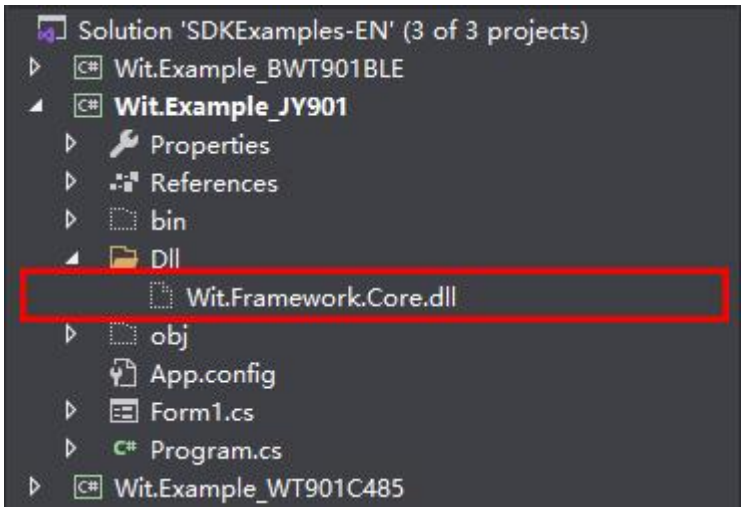
WitMotion Protocol SDK Quick Guide

WlitMotion Protocol: referred to as WitMotion protocol, it is the protocol used by WitMotion sensors, usually TTL or 232 level sensors use WitMotion protocol; the protocol stipulates that the sensor returns data packets starting with 55, and the pc software sends FF AA. Data packets beginning with AA;

This routine introduces how to use C# to develop the host computer to connect the WitMotion protocol sensor, receive sensor data, and communicate with the sensor;
Please read the relevant sensor manual before viewing this routine to understand the protocol used by the sensor and the basic functions of the sensor.

Routine directory

The routine project directory is as follows
Dll: The dependency files of the project, please import these dlls into your project before running the project.
Form1: There is only one Form window in the routine, all logic codes are in the Form window file, and there are no other files



Turn on the device

The JY901 object represents the JY901 device in the program, and you can communicate with the device through it; when opening the device, you need to specify the serial port number and baud rate of the sensor, and call the JY901.Open() method after specifying it.

```
/// <summary>
/// Turn on the device
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void openButton_Click(object sender, EventArgs e)
{
    // Get the serial number and baud rate of the connection
    string portName;
    int baudrate;
    try
    {
        portName = (string)portComboBox.SelectedItem;
        baudrate = (int)baudComboBox.SelectedItem;
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
        return;
    }

    // Do not open repeatedly
    if (JY901.IsOpen())
    {
        return;
    }

    // Turn on the device
    try
    {
        JY901.SetPortName(portName);
        JY901.SetBaudrate(baudrate);
        JY901.Open();
        // Implement logging data events
        JY901.OnRecord += JY901_OnRecord;
```

```
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
            return;
        }
}
```

Turn off the device

Close the device and call the JY901.Close() method.

```
/// <summary>
/// Turn off the device
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void closeButton_Click(object sender, EventArgs e)
{
        try
        {
            // Turn off the device if it is already on
            if (JY901.IsOpen())
            {
                JY901.OnRecord -= JY901_OnRecord;
                JY901.Close();
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
            return;
        }
}
```

