COGNITIVE PILOT

ПРОТОКОЛ ИНФОРМАЦИОННОГО ВЗАИМОДЕЙСТВИЯ

Универсальный радар CPRR

Содержание

1. Введение

Настоящий протокол определяет информационное взаимодействие с универсальным радаром CPRR.

Универсальный радар CPRR предназначен для решения задач обзора пространства и обнаружения различных категорий объектов в широком секторе углов в сферах автоматизации транспорта и промышленной безопасности.

Радар производит обзор пространства, осуществляет сбор и предварительную обработку данных о нахождении подвижных и неподвижных объектов в зоне видимости.

Рисунок 1 — Универсальный радар CPRR (без монтажного кронштейна)
виды спереди, сзади и сбоку

Протокол информационного взаимодействия с радаром CPRR может быть изменен путем выпуска его новых версий.

2. Информационный обмен с радаром CPRR

2.1 Передача данных

Данные передаются с помощью интерфейса Ethernet и транспортного протокола UDP. При этом устройство пользователя – это клиент, а изделие – сервер.

Радар имеет два режима работы: режим поточной передачи данных об окружающих объектах (см. рисунок 2) и режим запроса данных об окружающих объектах вручную (см. рисунок 3).

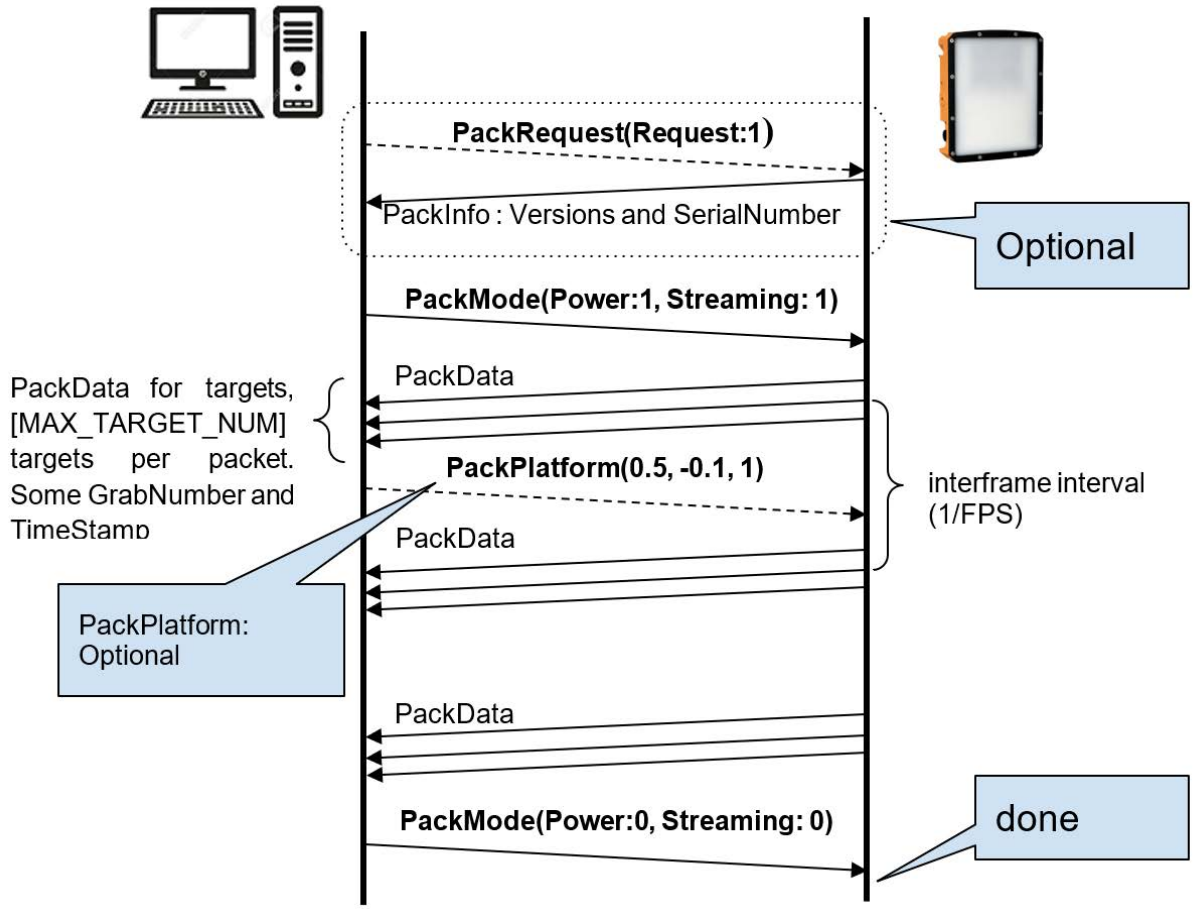

Рисунок 2 — Режим поточной передачи данных об окружающих объектах

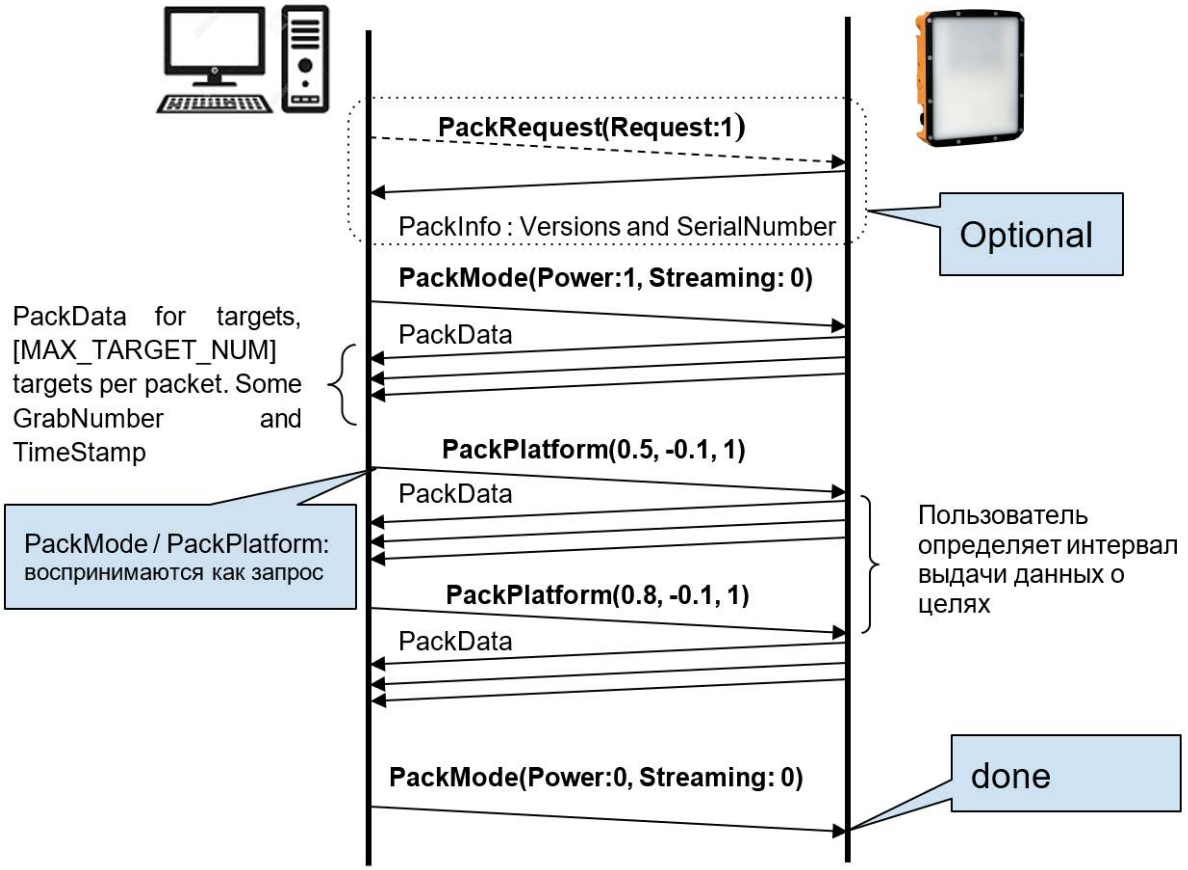

Рисунок 3 — Режим запроса данных об окружающих объектах вручную

