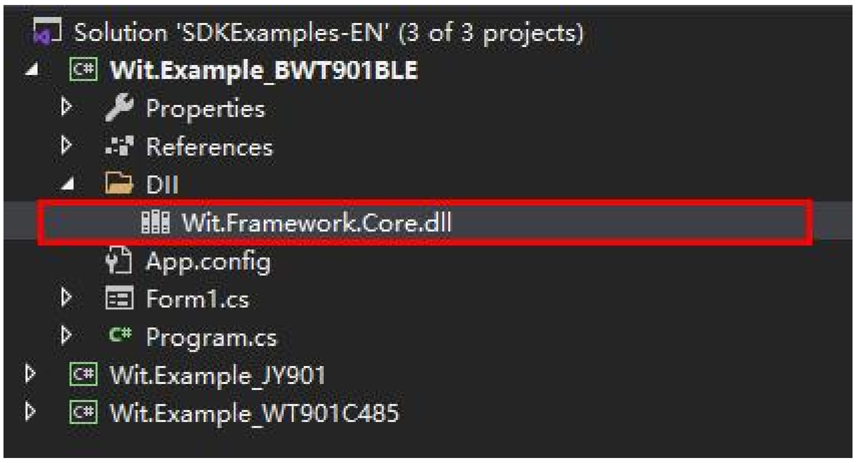
Bluetooth Protocol SDK Quick Guide

Bluetooth Protocol: It is the protocol used by WitMotion Bluetooth 5.0 sensors; the protocol stipulates that the sensor returns data packets beginning with 55, and the software sends data packets beginning with FF AA

This routine introduces how to use C# to develop the host computer to connect the Bluetooth 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



```
Solution 'SDKExamples-EN' (3 of 3 projects)
  Wit.Example_BWT901BLE
    Properties
    References
    Dll
      Wit.Framework.Core.dll
    App.config
    Form1.cs
    Program.cs
  Wit.Example_JY901
  Wit.Example_WT901C485
```

Open search

The BWT901BLE object represents the BWT901BLE device in the program, and you can communicate with the device through it; when searching for a device, call bluetoothManager.StartScan() to start the search; when opening the device, you need to specify the Bluetooth address of the Bluetooth sensor, and then call BWT901BLE.Open() method

```
/// <summary>
/// start search
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void startScanButton_Click(object sender, EventArgs e)
{
    // Clear found devices
    FoundDeviceDict.Clear();

    //Turn off previously turned on devices
    for (int i = 0; i < FoundDeviceDict.Count; i++)
    {
        var keyValue = FoundDeviceDict.ElementAt(i);
        BWT901BLE bWT901BLE = keyValue.Value;
        bWT901BLE.Close();
    }

    // Let the bluetooth manager start searching for devices
    WitBluetoothManager bluetoothManager =
    WitBluetoothManagerHelper.GetWitBluetoothManager();
    bluetoothManager.OnDeviceFound += new
    WitBluetoothManager.OnDeviceFoundHalder(OnFoundDevice);
    bluetoothManager.OnDeviceStatu += new
    WitBluetoothManager.OnDeviceStatuHalder(OnDeviceStatu);
    bluetoothManager.StartScan();
}

/// <summary>
/// Let the bluetooth manager start searching for devices
/// </summary>
/// <param name="macAddr"></param>
/// <param name="sName"></param>
/// <param name="dType"></param>
/// <param name="mType"></param>
```

```
private void OnFoundDevice(string macAddr, string sName, int dType, int mType)
{
    // If a Bluetooth device starting with WT is found
    if (sName != null && sName.Contains("WT"))
    {
        // If the device is newly found
        if (FoundDeviceDict.ContainsKey(macAddr) == false)
        {
            BWT901BLE bWT901BLE = new BWT901BLE();
            // Specify the connected Bluetooth MAC code
            bWT901BLE.SetMacAddr(macAddr);
            // set device name
            bWT901BLE.SetDeviceName(sName);
            FoundDeviceDict.Add(macAddr, bWT901BLE);
            // open this device
            bWT901BLE.Open();
            bWT901BLE.OnRecord += BWT901BLE_OnRecord;
        }
    }
}
```

Close search

Close the search and call the bluetoothManager.StopScan() method

```
/// <summary>
/// stop searching
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void stopScanButton_Click(object sender, EventArgs e)
{
    // stop the bluetooth manager from searching
    WitBluetoothManager bluetoothManager =
    WitBluetoothManagerHelper.GetWitBluetoothManager();
    bluetoothManager.StopScan();
}
```

