RIO Guillaume

Rapport séance 2

Mr Massons nous a été proposé de directement utiliser les chenilles d'un châssis d'un robot déjà constitué et avec des moteurs. Le problème est que les moteurs n'ont pas d'encoder permettant de récupérer notre vitesse. J'ai donc recherche s'il existe ce même moteur mais avec un encoder. Je n'ai rien trouvé de satisfaisant, je vais donc pour le moment ignorer cette manière de calculer la vitesse et on tentera de se débrouiller avec nos capteurs de distance et notre caméra.

J'avais comme première idée de faire une structure qui contiendra les paramètres de mon moteur et une classe de gestion des moteurs qui réalisera les fonctions lié au moteur. J'ai donc réaliser ceci :

La structure :

```
//définition
struct Moteur {
    int pinInteruptMoteur;
    int compteur;
    float mult;
    int vitessseToGo;
    float vitesse;
};
```

La classe :

```
class GestionMoteur
{
  private:

  public:
    static Moteur *moteur;
    static void calculVitesseMoteur();
    static void incrementeCompteurMoteur();
    //GestionMoteur(Moteur *moteurG);
};
```

Les méthode et initialisation :

```
void GestionMoteur::incrementeCompteurMoteur(){
  moteur->compteur +=1 ;
}


void GestionMoteur::calculVitesseMoteur(){
  moteur->vitesse = moteur->mult * moteur->compteur;
  moteur->compteur = 0;

}

struct Moteur moteurG = {
    pinInteruptMoteurG,
    0,
    multG,
    0,
    0
 };
 GestionMoteur gestionMoteur;
```

```
void setup() {
  gestionMoteur.moteur = &moteurG;
  Serial.begin(57600);
  Timer1.initialize(deltaTime*1000);
  Timer1.attachInterrupt(gestionMoteur.calculVitesseMoteur);
  attachInterrupt(pinInteruptMoteurG, gestionMoteur.incrementeCompteurMoteur, RISING);
}

void loop() {
  Serial.print("toto");
}
```

Le problème que j'ai rencontré est que l'interruption est lié à une méthode (ligne 4 et 5 de la fonction setup) or ce n'est pas autorisé.

```
37: undefined reference to `GestionMoteur::moteur'
37: undefined reference to `GestionMoteur::moteur'
```

Je vais donc supprimer ma classe et simplement placer mes fonctions en brut avant ma loop

Définition d'un .h pour ma structure

```
#ifndef H_XX
#define H_XX
/*
 * Struct moteur : est une structure stockant les differents paramètres d'un moteur
 *
 *  int pinInteruptMoteur : valeur correspondant au branchement pin de la carte encoder , 0 pour le pin 2
 *                          de la carte arduino, 1 pour le pin 3
 *  int compteur : compteur correspond au nombre de cran de l'encoder que l'on a compté
 *  float mult : valeur par laquel on va multiplier le compteur pour déterminer la vitesse en tour par
 *               seconde (depuis le dernier deltaTime)
 *  int vitessseToGo : vitesse à laquel on souhaite faire tourner le moteur
 *  float vitesse : vitesse actuel du moteur
 *
 */
struct Moteur {
    int pinInteruptMoteur;
    int compteur;
    float mult;
    int vitessseToGo;
    float vitesse;
};

#endif
```

Code loop

Définition des constantes

```
#include <TimerOne.h>
#include <struc_moteur.h>

//definition constante
const int deltaTime = 20;
const int pinInteruptMoteurG = 0; //corespond au pin 2 de la carte
const float multG =(1/300.0)/(deltaTime*0.001); //mult du moteur gauche

//initialisation du moteur gauche
struct Moteur moteurG = {
    pinInteruptMoteurG,
    0,
    multG,
    0,
    0
 };
```

Définition des fonctions pour les interruptions du moteur

```
/*
 * Focntion de comptage du nombre de crans de l'encoder du moteur gauche,
 * sera lever lors d'une interruption branché sur un pin du l'encoder
 */
void incrementeCompteurMoteurG(){
  moteurG.compteur +=1 ;
}
/*
 * Fonction qui va calculer la vitesse du moteur gauche puis actualiser sa valeur
 */
void calculVitesseMoteurG(){
  moteurG.vitesse = moteurG.mult * moteurG.compteur;
  moteurG.compteur = 0;
}
```

Définition du setup et loop

```
void setup() {
  Serial.begin(57600);

  //initialisation du timer
  //définition de la période
  Timer1.initialize(deltaTime*1000);
  //choix de la fonction à lever
  Timer1.attachInterrupt(calculVitesseMoteurG);

  //initialisation d'un intérruption selon ( un pin, une fonction a lever, mode pour appeler la focntion )
  attachInterrupt(pinInteruptMoteurG, incrementeCompteurMoteurG, RISING);
}

void loop() {

}
```