接口说明 v1.0

欧阳嘉鸿

2024 年 8 月 4 日

GetDeviceData 方法是 DeviceDataSource 类的一个公用方法，用于获取设备数据。该方法有多个重载版本，分别用于获取 string 和 double 类型的数据。其中，用于获取 double 类型数据的重载版本接受一个 DoubleKey 类型的参数，并返回一个 double? 类型的值，即可能返回 null 的 double 类型值。

GetDeviceData 方法（double 重载）的实现细节：

1. 方法首先获取传入 DoubleKey 对象的 Key 属性，该属性是一个字符串，表示要获取的数据的键。

2. 接着，方法检查 DeviceData 字典（假设是类的一个成员，用于存储设备数据）是否包含该键。

3. 如果包含，方法会尝试从字典中获取对应键的值，并将其转换为 double 类型。

4. 如果转换成功，方法返回该 double 值；如果转换失败（即值不是 double 类型），方法返回 null。

5. 如果 DeviceData 字典不包含该键，方法同样返回 null。

```
    /// <summary>
    /// 获得数据
    /// </summary>
    /// <param name="dataKey"></param>
    /// <returns></returns>
    public double? GetDeviceData(DoubleKey dataKey)
    {
      string key = dataKey.Key;
      if (DeviceData.ContainsKey(key))
      {
        object o = DeviceData[key];
        if (o is double)
        {
          return (double)o;
        }
        return null;
      }
      else
      {
        return null;
      }
    }

```

注：GetDeviceData 方法（double 重载）的定义

在程序外部，可以通过创建 DeviceDataSource 类的实例（如 BWT901BLE）并调用其 GetDeviceData 方法来获取设备数据。例如，要获取当前时刻的 x 轴加速度值，可以传入 WitSensorKey.AccX 作为参数。WitSensorKey 是一个包含设备数据关键字的类，AccX 是其一个静态成员，表示 x 轴加速度数据的关键字。

在我们项目的编程中，使用 BWT901BLE 作为*传感器实例*的名称，我们通过使用*方法访问操作符*（即“.”）访问 DeviceDataSource 实例的 GetDeviceData 方法，利用上述代码中的语句即可获得当前时刻的 x 轴加速度值。其中 WitSensorKey.AccX 为需要传入 GetDeviceData 方法的关键字，在使用的时候不需要管它的定义。

```
1    double AccX = BWT901BLE.GetDeviceData(WitSensorKey.AccX);
2
```

注：使用 GetDeviceData 方法的示例

如下代码中还列出了所有可能用到的数据关键字及其对应的数据类型。这些关键字包括芯片时间、加速度（x、y、z 轴和矢量和）、角速度（x、y、z 轴和矢量和）、角度（x、y、z 轴）、温度、四元数（0、1、2、3）、版本号、序列号和电量等。通过传入这些关键字作为参数，可以获取相应的设备数据。

```
1    // 芯片时间
2    string ChipTime =BWT901BLE.GetDeviceData(WitSensorKey.ChipTime);
3    // 加速度X
4    double AccX =BWT901BLE.GetDeviceData(WitSensorKey.AccX);
5    // 加速度Y
6    double AccY =BWT901BLE.GetDeviceData(WitSensorKey.AccY);
7    // 加速度Z
8    double AccZ =BWT901BLE.GetDeviceData(WitSensorKey.AccZ);
9    // 加速度矢量和
10   double AccM =BWT901BLE.GetDeviceData(WitSensorKey.AccM);
11   // 角速度
12   double AsX =BWT901BLE.GetDeviceData(WitSensorKey.AsX);
13   // 角速度
14   double AsY =BWT901BLE.GetDeviceData(WitSensorKey.AsY);
15   // 角速度
16   double AsZ =BWT901BLE.GetDeviceData(WitSensorKey.AsZ);
17   // 角速度Z矢量和
18   double AsM =BWT901BLE.GetDeviceData(WitSensorKey.AsM);
19   // 角度X
20   double AngleX =BWT901BLE.GetDeviceData(WitSensorKey.AngleX);
21   // 角度Y
22   double AngleY =BWT901BLE.GetDeviceData(WitSensorKey.AngleY);
23   // 角度Z
```

```
24      double AngleZ =BWT901BLE.GetDeviceData(WitSensorKey.AngleZ);
25      // 温度
26      double T =BWT901BLE.GetDeviceData(WitSensorKey.T);
27      // 四元数0
28      double Q0 =BWT901BLE.GetDeviceData(WitSensorKey.Q0);
29      // 四元数1
30      double Q1 =BWT901BLE.GetDeviceData(WitSensorKey.Q1);
31      // 四元数2
32      double Q2 =BWT901BLE.GetDeviceData(WitSensorKey.Q2);
33      // 四元数3
34      double Q3 =BWT901BLE.GetDeviceData(WitSensorKey.Q3);
35      // 版本号
36      string VersionNumber =BWT901BLE.GetDeviceData(WitSensorKey.
        VersionNumber);
37      // 序列号
38      string SerialNumber =BWT901BLE.GetDeviceData(WitSensorKey.SerialNumber)
        ;
39      // 电量
40      double PowerPercent =BWT901BLE.GetDeviceData(WitSensorKey.PowerPercent)
        ;
41
```

在具体使用过程中，首要任务是创建一个适宜的容器，用于存储一段时间内采集的数据，随后对这些数据进行分段处理。鉴于传感器以 200Hz 的频率获取数据，编程时需特别考虑容器的大小和计算效率，这对数据处理至关重要。同时，传感器的数据获取频率直接影响到分段处理的策略。因此，在编程过程中，必须兼顾实际需求与资源限制，以确保代码的可行性和可维护性。