Receive sensor data

GET DATA

The BWT901BLE object will automatically calculate the sensor data and save it on itself. The sensor data can be obtained through the BWT901BLE.GetDeviceData() method. BWT901BLE.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(BWT901BLE BWT901BLE)
{
        StringBuilder builder = new StringBuilder();
        builder.Append(BWT901BLE.GetDeviceName()).Append("\n");
        // acceleration

        builder.Append("AccX").Append(":").Append(BWT901BLE.GetDeviceData(WitSensorKey.
        AccX)).Append("g \t");

        builder.Append("AccY").Append(":").Append(BWT901BLE.GetDeviceData(WitSensorKey.
        AccY)).Append("g \t");

        builder.Append("AccZ").Append(":").Append(BWT901BLE.GetDeviceData(WitSensorKey.
        AccZ)).Append("g \n");
        // Angular velocity

        builder.Append("GyroX").Append(":").Append(BWT901BLE.GetDeviceData(WitSensorKe
        y.AsX)).Append("°/s \t");

        builder.Append("GyroY").Append(":").Append(BWT901BLE.GetDeviceData(WitSensorKey.
        AsY)).Append("°/s \t");

        builder.Append("GyroZ").Append(":").Append(BWT901BLE.GetDeviceData(WitSensorKey.
        AsZ)).Append("°/s \n");
        // angle

        builder.Append("AngleX").Append(":").Append(BWT901BLE.GetDeviceData(WitSensorKe
        y.AngleX)).Append("°  \t");

        builder.Append("AngleY").Append(":").Append(BWT901BLE.GetDeviceData(WitSensorKe
        y.AngleY)).Append("°  \t");
        builder.Append("AngleZ").Append(":").Append(BWT901BLE.GetDeviceData(WitSensorKe
        y.AngleZ)).Append("°  \n");
        // magnetic field

        builder.Append("MagX").Append(":").Append(BWT901BLE.GetDeviceData(WitSensorKey.
        HX)).Append("uT \t");

        builder.Append("MagY").Append(":").Append(BWT901BLE.GetDeviceData(WitSensorKey.
        HY)).Append("uT \t");

        builder.Append("MagZ").Append(":").Append(BWT901BLE.GetDeviceData(WitSensorKey.
        HZ)).Append("uT \n");
        // version number

        builder.Append("VersionNumber").Append(":").Append(BWT901BLE.GetDeviceData(Wit
        SensorKey.VersionNumber)).Append("\n");
        return builder.ToString();
}
```

RECORD DATA

The data of the sensor can be obtained through the BWT901BLE object, but usually the software needs to record the data of the sensor. BWT901BLE has an OnRecord event that will notify you when to record the data, and the OnRecord event can be realized when the device is turned on; then by cooperating with BWT901BLE.GetDeviceData( ) method to record data

```
/// <summary>
/// Let the bluetooth man ager start searching for devices
/// </summary>
/// <param name="macAddr"></param>
/// <param name="sName"></param>
/// <param name="dType"></param>
/// <param name="mType"></param>
private void OnFoundDevice(string macAddr, string sName, int dType, int mType)
{
        // If a Bluetooth device starting with WT is found
        if (sName != null && sName.Contains("WT"))
        {
                // If the device is newly found
                if (FoundDeviceDict.ContainsKey(macAddr) == false)
                {
                        BWT901BLE bWT901BLE = new BWT901BLE();
                        // Specify the connected Bluetooth MAC code
```

```
                    bWT901BLE.SetMacAddr(macAddr);
                    // set device name
                    bWT901BLE.SetDeviceName(sName);
                    FoundDeviceDict.Add(macAddr, bWT901BLE);
                    // open this device
                    bWT901BLE.Open();
                    bWT901BLE.OnRecord += BWT901BLE_OnRecord;
                }
        }
}

/// <summary>
/// This is called when sensor data is refreshed and you can log data here
/// </summary>
/// <param name="BWT901BLE"></param>
private void BWT901BLE_OnRecord(BWT901BLE BWT901BLE)
{
        string text = GetDeviceData(BWT901BLE);
        Debug.WriteLine(text);
}
```

Set up the sensor

The sensor can be operated by the method of BWT901BLE
BWT901BLE.UnlockReg() Send unlock register command
BWT901BLE.AppliedCalibration() Sends a plus calibration command
BWT901BLE.StartFieldCalibration() Send start magnetic field calibration command
BWT901BLE.EndFieldCalibration() Send end magnetic field calibration command
BWT901BLE.SetReturnRate() Send the command to set the return rate
BWT901BLE.SetBandWidth() Send command to set bandwidth
BWT901BLE.SendProtocolData() Send other commands

ACCELERATION CALIBRATION

Acceleration calibration of the sensor by calling the BWT901BLE.AppliedCalibration() method

```
/// <summary>
/// Acceleration calibration
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
```

```
private void appliedCalibrationButton_Click(object sender, EventArgs e)
{
        // All connected bluetooth devices are accelerometer calibrated
        for (int i = 0; i < FoundDeviceDict.Count; i++)
        {
                var keyValue = FoundDeviceDict.ElementAt(i);
                BWT901BLE bWT901BLE = keyValue.Value;

                if (bWT901BLE.IsOpen() == false)
                {
                        return;
                }

