#define nullptr NULL

#include <ros.h>

#include <sensor\_msgs/Joy.h>

#include <SabertoothSimplified.h>

SabertoothSimplified ST1(Serial1); // Utilise Serial1 pour communiquer avec le premier Sabertooth sur un Arduino Mega

SabertoothSimplified ST2(Serial2); // Utilise Serial2 pour communiquer avec le deuxième Sabertooth sur un Arduino Mega

// Définir la broche de la LED

#define LED 13

// Définir un objet ROS NodeHandle

ros::NodeHandle nh;

// Déclarer la fonction de rappel pour le topic /joy

void joyCallback(const sensor\_msgs::Joy& msg);

// Définir un abonnement au topic /joy

sensor\_msgs::Joy joy\_msg;

ros::Subscriber<sensor\_msgs::Joy> sub("/chatter", &joyCallback);

void setup() {

// Configurer la broche de la LED comme sortie

pinMode(LED, OUTPUT);

// Initialise le port série pour la communication avec le premier Sabertooth

Serial1.begin(9600);

// Initialise le port série pour la communication avec le deuxième Sabertooth

Serial2.begin(9600); // Port série différent du premier Sabertooth

// Initialiser la communication ROS

nh.initNode();

nh.subscribe(sub);

}

void loop() {

nh.spinOnce(); // Traiter les messages ROS

// Votre code principal ici

}

// Définir la fonction de rappel pour le topic /joy

void joyCallback(const sensor\_msgs::Joy& msg) {

// Obtenez les valeurs des axes et des boutons depuis le message ROS

float xAxis = msg.axes[0];

float yAxis = msg.axes[1];

float xAxis2 = msg.axes[2]; // Axes du deuxième joystick

float yAxis2 = msg.axes[3]; // Axes du deuxième joystick

int buttonA = msg.buttons[0]; // Par exemple, si le bouton A est à l'indice 0

Serial.println("\nAxis 1: ");

Serial.println(xAxis);

Serial.println("\nAxis 2: ");

Serial.println(yAxis);

Serial.println("\nAxis 3: ");

Serial.println(xAxis2);

Serial.println("\nAxis 4: ");

Serial.println(yAxis2);

Serial.println("\nButton A: ");

Serial.println(buttonA);

// Allumer la LED en fonction de l'état du bouton A

if (buttonA == 1) {

digitalWrite(LED, HIGH);

}

else {

digitalWrite(LED, LOW);

}

// Utiliser les valeurs des axes du premier joystick pour contrôler le premier Sabertooth

float motor1Speed = map(yAxis, -1.0, 1.0, -127, 127);

float motor2Speed = map(xAxis, -1.0, 1.0, -127, 127);

// Calculer la vitesse et la direction des moteurs du premier Sabertooth

int m1 = motor1Speed + motor2Speed;

int m2 = motor1Speed - motor2Speed;

// Limiter les valeurs entre -127 et 127

m1 = constrain(m1, -127, 127);

m2 = constrain(m2, -127, 127);

// Envoyer les commandes au premier Sabertooth

ST1.motor(1, m1);

ST1.motor(2, m2);

// Utiliser les valeurs des axes du deuxième joystick pour contrôler le deuxième Sabertooth

float motor3Speed = map(xAxis2, -1.0, 1.0, -127, 127);

float motor4Speed = map(yAxis2, -1.0, 1.0, -127, 127);

// Calculer la vitesse et la direction des moteurs du deuxième Sabertooth

int m3 = motor3Speed + motor4Speed;

int m4 = motor3Speed - motor4Speed;

// Limiter les valeurs entre -127 et 127

m3 = constrain(m3, -127, 127);

m4 = constrain(m4, -127, 127);

// Envoyer les commandes au deuxième Sabertooth

ST2.motor(1, m3);

ST2.motor(2, m4);

}