IIoT
НА ГРАНИЦЕ HW И .NET

Антон Сысоев
Руководитель отдела разработки

“Вещи” в IIoT

- Промышленные датчики
- Приборы учета
- Промышленное оборудование
- Все, что не IoT, но хочет в Сеть :)

Внедрение IIoT

- ~~Построить “с нуля”~~ дорогую
- Поставить ~~“умную”~~ железку
- Подключиться “напрямую”

Типовая схема IoT

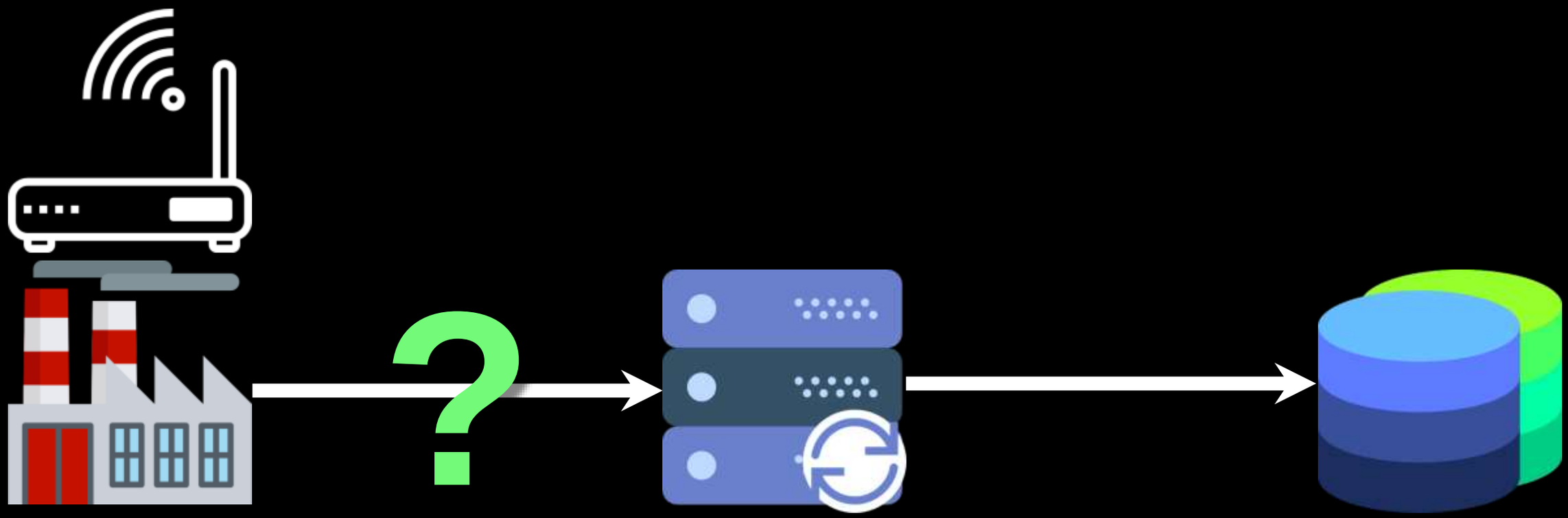
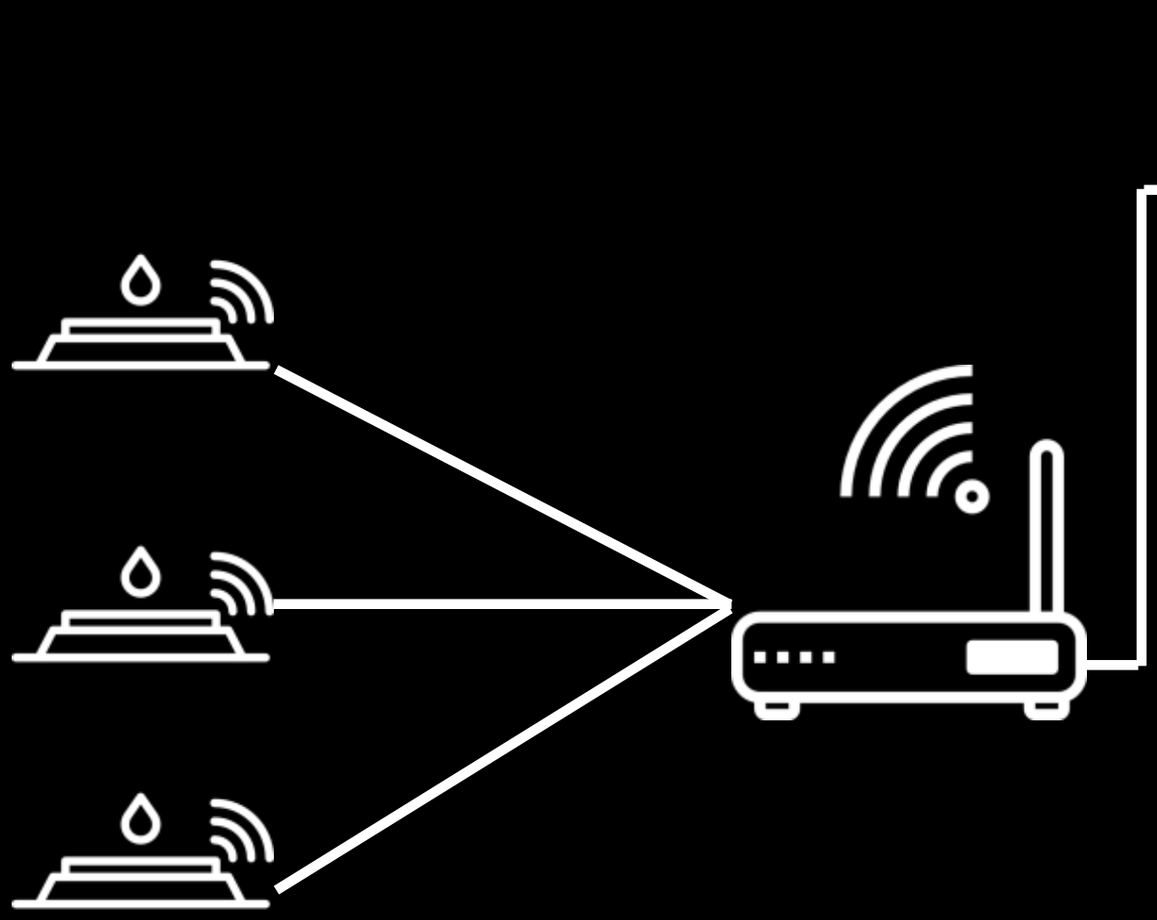
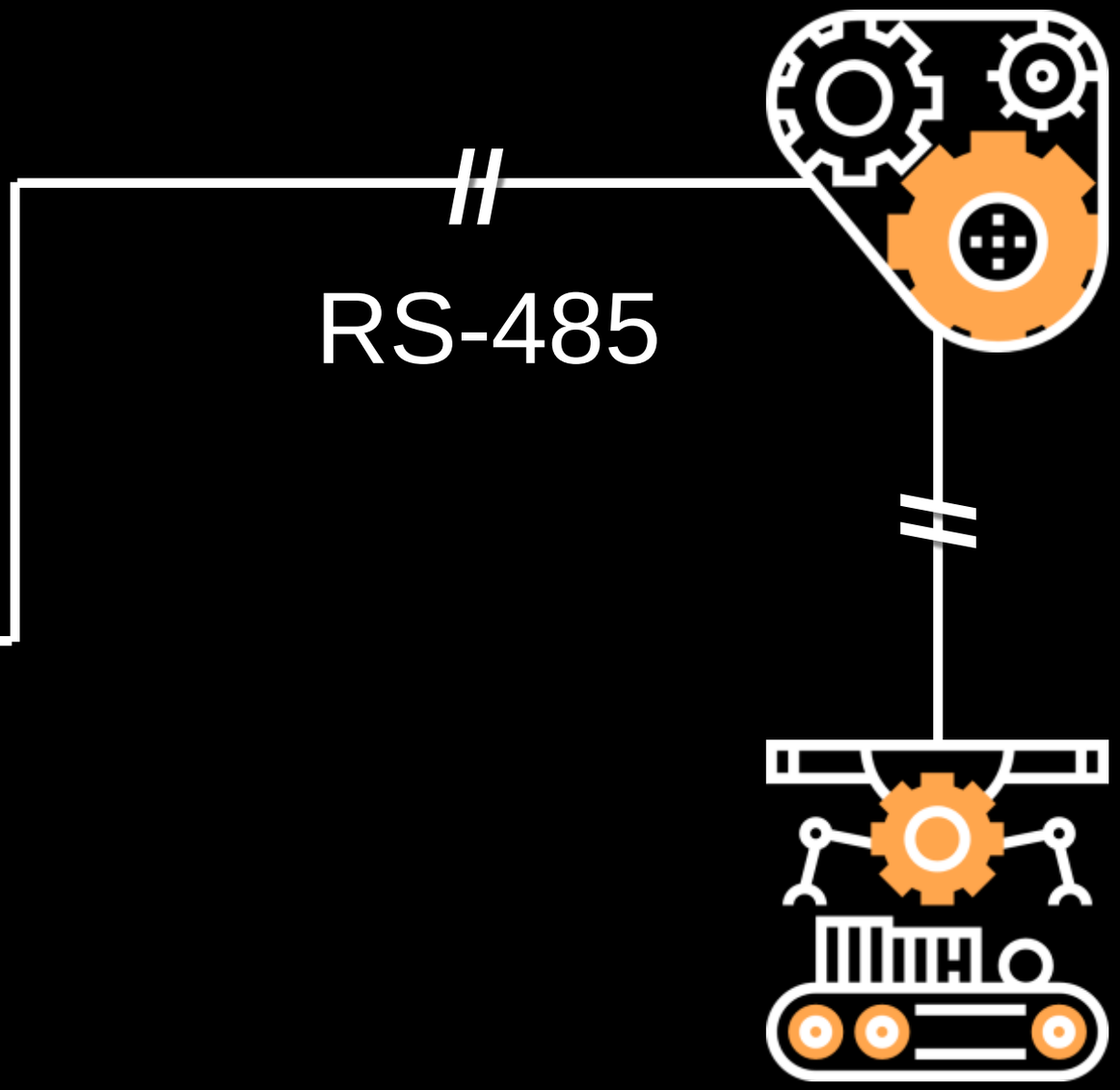
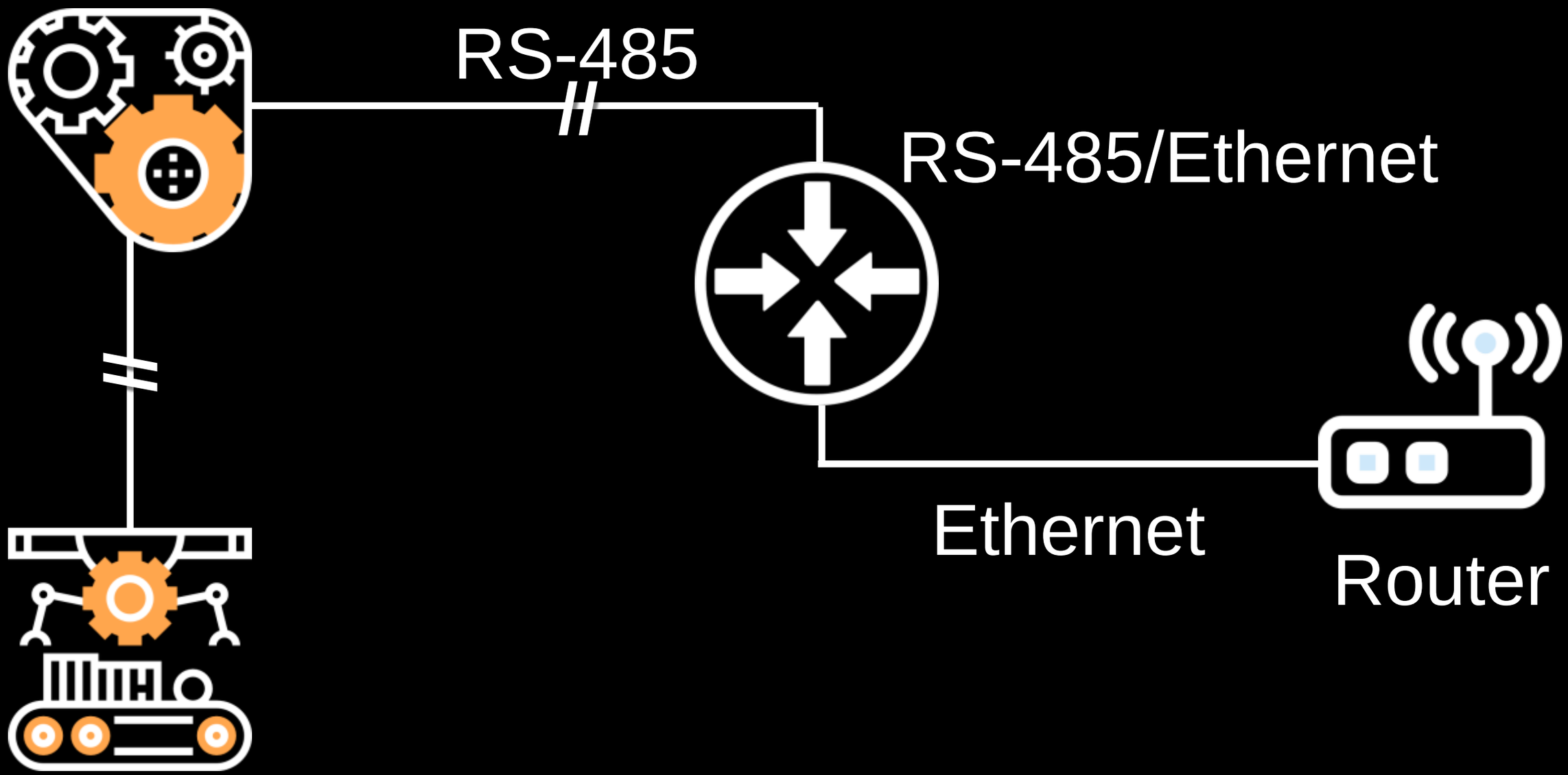