                try
                {
                        // Unlock registers and send commands
                        bWT901BLE.UnlockReg();
                        bWT901BLE.AppliedCalibration();
                        // The following two lines are equivalent to the above, and it is recommended
                        to use the above
                        //bWT901BLE.SendProtocolData(new byte[] { 0xff, 0xaa, 0x69, 0x88, 0xb5 });
                        //bWT901BLE.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 BWT901BLE .StartFieldCalibration() method and the BWT901BLE.EndFieldCalibration() method

```
/// <summary>
/// Start magnetic field calibration
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void startFieldCalibrationButton_Click(object sender, EventArgs e)
{
        // Start magnetic field calibration of all connected bluetooth devices
        for (int i = 0; i < FoundDeviceDict.Count; i++)
        {
            var keyValue = FoundDeviceDict.ElementAt(i);
            BWT901BLE bWT901BLE = keyValue.Value;

            if (bWT901BLE.IsOpen() == false)
            {
                return;
            }
            try
            {
                // Unlock registers and send commands
                bWT901BLE.UnlockReg();
                bWT901BLE.StartFieldCalibration();
                // The following two lines are equivalent to the above, and it is recommended
                to use the above
                //bWT901BLE.SendProtocolData(new byte[] { 0xff, 0xaa, 0x69, 0x88, 0xb5 });
                //bWT901BLE.SendProtocolData(new byte[] { 0xff, 0xaa, 0x01, 0x07, 0x00 });
                MessageBox.Show("To start the magnetic field calibration, please make one
                turn around the XYZ axis of the sensor, and click [End Magnetic Field
                Calibration]");
            }
            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)
    {

        // End the magnetic field calibration of all connected Bluetooth devices
        for (int i = 0; i < FoundDeviceDict.Count; i++)
        {
            var keyValue = FoundDeviceDict.ElementAt(i);
            BWT901BLE bWT901BLE = keyValue.Value;
            if (bWT901BLE.IsOpen() == false)
            {
                return;
            }
            try
            {
                // Unlock registers and send commands
                bWT901BLE.UnlockReg();
                bWT901BLE.EndFieldCalibration();
                // The following two lines are equivalent to the above, and it is recommended
                to use the above
                //bWT901BLE.SendProtocolData(new byte[] { 0xff, 0xaa, 0x69, 0x88, 0xb5 });
                //bWT901BLE.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 BWT901BLE. 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)
{
    for (int i = 0; i < FoundDeviceDict.Count; i++)
    {
        var keyValue = FoundDeviceDict.ElementAt(i);
        BWT901BLE bWT901BLE = keyValue.Value;

        if (bWT901BLE.IsOpen() == false)
        {
            return;
        }
        try
        {
            // Unlock registers and send commands
            bWT901BLE.UnlockReg();
            bWT901BLE.SetReturnRate(0x06);
            // The following two lines are equivalent to the above, and it is recommended
            to use the above
            //bWT901BLE.SendProtocolData(new byte[] { 0xff, 0xaa, 0x69, 0x88, 0xb5 });
            //bWT901BLE.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 BWT901BLE. SetBandWidth() method.

```
/// <summary>
/// Set the bandwidth 20Hz
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void bandWidth20_Click(object sender, EventArgs e)
{
    for (int i = 0; i < FoundDeviceDict.Count; i++)
    {
        var keyValue = FoundDeviceDict.ElementAt(i);
        BWT901BLE bWT901BLE = keyValue.Value;

        if (bWT901BLE.IsOpen() == false)
        {
            return;
        }
        try
        {
            // Unlock registers and send commands
            bWT901BLE.UnlockReg();
            bWT901BLE.SetBandWidth(0x04);
            // The following two lines are equivalent to the above, and it is recommended
            to use the above
            //bWT901BLE.SendProtocolData(new byte[] { 0xff, 0xaa, 0x69, 0x88, 0xb5 });
            //bWT901BLE.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 BWT901BLE.SendReadReg() method, or the BWT901BLE.SendProtocolData() method can be used
After sending the read command, the register value will be saved in BWT901BLE, and the register data needs to be obtained through BWT901BLE.GetDeviceData().

```
/// <summary>
/// read 03 register
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void readReg03Button_Click(object sender, EventArgs e)
{
    string reg03Value = "";
    // Read the 03 register of all connected bluetooth devices
    for (int i = 0; i < FoundDeviceDict.Count; i++)
    {
        var keyValue = FoundDeviceDict.ElementAt(i);
        BWT901BLE bWT901BLE = keyValue.Value;

        if (bWT901BLE.IsOpen() == false)
        {
            return;
        }
        try
        {
            // wait time
            int waitTime = 3000;
```

```
                // Send a read command and wait for the sensor to return data. If you don't
                read it, you can extend the waitTime or read it several times.
                bWT901BLE.SendReadReg(0x03, waitTime);

                // The following line is equivalent to the above. It is recommended to use the
                above
                //bWT901BLE.SendProtocolData(new byte[] { 0xff, 0xaa, 0x27, 0x03, 0x00 },
                waitTime);

                // Get the value of all connected bluetooth devices
                reg03Value += bWT901BLE.GetDeviceName() + " register 03 value is:" +
                bWT901BLE.GetDeviceData("03") + "\r\n";
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
        MessageBox.Show(reg03Value);
    }
```

BWT901BLE API

Method	Directions	Parameter introduction	Return value
void SetMacAddr(string macAddr)	Set the bluetooth address to open	macAddr: bluetooth address	void
void SetDeviceName(string DeviceName)	Set device name	DeviceName: device name	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