Coordinate control

Coordinate control is to make the robot arm move to a specified point with a specified posture, which is divided into x, y, z, rx, ry, rz. X, Y and Z represents the position of the robot arm head in space (The coordinate system is Cartesian coordinate system); [rx,ry,rz] represents the posture of such head at this point (The coordinate system is Euler coordinates).

1 Single parameter coordinate

1.1 Sending single parameter coordinates

SendOneCoord(int coord, int value, int speed)
Return value: none
Parameter description: Parameter 1: coordinate number(1-6(x, y, z, rx, ry, rz)); Parameter 2: coordinate (X, Y and Z range from -300 to 300.00mm; RX, RY and RZ range from -180 to 180); Parameter 3: speed (0-100).
Case:

```
mc.SendOneCoord(1, 160, 30);
```

2 Multiple parameter coordinates

2.1 Getting all coordinates

GetCoords()
Return value: return an array of int type, int[], length: 6
Description of parameters: none
Case:

```
var recv = mc.GetCoords();
```

2.2 Sending single parameter coordinates

SendCoords(int[] coords, int speed, int mode)
Return value: none
Parameter description: Parameter 1: all coordinates (X, Y and Z range from -300 to 300.00mm; RX, RY and RZ range from -180 to 180); Parameter 2: speed (0-100); Parameter 3: mode (0 - angular, and1 - linear)
Case:

```
int[] coords = new[] {160, 160, 160, 0, 0, 0};
mc.SendCoords(coords ,30);
int[] coords = new[] {160, 160, 160, 0, 0, 0};
mc.SendCoords(coords,30);
```

3 Complete use cases

The program.cs in the project is a complete use case program, which can be modified as needed on this basis.
using System;

```
namespace Mycobot.csharp
{
    class Test
    {
        static void Main(string[] args)
        {
            MyCobot mc = new MyCobot("/dev/ttyUSB0");
            mc.Open();
            // int[] angles = new[] {100, 100, 100, 100, 100, 100};
            // mc.SendAngles(angles, 50);
            // Thread.Sleep(5000);
            // var recv = mc.GetAngles();
            // foreach (var v in recv)
            // {
                // Console.WriteLine(v);
            // }

            // int[] coords = new[] {160, 160, 160, 0, 0, 0};
            // mc.SendCoords(coords, 90, 1);
            // Thread.Sleep(5000);
            // var recv = mc.GetCoords();
            // foreach (var v in recv)
            // {
                // Console.WriteLine(v);
            // }

            mc.SendOneAngle(1, 100,70);

            // byte[] setColor = {0xfe, 0xfe, 0x05, 0x6a, 0xff, 0x00, 0x00,
0xfa};
            mc.Close();
        }
    }
}
```