Схема подключения

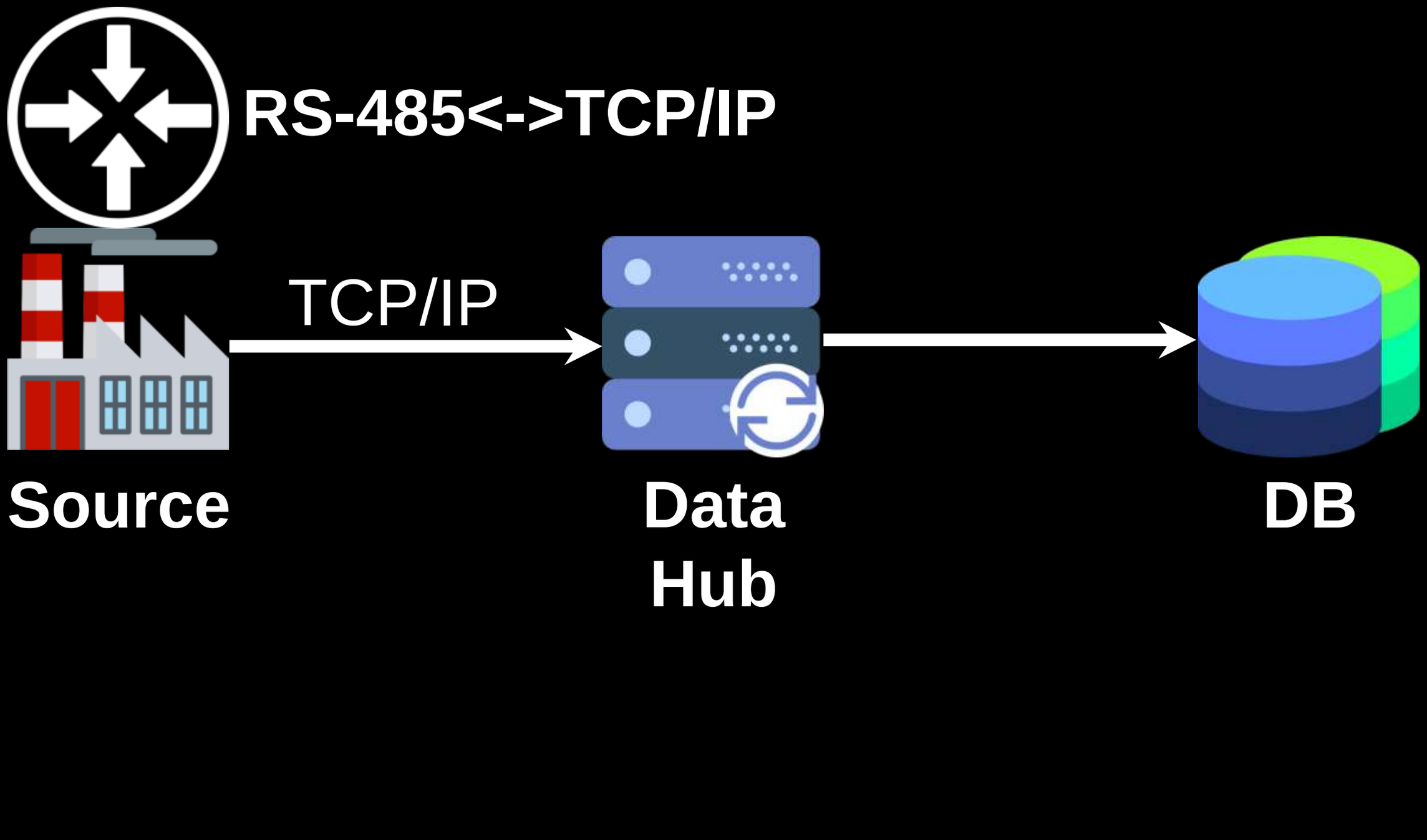




RS-485<->TCP/IP



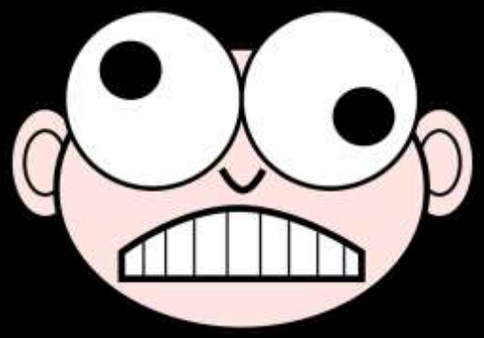
“Прямая” схема IIoT



Проблемы связи

Монтаж

Засыпания устройств



Ошибки fw

Помехи

Упрощенные протоколы

Request

ADDR	BODY (N bytes)	CRC

Response

ADDR	BODY (M bytes)	CRC

ERROR

ADDR	BODY (K bytes)	CRC

Потоковая передача

	start	0	1	2	...	6	7	par	stop	

FIFO buffer

byte 1	byte 2	byte 3	...	byte N-1	byte N

Толстеющие протоколы

Request (v.1)

ADDR	N bytes	CRC

Request (v.2)

ADDR	N bytes	1 byte	CRC

Request (v.3)

ADDR	Y bytes	1 byte	Z bytes	CRC

Потоковые протоколы

[...Noise...]

