#if defined(\_\_linux\_\_) || defined(\_\_APPLE\_\_)

#include <fcntl.h>

#include <termios.h>

#include <unistd.h>

#define STDIN\_FILENO 0

#endif

#include <stdlib.h>

#include <stdio.h>

#include "ros/ros.h"

#include "dynamixel\_sdk.h" // Uses Dynamixel SDK library

#include "ros\_brand/Message1.h"

// Control table address

#define ADDR\_MX\_TORQUE\_ENABLE 24 // Control table address is different in Dynamixel model

#define ADDR\_MOVING\_SPEED 32

#define ADDR\_CCW\_LIMIT\_ANGLE 8

// Protocol version

#define PROTOCOL\_VERSION 1.0 // See which protocol version is used in the Dynamixel

// Default setting

#define DXL1\_ID 1 // Dynamixel ID: 1

#define DXL2\_ID 2 // Dynamixel ID: 2

#define DXL3\_ID 3 // Dynamixel#2 ID: 3

#define DXL4\_ID 4 // Dynamixel#2 ID: 4

#define BAUDRATE 1000000

#define DEVICENAME "/dev/ttyUSB0" // Check which port is being used on your controller

#define TORQUE\_ENABLE 1 // Value for enabling the torque

#define TORQUE\_DISABLE 0 // Value for disabling the torque

#define DXL\_MINIMUM\_POSITION\_VALUE 100 // Dynamixel will rotate between this value

#define DXL\_MAXIMUM\_POSITION\_VALUE 4000 // and this value (note that the Dynamixel would not move when the position value is out of movable range. Check e-manual about the range of the Dynamixel you use.)

#define DXL\_MOVING\_STATUS\_THRESHOLD 10 // Dynamixel moving status threshold

#define ESC\_ASCII\_VALUE 0x1b

int getch()

{

#if defined(\_\_linux\_\_) || defined(\_\_APPLE\_\_)

 struct termios oldt, newt;

 int ch;

 tcgetattr(STDIN\_FILENO, &oldt);

 newt = oldt;

 newt.c\_lflag &= ~(ICANON | ECHO);

 tcsetattr(STDIN\_FILENO, TCSANOW, &newt);

 ch = getchar();

 tcsetattr(STDIN\_FILENO, TCSANOW, &oldt);

 return ch;

#endif

}

int kbhit(void)

{

#if defined(\_\_linux\_\_) || defined(\_\_APPLE\_\_)

 struct termios oldt, newt;

 int ch;

 int oldf;

 tcgetattr(STDIN\_FILENO, &oldt);

 newt = oldt;

 newt.c\_lflag &= ~(ICANON | ECHO);

 tcsetattr(STDIN\_FILENO, TCSANOW, &newt);

 oldf = fcntl(STDIN\_FILENO, F\_GETFL, 0);

 fcntl(STDIN\_FILENO, F\_SETFL, oldf | O\_NONBLOCK);

 ch = getchar();

 tcsetattr(STDIN\_FILENO, TCSANOW, &oldt);

 fcntl(STDIN\_FILENO, F\_SETFL, oldf);

 if (ch != EOF)

 {

 ungetc(ch, stdin);

 return 1;

 }

 return 0;

#endif

}

int x\_posit;

int y\_posit; //make global varialbe

void msgCallback(const ros\_brand::Message1::ConstPtr &msg) //message callback function

{

 x\_posit = msg->x\_coordinate; //pixel x coordinate

 y\_posit = msg->y\_coordinate; //pixel y coordinate

}

int main(int argc, char \*\*argv)

{

 ros::init(argc, argv, "subscriber\_node");

 ros::NodeHandle nh;

 ros::Subscriber ros\_brand\_sub = nh.subscribe("ros\_brand\_msg", 100, msgCallback);

 // Initialize PortHandler instance

 // Set the port path

 // Get methods and members of PortHandlerLinux or PortHandlerWindows

 dynamixel::PortHandler \*portHandler = dynamixel::PortHandler::getPortHandler(DEVICENAME);

 // Initialize PacketHandler instance

 // Set the protocol version

 // Get methods and members of Protocol1PacketHandler or Protocol2PacketHandler

 dynamixel::PacketHandler \*packetHandler = dynamixel::PacketHandler::getPacketHandler(PROTOCOL\_VERSION);

 int dxl\_comm\_result = COMM\_TX\_FAIL; // Communication result

 int spd=150 ;

 uint8\_t dxl\_error = 0; // Dynamixel error

 // Open port

 if (portHandler->openPort())

 {

 printf("Succeeded to open the port!\n");

