la led en cas angle <25

osMutexRelease(mutexGPIO);

}

}

}

***Résultat des commandes***