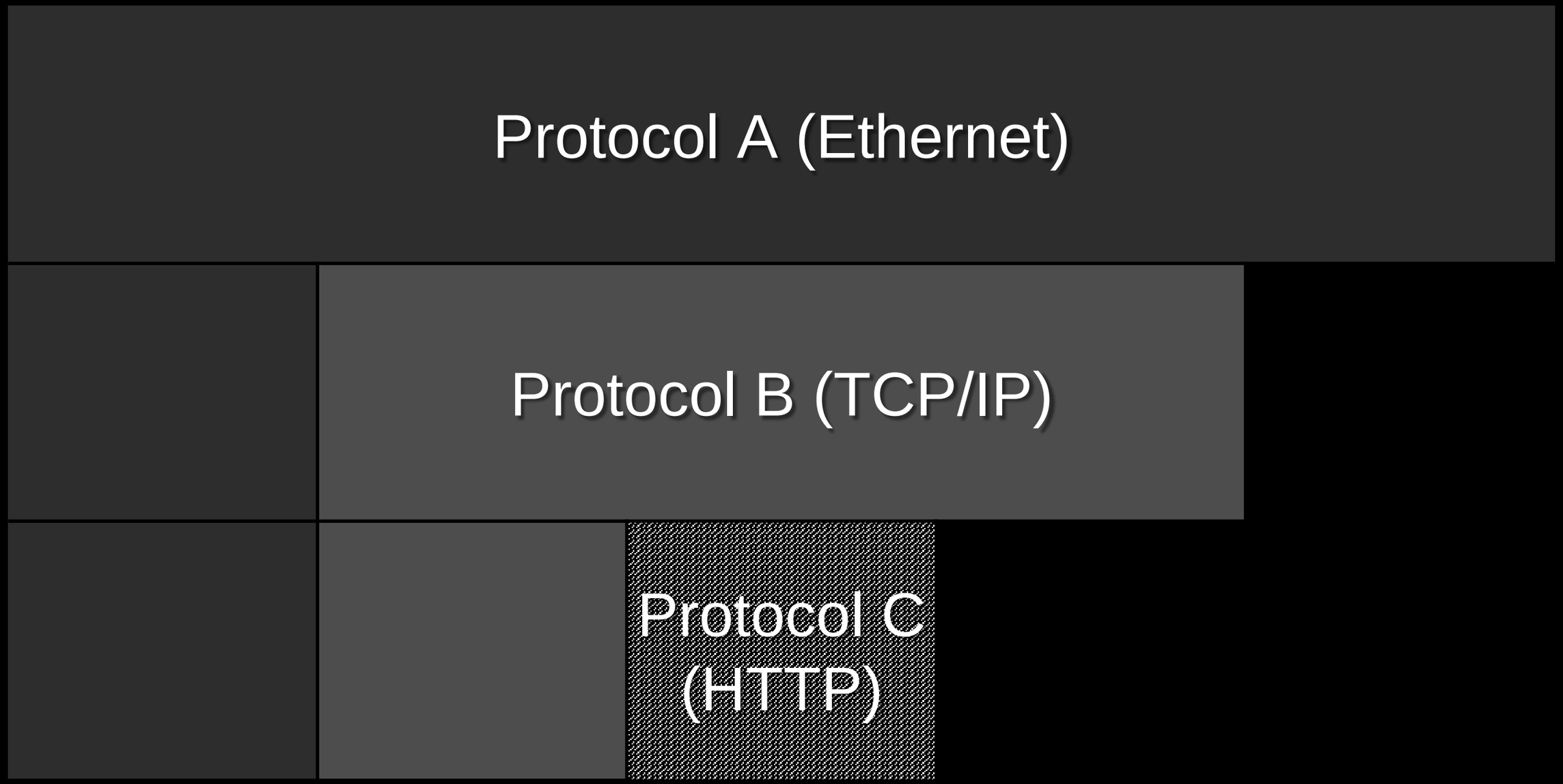
Start marker

Escaped payload data

End marker

[...Noise...]

Инкапсуляция протоколов



Дизайн драйвера

```
public int  IHi  {
}

public interf  Level{
}
```

Обработка ошибок

- Проверка CRC
- Не всегда Exception однозначен
- Повторы запросов

Реализация протокола

- Наследование Stream
- Потоковая или блочная обработка
- Наследование при инкапсуляции протоколов

Транспортные протоколы

- Modbus
- IEC (МЭК)
- DLMS/COSEM
- …

Разбор данных

- Работа с массивом байт
- BitConverter
- BinaryReader
- Reflection
- Unsafe code (C-style)
- Marshaling

Упаковка пакета

```
UInt16 ch2 = 0;
ch2 |= (UInt16)(1 <<

buf[len++] = (byte) ch
buf[len++] = (byte)(ch2

buf[len++] = 0x00;
buf[len++] = 0x00;

buf[len++] = (by
buf[len++] = (                8);
```

Р а пакета

```
      = reply[1];
    ter = (statusB & 0x08) == 8;
  otFull = (statusB & 0x02) == 2;
  t17 = (statusB >> 4) & 0x1;
```

Типовой пакет данных

HEADER - заголовок, общий для всех пакетов
BODY - тело пакета с данными
CRC - контрольная сумма, общая для всех пакетов

Marshaling Helper

```
public static byte[] BytesFromStructArray(TValue[] srcValues);
public static TValue[] StructArrayFromBytes(byte[] srcBytes,
                                            int offset = 0,
                                            int count = 0);
public static TValue StructFromBytes(byte[] srcBytes,
                                     int offset = 0,