 }

 else

 {

 printf("Failed to open the port!\n");

 printf("Press any key to terminate...\n");

 getch();

 return 0;

 }

 // Set port baudrate

 if (portHandler->setBaudRate(BAUDRATE))

 {

 printf("Succeeded to change the baudrate!\n");

 }

 else

 {

 printf("Failed to change the baudrate!\n");

 printf("Press any key to terminate...\n");

 getch();

 return 0;

 }

// Enable Dynamixel#1 Torque

 dxl\_comm\_result = packetHandler->write1ByteTxRx(portHandler, DXL1\_ID, ADDR\_MX\_TORQUE\_ENABLE, TORQUE\_ENABLE, &dxl\_error);

 if (dxl\_comm\_result != COMM\_SUCCESS)

 {

 printf("%s\n", packetHandler->getTxRxResult(dxl\_comm\_result));

 }

 else if (dxl\_error != 0)

 {

 printf("%s\n", packetHandler->getRxPacketError(dxl\_error));

 }

 else

 {

 printf("Dynamixel#%d has been successfully connected \n", DXL1\_ID);

 }

 // Enable Dynamixel#2 Torque

 dxl\_comm\_result = packetHandler->write1ByteTxRx(portHandler, DXL2\_ID, ADDR\_MX\_TORQUE\_ENABLE, TORQUE\_ENABLE, &dxl\_error);

 if (dxl\_comm\_result != COMM\_SUCCESS)

 {

 printf("%s\n", packetHandler->getTxRxResult(dxl\_comm\_result));

 }

 else if (dxl\_error != 0)

 {

 printf("%s\n", packetHandler->getRxPacketError(dxl\_error));

 }

 else

 {

 printf("Dynamixel#%d has been successfully connected \n", DXL2\_ID);

 }

 // Enable Dynamixel#3 Torque

 dxl\_comm\_result = packetHandler->write1ByteTxRx(portHandler, DXL3\_ID, ADDR\_MX\_TORQUE\_ENABLE, TORQUE\_ENABLE, &dxl\_error);

 if (dxl\_comm\_result != COMM\_SUCCESS)

 {

 printf("%s\n", packetHandler->getTxRxResult(dxl\_comm\_result));

 }

 else if (dxl\_error != 0)

 {

 printf("%s\n", packetHandler->getRxPacketError(dxl\_error));

 }

 else

 {

 printf("Dynamixel#%d has been successfully connected \n", DXL3\_ID);

 }

 // Enable Dynamixel#4 Torque

 dxl\_comm\_result = packetHandler->write1ByteTxRx(portHandler, DXL4\_ID, ADDR\_MX\_TORQUE\_ENABLE, TORQUE\_ENABLE, &dxl\_error);

 if (dxl\_comm\_result != COMM\_SUCCESS)

 {

 printf("%s\n", packetHandler->getTxRxResult(dxl\_comm\_result));

 }

 else if (dxl\_error != 0)

 {

 printf("%s\n", packetHandler->getRxPacketError(dxl\_error));

 }

 else

 {

 printf("Dynamixel#%d has been successfully connected \n", DXL4\_ID);

 }

 //Convert Dynamixel into Wheelmode -> CW and CCW limit to value 0

 //CW initial value is already 0

 dxl\_comm\_result = packetHandler->write2ByteTxRx(portHandler, DXL1\_ID, ADDR\_CCW\_LIMIT\_ANGLE, 0, &dxl\_error);

 dxl\_comm\_result = packetHandler->write2ByteTxRx(portHandler, DXL2\_ID, ADDR\_CCW\_LIMIT\_ANGLE, 0, &dxl\_error);

 dxl\_comm\_result = packetHandler->write1ByteTxRx(portHandler, DXL3\_ID, ADDR\_CCW\_LIMIT\_ANGLE, 0, &dxl\_error);

 dxl\_comm\_result = packetHandler->write1ByteTxRx(portHandler, DXL4\_ID, ADDR\_CCW\_LIMIT\_ANGLE, 0, &dxl\_error);

 ros::Rate rate(15.0); //set the frequency

 while (ros::ok())

 {

 int dist\_x;

 int dist\_y;

 int goalposition\_x = 320;

 int goalposition\_y = 240;

 //move to x axis

 if (x\_posit > goalposition\_x) // ceter of x is 320

 {

 dxl\_comm\_result = packetHandler->write2ByteTxRx(portHandler, DXL1\_ID, ADDR\_MOVING\_SPEED, x\_posit-goalposition\_x , &dxl\_error);//ccw rota\tion

 dxl\_comm\_result = packetHandler->write2ByteTxRx(portHandler, DXL2\_ID, ADDR\_MOVING\_SPEED, x\_posit-goalposition\_x + 1024 , &dxl\_error);//cw rota\tion