Receive sensor data

GET DATA

The JY901 object will automatically solve the sensor data and save it on itself. The sensor data can be obtained through the JY901.GetDeviceData() method. JY901.GetDeviceData() needs to pass in a key to get sensor data. Please check the key to be used in the routine, the keys are all stored in the WitSensorKey class

```
/// <summary>
/// Get device data
/// </summary>
private string GetDeviceData(JY901 JY901)
{
    StringBuilder builder = new StringBuilder();
    builder.Append(JY901.GetDeviceName()).Append("\n");
    // acceleration

    builder.Append("AccX").Append(":").Append(JY901.GetDeviceData(WitSensorKey.AccX)).
    Append("g \t");

    builder.Append("AccY").Append(":").Append(JY901.GetDeviceData(WitSensorKey.AccY)).
    Append("g \t");

    builder.Append("AccZ").Append(":").Append(JY901.GetDeviceData(WitSensorKey.AccZ)).
    Append("g \n");
    // Angular velocity

    builder.Append("GyroX").Append(":").Append(JY901.GetDeviceData(WitSensorKey.AsX)).
    Append("°/s \t");

    builder.Append("GyroY").Append(":").Append(JY901.GetDeviceData(WitSensorKey.AsY)).
    Append("°/s \t");

    builder.Append("GyroZ").Append(":").Append(JY901.GetDeviceData(WitSensorKey.AsZ)).
    Append("°/s \n");
    // angle

    builder.Append("AngleX").Append(":").Append(JY901.GetDeviceData(WitSensorKey.Ang
    leX)).Append("°  \t");

    builder.Append("AngleY").Append(":").Append(JY901.GetDeviceData(WitSensorKey.Angl
    eY)).Append("°  \t");
```

```
builder.Append("AngleZ").Append(":").Append(JY901.GetDeviceData(WitSensorKey.AngleZ)).Append("°  \n");
// magnetic field

builder.Append("MagX").Append(":").Append(JY901.GetDeviceData(WitSensorKey.HX)).Append("uT \t");

builder.Append("MagY").Append(":").Append(JY901.GetDeviceData(WitSensorKey.HY)).Append("uT \t");

builder.Append("MagZ").Append(":").Append(JY901.GetDeviceData(WitSensorKey.HZ)).Append("uT \n");
// latitude and longitude

builder.Append("Lon").Append(":").Append(JY901.GetDeviceData(WitSensorKey.Lon)).Append("′  \t");

builder.Append("Lat").Append(":").Append(JY901.GetDeviceData(WitSensorKey.Lat)).Append("′  \n");
// port number

builder.Append("D0").Append(":").Append(JY901.GetDeviceData(WitSensorKey.D0)).Append("\t");

builder.Append("D1").Append(":").Append(JY901.GetDeviceData(WitSensorKey.D1)).Append("\t");

builder.Append("D2").Append(":").Append(JY901.GetDeviceData(WitSensorKey.D2)).Append("\t");

builder.Append("D3").Append(":").Append(JY901.GetDeviceData(WitSensorKey.D3)).Append("\n");
// Quaternion

builder.Append("Q0").Append(":").Append(JY901.GetDeviceData(WitSensorKey.Q0)).Append("\t");

builder.Append("Q1").Append(":").Append(JY901.GetDeviceData(WitSensorKey.Q1)).Append("\t");

builder.Append("Q2").Append(":").Append(JY901.GetDeviceData(WitSensorKey.Q2)).Append("\t");
```

```
        builder.Append("Q3").Append(":").Append(JY901.GetDeviceData(WitSensorKey.Q3)).Ap
        pend("\n");
        // air pressure

        builder.Append("P").Append(":").Append(JY901.GetDeviceData(WitSensorKey.Q1)).Appe
        nd("Pa \t");

        builder.Append("H").Append(":").Append(JY901.GetDeviceData(WitSensorKey.Q2)).App
        end("m \t");
        // temperature

        builder.Append("T").Append(":").Append(JY901.GetDeviceData(WitSensorKey.T)).Appen
        d("℃  \n");
        // GPS

        builder.Append("GPSHeight").Append(":").Append(JY901.GetDeviceData(WitSensorKey.
        GPSHeight)).Append(" m \t");

        builder.Append("GPSYaw").Append(":").Append(JY901.GetDeviceData(WitSensorKey.GP
        SYaw)).Append("°  \t");

        builder.Append("GPSV").Append(":").Append(JY901.GetDeviceData(WitSensorKey.GPSV))
        .Append("km/h \n");
        // positioning accuracy

        builder.Append("PDOP").Append(":").Append(JY901.GetDeviceData(WitSensorKey.PDO
        P)).Append("\t");

        builder.Append("VDOP").Append(":").Append(JY901.GetDeviceData(WitSensorKey.VDO
        P)).Append("\t");

        builder.Append("HDOP").Append(":").Append(JY901.GetDeviceData(WitSensorKey.HDO
        P)).Append("\n");
        // version number

        builder.Append("VersionNumber").Append(":").Append(JY901.GetDeviceData(WitSensor
        Key.VersionNumber)).Append("\n");
        return builder.ToString();
    }
```