                                     int count = 0)
public static byte[] BytesFromStruct(TValue value)
```

```
public interface IRequest
{
    byte[] GetBytes();
}
public interface IResponse
{
    void SetBytes(byte[] bytes, int offset, int count);
    ushort FrameCrc { get; }
}
public interface IPackageBody
{
    byte[] GetBytes();
    void SetBytes(byte[] bytes, int offset, int count);
    byte GetLength();
}
```

```
public class RequestDataFrame:IRequest
{
    private byte[] _bodyData = null;
    public IPackageBody Body
    {
        set { _bodyData = value.GetBytes(); }
    }
        public void SetBody<TFrameBody>(TFrameBody body) where
                                                TFrameBody : IPackageBody
        {
            _bodyData = body.GetBytes();
        }
//Skiped code
    }
```

```
[StructLayout(LayoutKind.Sequential, Pack = 1)]
internal struct DataHolder
{
    [MarshalAs(UnmanagedType.U4)]
    public uint Mask;
        ...
}
public class SampleRequestBody : IPackageBody
{
    private DataHolder _dataHolder;
    public BitVector32 Mask { set { _dataHolder.Mask = value.Data; } }
    public byte[] GetBytes()
    {
        return MarshallingHelper<DataHolder>.BytesFromStruct(_dataHolder);
    }
    public byte GetLength()
    {
        return (byte)Marshal.SizeOf(typeof(DataHolder));
    }
}
```

Запрос данных по маске

Request Mask (0x06)

0	0	0	0	0	1	1	0
7	6	5	4	3	2	1	0

Response

[Header]
Data 2
Data 3
[Footer]

Вопросы