 // printf("speed : %d", x\_posit-157);

 }

 else if(x\_posit < 0)

 {

 dxl\_comm\_result = packetHandler->write2ByteTxRx(portHandler, DXL1\_ID, ADDR\_MOVING\_SPEED, 0 , &dxl\_error);//ccw rota\tion

 dxl\_comm\_result = packetHandler->write2ByteTxRx(portHandler, DXL2\_ID, ADDR\_MOVING\_SPEED, 0 , &dxl\_error);//cw rota\tion

 // printf("speed : %d", x\_posit-157);

 }

 else

 {

 dxl\_comm\_result = packetHandler->write2ByteTxRx(portHandler, DXL1\_ID, ADDR\_MOVING\_SPEED, goalposition\_x-x\_posit + 1024, &dxl\_error);//cw rota\tion

 dxl\_comm\_result = packetHandler->write2ByteTxRx(portHandler, DXL2\_ID, ADDR\_MOVING\_SPEED, goalposition\_x-x\_posit , &dxl\_error);//ccw rota\tion

 //printf( "speed : %d", 157-x\_posit);

 }

 //move to y axis

 if ( y\_posit > goalposition\_y)

 {

 dxl\_comm\_result = packetHandler->write2ByteTxRx(portHandler, DXL3\_ID, ADDR\_MOVING\_SPEED, y\_posit-goalposition\_y , &dxl\_error);//ccw rota\tion

 }

 else if(y\_posit<0)

 {

 dxl\_comm\_result = packetHandler->write2ByteTxRx(portHandler, DXL3\_ID, ADDR\_MOVING\_SPEED, 0 , &dxl\_error);//ccw rota\tion

 }

 else

 {

 dxl\_comm\_result = packetHandler->write2ByteTxRx(portHandler, DXL3\_ID, ADDR\_MOVING\_SPEED, goalposition\_y-y\_posit + 1024 , &dxl\_error);//cw rota\tion

 }

 ros::spinOnce(); //run the msgcallback when it received topic

 //rate.sleep();

 }

// Disable Dynamixel#1 Torque

 dxl\_comm\_result = packetHandler->write1ByteTxRx(portHandler, DXL1\_ID, ADDR\_MX\_TORQUE\_ENABLE, TORQUE\_DISABLE, &dxl\_error);

 if (dxl\_comm\_result != COMM\_SUCCESS)

 {

 printf("%s\n", packetHandler->getTxRxResult(dxl\_comm\_result));

 }

 else if (dxl\_error != 0)

 {

 printf("%s\n", packetHandler->getRxPacketError(dxl\_error));

 }

 // Disable Dynamixel#2 Torque

 dxl\_comm\_result = packetHandler->write1ByteTxRx(portHandler, DXL2\_ID, ADDR\_MX\_TORQUE\_ENABLE, TORQUE\_DISABLE, &dxl\_error);

 if (dxl\_comm\_result != COMM\_SUCCESS)

 {

 printf("%s\n", packetHandler->getTxRxResult(dxl\_comm\_result));

 }

 else if (dxl\_error != 0)

 {

 printf("%s\n", packetHandler->getRxPacketError(dxl\_error));

 }

 // Disable Dynamixel#3 Torque

 dxl\_comm\_result = packetHandler->write1ByteTxRx(portHandler, DXL3\_ID, ADDR\_MX\_TORQUE\_ENABLE, TORQUE\_DISABLE, &dxl\_error);

 if (dxl\_comm\_result != COMM\_SUCCESS)

 {

 printf("%s\n", packetHandler->getTxRxResult(dxl\_comm\_result));

 }

 else if (dxl\_error != 0)

 {

 printf("%s\n", packetHandler->getRxPacketError(dxl\_error));

 }

 // Disable Dynamixel#4 Torque

 dxl\_comm\_result = packetHandler->write1ByteTxRx(portHandler, DXL4\_ID, ADDR\_MX\_TORQUE\_ENABLE, TORQUE\_DISABLE, &dxl\_error);

 if (dxl\_comm\_result != COMM\_SUCCESS)

 {

 printf("%s\n", packetHandler->getTxRxResult(dxl\_comm\_result));

 }

 else if (dxl\_error != 0)

 {

 printf("%s\n", packetHandler->getRxPacketError(dxl\_error));

 }

 return 0;

}