RECORD DATA

The data of the sensor can be obtained through the JY901 object, but usually the host computer needs to record the data of the sensor. JY901 has an OnRecord event that will notify you when the data should be recorded, and the OnRecord event can be realized when the device is turned on;) method to record data

```
/// <summary>
/// Turn on the device
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void openButton_Click(object sender, EventArgs e)
{
    // Get the serial number and baud rate when connect
    string portName;
    int baudrate;
    try
    {
        portName = (string)portComboBox.SelectedItem;
        baudrate = (int)baudComboBox.SelectedItem;
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
        return;
    }

    // Do not open again
    if (JY901.IsOpen())
    {
        return;
    }

    // Turn on the device
    try
    {
        JY901.SetPortName(portName);
        JY901.SetBaudrate(baudrate);
        JY901.Open();
        // Implement logging data events
        JY901.OnRecord += JY901_OnRecord;
    }
    catch (Exception ex)
    {
```

```
            MessageBox.Show(ex.Message);
            return;
        }
}

/// <summary>
/// This is called when sensor data is refreshed and you can log data here
/// </summary>
/// <param name="jY901"></param>
private void JY901_OnRecord(JY901 jY901)
{
        string text = GetDeviceData(JY901);
        Debug.WriteLine(text);
}
```

Set up the sensor

The sensor can be operated by the method of JY901
JY901.UnlockReg() Send unlock register command
JY901.AppliedCalibration() Sends the addition calibration command
JY901.StartFieldCalibration() Send start field calibration command
JY901.EndFieldCalibration() Send end field calibration command
JY901.SetReturnRate() Send the command to set the return rate
JY901.SetBandWidth() Send command to set bandwidth
JY901.SendProtocolData() Send other commands

ACCELERATION CALIBRATION

Acceleration calibration of the sensor by calling the JY901.AppliedCalibration() method

```
/// <summary>
/// Acceleration calibration
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void appliedCalibrationButton_Click(object sender, EventArgs e)
{
        if (JY901.IsOpen() == false)
        {
            return;
        }
```

```
    try
    {
        // Unlock registers and send commands
        JY901.UnlockReg();
        JY901.AppliedCalibration();
        // The following two lines are equivalent to the above, and it is recommended to
        use the above
        //JY901.SendProtocolData(new byte[] { 0xff, 0xaa, 0x69, 0x88, 0xb5 });
        //JY901.SendProtocolData(new byte[] { 0xff, 0xaa, 0x01, 0x01, 0x00 });
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```

MAGNETIC FIELD CALIBRATION

Perform magnetic field calibration on the sensor by calling the JY901.StartFieldCalibration() method and the JY901.EndFieldCalibration() method

```
/// <summary>
/// Start magnetic field calibration
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void startFieldCalibrationButton_Click(object sender, EventArgs e)
{
    if (JY901.IsOpen() == false)
    {
        return;
    }
    try
    {
        // Unlock registers and send commands
        JY901.UnlockReg();
        JY901.StartFieldCalibration();
        // The following two lines are equivalent to the above, and it is recommended to
        use the above
        //JY901.SendProtocolData(new byte[] { 0xff, 0xaa, 0x69, 0x88, 0xb5 });
        //JY901.SendProtocolData(new byte[] { 0xff, 0xaa, 0x01, 0x07, 0x00 });
        MessageBox.Show("To start the magnetic field calibration, please make one turn
        around each of the XYZ axes of the sensor, and click [End Magnetic Field
        Calibration] after completing the rotation.");
```

```
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
}

/// <summary>
/// end magnetic field calibration
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void endFieldCalibrationButton_Click(object sender, EventArgs e)
{
        if (JY901.IsOpen() == false)
        {
            return;
        }
        try
        {
            // Unlock registers and send commands
            JY901.UnlockReg();
            JY901.EndFieldCalibration();
            // The following two lines are equivalent to the above, and it is recommended to
            use the above
            //JY901.SendProtocolData(new byte[] { 0xff, 0xaa, 0x69, 0x88, 0xb5 });
            //JY901.SendProtocolData(new byte[] { 0xff, 0xaa, 0x01, 0x00, 0x00 });
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
}
```

SET THE RETURN RATE

Set the return rate of the sensor by calling JY901. SetReturnRate() method.

