```
/*
 * Author: Tiange  Date: 18/08/22  Version:1.0
 * This code is used for controlling the servoes
 * It receive the instruction about the speed of servo from Raspberry PI
 * It send back the messages about status of servo to Raspberry PI, including the mode(TELEOP, CONT
ROLLING, DEADMAN) and the speed.
 *
 */
#include <Servo.h>
#include <ros.h>
#include <std_msgs/String.h>
#include <std_msgs/Bool.h>
#include <std_msgs/Int8.h>
#include <std_msgs/UInt8MultiArray.h>
#include <std_srvs/SetBool.h>


using std_srvs::SetBool;
#define forwardMax 1100
#define backwardMax 1900
#define Stop 1500
#define TeleMode 1250
#define ControlMode 1700
#define MinOfStop 1350
#define MaxOfStop 1650
/*
 * Callback functions declaration
 */

bool EnableCallback(const SetBool::Request &enable_rqst, SetBool::Request &enable_rsp);
void servoStateCallback(const std_msgs::UInt8MultiArray& msg);


/*
 * There are two messages published and two messages subscribed
 * driveStatus message is published telling PI about the current mode(TELEOP, CONTROLLING, DEAD
MAN)
 * speedStatus message is published telling PI about the current speeds of servos, first is right servo, the
n left
 * enable message is subscribed to start Arduino
 * servoState message is subscribed to get the speed of the servoes,including two numbers ranging from
* 0 to 180:
 * the first number is for the right,and the second number is for the left;
```

```
 *  -100 means maximal forward speed, 0 means stop, +100 means maximal backward speed
 */
std_msgs::String str_msg;
std_msgs::Int8 servoSpeedR;
std_msgs::Int8 servoSpeedL;
ros::Publisher driveStatus("driveStatus", &str_msg);
ros::Publisher speedStatusR("speedStatusR", &servoSpeedR);
ros::Publisher speedStatusL("speedStatusL", &servoSpeedL);
ros::ServiceServer<SetBool::Request, SetBool::Response> enable("arduinoEnable",&EnableCallback);
ros::Subscriber<std_msgs::UInt8MultiArray> servoState("servoState",&servoStateCallback);
ros::NodeHandle  nh;


/*
 * Servo controling related parameter declaration
 * MyservoR is the servo on the right side, MyservoL is the servo on the left side
 * ServoInR is the pin number of the right servo, ServoInL is the pin number of the left servo
 * pwmvalue1 is used to record the PWM value from right-left control channel
 * pwmvalue2 is used to record the PWM value from forward-backward control channel
 * SpeedR is the PWM value of the right servo, SpeedL is for the left
 * PWMIn_RL is the pin number of right-left control channel, PWMIN_FB is for forward-backward
* control, PWMIN_MODE is for mode switch channel
 * Startarduino is uesd to strat the loop()
 */
Servo MyservoR;
Servo MyservoL;
const int ServoInR = 9;
const int ServoInL = 10;
int pwmvalue1 = 0;
int pwmvalue2 = 0;
const int PWMIn_RL = 3;
const int PWMIn_FB = 5;
const int PWMIn_MODE = 6;
int SpeedR = 90;
int SpeedL = 90;

bool Startarduino = false;
/*
 * mode status message
 */
char control[] = "remote-control";
char deadman[] = "deadman";
char teleop[] = "teleoperate";
/*
```

```
 * Callback function
 */
bool EnableCallback(const SetBool::Request &enable_rqst, SetBool::Request &enable_rsp)
{
  Startarduino = enable_rqst.data;
  return true;
}

void servoStateCallback(const std_msgs::UInt8MultiArray& msg)
{
  MyservoR.writeMicroseconds(map(msg.data[0],-100,100,forwardMax,backwardMax));
  MyservoL.writeMicroseconds(map(msg.data[1],-100,100,forwardMax,backwardMax));
}

void doArduino();

void setup()
{
  nh.initNode();
  nh.advertise(driveStatus);
  nh.advertise(speedStatusR);
  nh.advertise(speedStatusL);
  nh.advertiseService(enable);
  nh.subscribe(servoState);
  MyservoR.attach(ServoInR);
  MyservoL.attach(ServoInL);
  pinMode(PWMIn_RL,INPUT);
  pinMode(PWMIn_FB,INPUT);
  pinMode(PWMIn_MODE,INPUT);
  pinMode(ServoInR,OUTPUT);
  pinMode(ServoInL,OUTPUT);
}

void loop()
{
  nh.spinOnce();
  delay(1000);
  if(Startarduino)
    doArduino();
}

void doArduino()
{
  /*******************************remote control Mode*********************************/
```

```
if( pulseIn(PWMIn_MODE,HIGH) > ControlMode)
{
  /**********right-left control**********/
  pwmvalue1 = pulseIn(PWMIn_RL,HIGH);
  if(pwmvalue1 < MinOfStop)          /*turn right*/
  {
    SpeedR = forwardMax;
    SpeedL = Stop;
    MyservoR.writeMicroseconds(SpeedR);
    MyservoL.writeMicroseconds(SpeedL);
  }else if(pwmvalue1 > MaxOfStop)   /*turn left*/
  {
    SpeedL = forwardMax;
    SpeedR = Stop;
    MyservoR.writeMicroseconds(SpeedR);
    MyservoL.writeMicroseconds(SpeedL);
  }
  /********forward-back control**********/
  pwmvalue2 = pulseIn(PWMIn_FB,HIGH);
  if(pwmvalue2 < MinOfStop)          /*forward*/
  {
    SpeedR = pwmvalue2;
    SpeedL = pwmvalue2;
    MyservoR.writeMicroseconds(SpeedR);
    MyservoL.writeMicroseconds(SpeedL);
  }else if(pwmvalue2 > MaxOfStop)   /*backward*/
  {
    SpeedR = pwmvalue2;
    SpeedL = pwmvalue2;
    MyservoR.writeMicroseconds(SpeedR);
    MyservoL.writeMicroseconds(SpeedL);
  }else                              /*stop*/
  {
    if(pwmvalue1 > MinOfStop && pwmvalue1 <MaxOfStop)
    {
      SpeedR = Stop;
      SpeedL = Stop;
      MyservoR.writeMicroseconds(SpeedR);
      MyservoL.writeMicroseconds(SpeedL);
    }
  }
  /******publish driveStatus message*****/
  str_msg.data = control;
  driveStatus.publish (&str_msg);
```

```
    servoSpeedR.data = map(SpeedR,forwardMax,backwardMax,-100,100);
    speedStatusR.publish (&servoSpeedR);
    servoSpeedL.data = map(SpeedL,forwardMax,backwardMax,-100,100);
    speedStatusL.publish (&servoSpeedL);
    nh.spinOnce();
    delay(1000);
  }

  /**********************************teleop Mode**********************************/
  else if( pulseIn(PWMIn_MODE,HIGH) <TeleMode && pulseIn(PWMIn_MODE,HIGH) !=0)
  {
    /******publish driveStatus message******/
    str_msg.data = teleop;
    driveStatus.publish (&str_msg);
    nh.spinOnce();
    delay(1000);
  }

  /********************************* deadmanMode ************************************/
  else
  {
    /******publish driveStatus message******/
    str_msg.data = deadman;
    driveStatus.publish (&str_msg);
    nh.spinOnce();
    delay(1000);
  }
}
```