ПРИМЕЧАНИЕ:

`MAX_TARGET_NUM` – это максимальное число детектированных объектов, информация о которых передается в одном пакете.

Если выбран режим запроса данных об окружающих объектах вручную и изделие получило пакет **PackPlatform** до того, как информация об обнаруженных объектах была обновлена (частота запроса выше, чем FPS изделия), то изделие передаст пакет **PackData** без поля Targets, а другие поля будут иметь те же значения, что и в предыдущем пакете **PackData**.

2.2 Структура UDP пакета

Структура пакета – единая для всех передаваемых и получаемых пакетов (см. рисунок 4).

Рисунок 4 — Структура пакета UDP

```
typedef struct {
      uint32_t Preamble;
      uint32_t Len;
      uint32_t Type
      Data_t Data;
} Pack;
```

Описание полей пакета приведено в таблице 1

Таблица 1 — Поля UDP пакета

Поле	**Описание**
Preamble	Используется для определения начала следующего пакета, всегда имеет значение 0x0000ABCD
Len	Равно длине поля **Data** в байтах
Type	Тип пакета *(см. таблицу 2)*
Data	Зависит от типа пакета

Таблица 2 — Соответствие значения поля Type типу пакета

Значение поля «Type»	**Тип пакета**
1	PackRequest
2	PackMode
3	PackPlatform
4	PackData

5	PackInfo
6	PackError

Далее представлено подробное описание структуры поля **Data** в зависимости от типа пакета.