```
/// <summary>
/// Set the return rate 10Hz
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
```

```
private void returnRate10_Click(object sender, EventArgs e)
{
    if (JY901.IsOpen() == false)
    {
        return;
    }
    try
    {
        // Unlock registers and send commands
        JY901.UnlockReg();
        JY901.SetReturnRate(0x06);
        // The following two lines are equivalent to the above, and it is recommended to
        use the above
        //JY901.SendProtocolData(new byte[] { 0xff, 0xaa, 0x69, 0x88, 0xb5 });
        //JY901.SendProtocolData(new byte[] { 0xff, 0xaa, 0x03, 0x06, 0x00 });
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```

SET BANDWIDTH

Set the bandwidth of the sensor by calling JY901. SetBandWidth() method.

```
/// <summary>
/// Set the bandwidth 20Hz
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void bandWidth20_Click(object sender, EventArgs e)
{
    if (JY901.IsOpen() == false)
    {
        return;
    }
    try
    {
        // Unlock registers and send commands
        JY901.UnlockReg();
        JY901.SetBandWidth(0x04);
        // The following two lines are equivalent to the above, and it is recommended to
        use the above
        //JY901.SendProtocolData(new byte[] { 0xff, 0xaa, 0x69, 0x88, 0xb5 });
```

```
            //JY901.SendProtocolData(new byte[] { 0xff, 0xaa, 0x1F, 0x04, 0x00 });
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
}
```

Read sensor register

The sensor register can be read through the JY901.SendReadReg() method, or the JY901.SendProtocolData() method can be used
After sending the read command, the register value will be saved in JY901, and the register data needs to be obtained through JY901.GetDeviceData().

```
/// <summary>
/// read 03 register
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void readReg03Button_Click(object sender, EventArgs e)
{
    if (JY901.IsOpen() == false)
    {
        return;
    }
    try
    {
        // wait time
        int waitTime = 150;
        // Send the read command and wait for the sensor to return the data. If you don't read it, please extend the waiting time or read it several times.
        JY901.SendReadReg(0x03, waitTime);
```

```
            // The following line is equivalent to the above. It is recommended to use the
            above
            //JY901.SendProtocolData(new byte[] { 0xff, 0xaa, 0x27, 0x03, 0x00 }, waitTime);

            string reg03Value = JY901.GetDeviceData("03");
            MessageBox.Show($"The value of register 03 is : {reg03Value}");
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
}
```

Method	Directions	Parameter introduction	Return value
void SetPortName(string portName)	set serial port	portName： serial number	void
void SetBaudrate(int baudRate)	Specify the baud rate to turn on	baudRate： Baud rate	void
void Open()	Turn on the device	NO	void
bool IsOpen()	Is the device turned on	NO	Return whether to open open: true off: false
void Close()	Turn off the device	NO	void
void SendData(byte[] data, out byte[] returnData, bool isWaitReturn, int waitTime , int repetition)	send data	data: data to be sent returnData: the data returned by the sensor isWaitReturn: Whether the sensor needs to return data waitTime: time to wait for the sensor to return data, in ms, default 100ms repetition: repeated sending times	void
void SendProtocolData(byte[] data)	Send data with protocol	data: data to be sent	void
void SendProtocolData(byte[] data, int waitTime)	Send the data with the protocol and specify the waiting time	data: data to be sent waitTime: wait time	void
void SendReadReg(byte reg, int waitTime)	Send the command to read the register	reg: command to be sent wait time: wait time	void
void UnlockReg()	unlock register	NO	void
void SaveReg()	save register	NO	void
void AppliedCalibration()	Acceleration calibration	NO	void

void StartFieldCalibration()	Start magnetic field calibration	NO	void
void EndFieldCalibration()	end magnetic field calibration	NO	void
void SetReturnRate(byte rate)	Set return rate	rate: the return rate to be set	void
void SetBandWidth(byte band)	Set bandwidth	Band: the bandwidth to be set	void
string GetDeviceName()	get device name	NO	return device name
string GetDeviceData(string key)	Get key value data	key: data key value	return data value