Преобразование и хранение данных нового протокола в СКАУТ-Платформе

Процесс формирования сообщений и передачи их в Ядро СКАУТ-Платформы следующий:

1. Парсинг сообщения
2. Формирование сообщения (DeviceMessage) при помощи класса DeviceMessageBuilder
3. Отправка сообщений в хранилище, при помощи IDeviceDataStorage

`DeviceMessageBuilder` – построитель сообщений

`DeviceMessage` – сообщение в формате СКАУТ-Платформы

`IDeviceDataStorage` – интерфейс хранилища данных для сообщений

Перед отправкой на сохранение сообщения необходимо преобразовать в `DeviceMessage`. Сам формат сообщения не очень интересен, так как есть удобный в использовании класс `DeviceMessageBuilder`, данный класс работает как `StringBuilder`. То есть мы постепенно или сразу (как удобнее) заносим данные в построитель.

Для добавления значений датчиков в классе `DeviceMessageBuilder` присутствует метод `void SetSensorValue<T>(T value, DeviceSensorDataType dataSourceType, ushort dataSourcePort)`

T – один из примитивных типов .NET (`Byte, SByte, Int16, UInt16, Int32, UInt32, Int64, UInt64, Single, Double, Boolean, Char, String`)

`value` – значение датчика

`dataSourceType` – тип датчика, следует использовать `DeviceSensorDataType.Multi`

`dataSourcePort` – номер порта устройства

Данный метод заносит значения датчиков в коллекцию, поэтому может быть вызван неоднократно для одного сообщения для разных датчиков и портов.

Когда сообщение за определенную дату полностью сформировано, запрашиваем у построителя экземпляр сообщения с помощью метода `ToDataMessage`.

Процессом сохранения сообщений занимается сама платформа, от обработчика протокола требуется только воспользоваться объектом dataStorage, который реализует интерфейс `IDeviceDataStorage`.

В `IDeviceDataStorage` есть 2 интересующие нас перегрузки метода `Store`:

- `Store(DeviceMessage message)` – сохраняет одно сообщение
- `Store(ICollection<DeviceMessage> deviceMessages)` – сохраняет

коллекцию сообщений.