2.2.1 Тип пакета **PackMode**

Структура поля **Data** для типа пакета **PackMode** показана на рисунке 5, описание ячеек поля приведено в таблице 3.

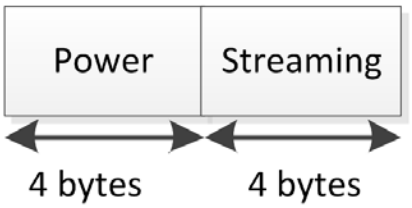

Рисунок 5 — Структура поля **Data** пакета **PackMode**

Таблица 3 — Описание ячеек поля **Data** пакета **PackMode**

Ячейка	**Описание**
Power	Команда включения: 1 – включено; 0 – выключено.
Streaming	Поточная передача данных об обнаруженных объектах: 1 – начать потоковую передачу; 0 – остановить потоковую передачу.

2.2.2 Тип пакета **PackPlatform**

Структура поля **Data** пакета **PackPlatform** показана на рисунке 6, описание ячеек поля приведено в таблице 4.

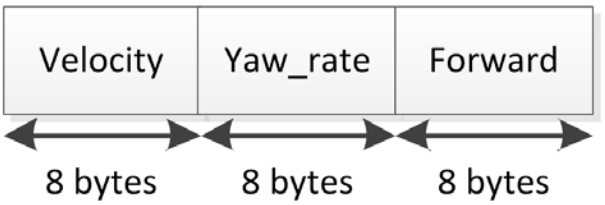

*Рисунок 6 — Структура поля **Data** пакета **PackPlatform***

```
typedef struct {
      double Velocity;
      double Yaw_rate;
      uint64_t Forward;
} PackPlatform_t;
```

Таблица 4 — Описание ячеек поля Data пакета PackPlatform

Ячейка	**Описание**
Velocity	Собственная скорость в м/с
Yaw _rate	Угловая скорость в азимутальной плоскости в рад/с
Forward	Направление движения: 1 – движение сонаправлено оси Y; 0 – движение противонаправлено оси Y.

2.2.3 Тип пакета **PackData**

Структура поля **Data** пакета **PackData** показана на рисунке 7, описание ячеек поля приведено в таблице 5.

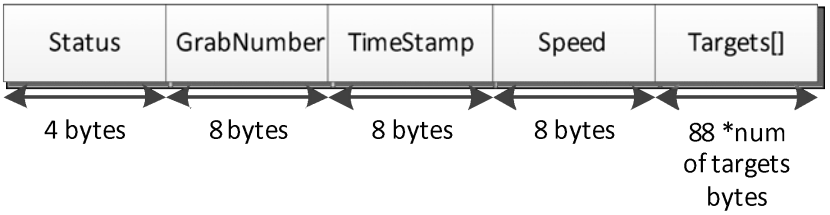

*Рисунок 7 — Структура поля **Data** пакета **PackData***

```
typedef struct {
      uint32_t Status;
      uint64_t GrabNumber;
      uint64_t TimeStamp;
      double   Speed;
      Target_t Targets [MAX_TARGET_NUM];
} PackData_t;
```

Таблица 5 — Описание ячеек поля **Data** пакета **PackData**

Ячейка	**Описание**
Status	Битовая маска *(см. таблицу 6)*
GrabNumber	Номер кадра
TimeStamp	Время захвата кадра в микросекундах с момента включения CPRR
Speed	Собственная скорость устройства
Targets []	Массив элементов типа **Target_t**. Количество элементов равно количеству обнаруженных объектов *(в данном пакете, не более **MAX_TARGET_NUM**)*

Таблица 6 — Описание бит в ячейке **Status**

Количество бит в ячейке Status	**Описание**
0	Исправность: 1 – работает нормально;

	0 – неисправно.
1	Зарезервировано
..	
31	

Структура элементов массива **Targets** показана на рисунке 8, а описание ячеек приведено в таблице 7.

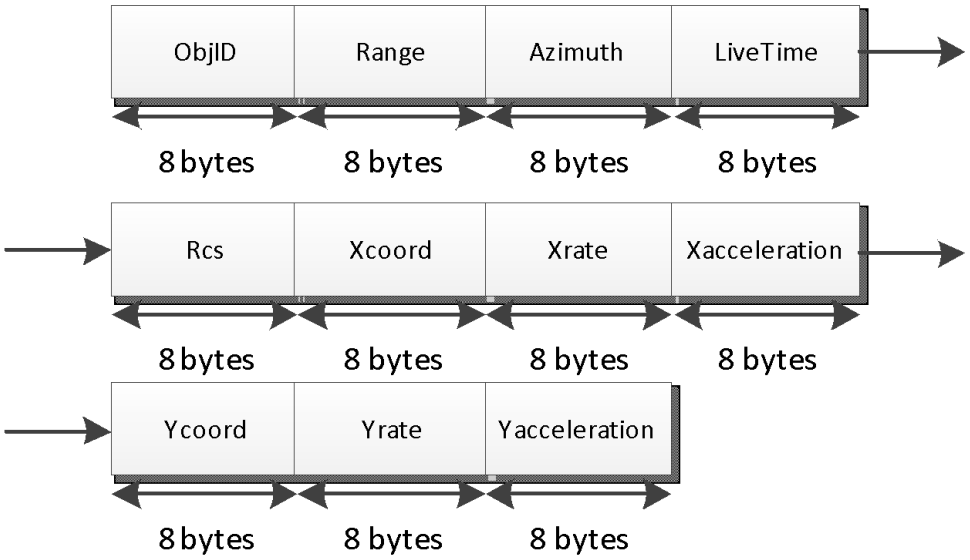

*Рисунок 8 — Структура массива **Targets***

```
typedef struct {
      int64_t ObjID;
      double Range;
      double Azimuth;
      int64_t LiveTime;
      double Rcs;
      double Xcoord;
      double Xrate;
      double Xacceleration;
      double Ycoord;
      double Yrate;
      double Yacceleration;
```

```
} Target_t;
```

Таблица 7 — Описание ячеек массива **Targets**

Ячейка	**Описание**
ObjID	Идентификатор обнаруженного объекта *(ID)*
Range	Расстояние до объекта в метрах
Azimuth	Угол между осью Y и направлением до детектированного объекта в азимутальной плоскости, в градусах. В этом случае направление вращения против часовой стрелки вокруг оси Z принимается как отрицательное
LiveTime	Время в миллисекундах, в течение которого данный объект был вне поля зрения
Rcs	Уровень сигнала, отраженного от объекта
Xcoord	Координата X данного объекта, в метрах
Xrate	Скорость вдоль оси X, в м/с
Xacceleration	Ускорение вдоль оси X, в м/с^2
Ycoord	Координата Y данного объекта, в метрах
Yrate	Скорость вдоль оси Y, в м/с
Yacceleration	Ускорение вдоль оси Y, в м/с^2

2.2.4 Тип пакета **PackError**

Данный тип пакета изделие отправляет клиенту, если изделие неисправно.

```
typedef struct {
      uint32_t Error;
} PackError_t;
```

Расшифровка кодов ошибок приведена в таблице 8.

Таблица 8 — Коды ошибок пакета PackError

Код ошибки	Описание
0	Попадание грязи или снега на изделие либо его обледенение
1	Последний полученный пакет был некорректным
2	Зарезервировано
…	
31	

2.2.5 Тип пакета **PackRequest**

Данный тип пакета клиент отправляет на изделие для запроса **PackInfo**.

```
typedef struct {
      uint32_t Request;
} PackRequest_t;
```

Расшифровка кодов приведена в таблице 9.

Таблица 9 — Допустимые значения поля **Request** пакета **PackRequest**

Поле	**Описание**
Request	1 – информация о версии *(ответ: **PackInfo**)*

2.2.6 Тип пакета **PackInfo**

Структура поля **Data** пакета **PackInfo** показана на рисунке 9, описание ячеек поля приведено в таблице 10.

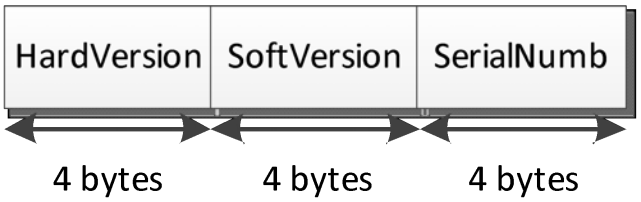

*Рисунок 9 — Структура поля **Data** пакета **PackInfo***

```
typedef struct {
     Version HardVersion;
     Version SoftVersion;
     Version SerialNumb;
} PackInfo_t;
```

```
typedef struct {
     uint16_t minor;
     uint16_t major;
} Version;
```

Таблица 10 — Описание ячеек поля **Data** пакета **PackInfo**

Поле	**Описание**
HardVersion	Номер версии аппаратного обеспечения изделия. Состоит из двух частей *(старшей и младшей)*
SoftVersion	Номер версии программного обеспечения изделия. Состоит из двух частей *(старшей и младшей)*
SerialNumb	Серийный номер изделия *(в паспорте указан в десятичном формате)*