Dicas e sequência de implementação do protocolo

Rastreador New Tracker

NT20

www.x3tech.com.br

Rev. 2.1 – agosto-19

Este documento descreve uma sequência lógica de implementação do protocolo TCP.

1) Formato das mensagens trafegadas:

Do servidor para o rastreador:

0x78 0x78 <Lenght (1 byte)><Protocolo (1 byte)><Conteúdo (N bytes)><Serial (N bytes)><Error check (2 bytes)> 0x0D 0x0A

O número do protocolo assume os valores conforme Tabela 1:

Tabela 1 -Número do Protocolo

Type	Value
Login Message	0x01
Location Data	0x12
Status information	0x13
String information	0x15
Alarm data	0x16
Location Data X3Tech (*)	0x22
GPS, query address information by phone number	0x1A
Command information sent by the server to the terminal	0x80

- Considerar apenas o conteúdo das mensagens entre os dois delimitadores 0x78 0x78 e 0x0D 0x0A, desprezando algo que seja transmitido fora destes.
- Pode vir mais de uma mensagem dentro de um pacote TCP, assim como uma mesma mensagem pode se iniciar num pacote e terminar em outro pacote TCP
- O algoritmo de checksum está descrito em apêndice do documento de descrição no manual do protocolo. O método utilizado é o CRC-ITU.

2) Implementar confirmação do login

- Estabelece a conexão do equipamento com o servidor;
- Servidor deve responder com a mensagem de confirmação;
- A partir do login bem sucedido o equipamento comunica regularmente;
- Ver item 5.1 no manual do protocolo nas pag. 8 e 9

3) Implementar confirmação do heartbeat

- Toda mensagem de heartbeat (status) deve ter sua resposta imediata a fim de manter o equipamento conectado ao servidor;
- Ver item 5.4 no manual do protocolo pag 18 a 20

4) Pacotes de localização

O rastreador NT20 está com melhorias no protocolo, com a introdução de um novo pacote de dados de localização.

O protocolo pode enviar o comando 0x12 padrão antigo ou com 0x22 novo formato.

Para manter a configuração desejada, envie o comando conforme necessário:

SETLOCX12 # - Envia a localização da versão antiga do quadro 0x12

SETLOCX22 # - Envia localização com novo quadro 0x22

Este novo frame 0x22, contém informações como, número de identificação do terminal, data e hora do GPS, data e hora da mensagem, odômetro (quilometragem), horímetro (futuro), valores de tensão da alimentação principal e bateria interna, entre outras. Ver detalhes no item 5.5 no manual do protocolo pag 21.

5) Envio de comandos

- O envio de comando é feito através de pacotes do tipo 0x80, ver formato na Tabela 2.
- Tem a seguinte formatação:

Tabela 2 – Formato Comando

Format		Length (Byte)
Start Bit		2
Packet length		1
Protocol Number 0x80		1
Information Content	Length of Command	1
	Server Flag Bit	4
	Command Content	M
Information Serial Number		2
Error Check		2
Stop Bit		2

- M é o comando propriamente. Por exemplo, a requisição de uma localização é feita através do comando WHERE#.
- O tamanho do comando inclui: *Length of Command = Server Flag Bit + Length of Command Content + Information Serial Number*,
- O rastreador responde se o comando foi aceito ou não.

X3 Tech – Tecnologia em Rastreamento
www.x3tech.com.br (11) 3197-4781 (11) 94514-2044

6) Exemplo do frame 0x22

7878-3c-22-01-0365119068397451-1307190c0d07-1307190c0d08-cb-028470ba-05078c1c-00-3837-09-02d4-0a-24c3-0001bb- 46-0514-2a-60-0002-000000-000000-002f-22b7-0d0a

	Format		Length (Byte)	Ref. Manual Protocolo	Exemplo
Information Content	Start Bit		2	5.5.1.1	7878
	Packet Length		1	5.5.1.2	3c
	Protocol Number		1	5.5.1.3	22
	Location Source Type		1	5.5.1.4	01
	Terminal ID		8	5.5.1.5	0365119068397451
	Internal Date Time (*)		6	5.5.1.6	13 07 19 0C 0D 07 (19-07-25 12:13:07)
	GPS Information	Data Time GPS (*)	6	5.5.1.7	13 07 19 0C 0D 08 (19-07-25 12:13:08)
		Quantity of GPS information satellites	1	5.5.1.8	CB (B=11 satélites fixos)
		Latitude	4	5.5.1.9	028470BA
		Longitude	4	5.5.1.10	05078C1C
		Speed	1	5.5.1.11	00
		Course, Status	2	5.5.1.12	3837
	LBS Information	LBS Length	1	5.5.1.13	09
		MCC	2	5.5.1.14	02D4
		MNC	1	5.5.1.15	0A
		LAC	2	5.5.1.16	24C3
		Cell ID	3	5.5.1.17	0001BB
	Status Information	Terminal Information Content	1	5.5.1.18	46
		Power Voltage	2	5.5.1.19	0514
		Battery Voltage	1	5.5.1.20	2A
		GSM Signal Strength	1	5.5.1.21	60
		Alarm/Language	2	5.5.1.22	0002
	Milleage		3	5.5.1.23	000000
	TotalHoursSum		3	5.5.1.24	000000
	Serial Number		2	5.5.1.25	002F
	Error Check		2	5.5.1.26	22B7
	Stop Bit		2	5.5.1.27	0D0A

(*) O *Internal Data Time* é a data e hora em que o pacote foi gerado, e o *Data Time GPS* é a data e hora que a localização do GPS foi gerada.

ATENÇÂO!

É importante ressaltar que caso o rastreador esteja em área de sombra do GPS (garagens subterrâneas, túnel, etc.) o pacote será gerado com a *GPS Information* incluindo a *Data Time GPS* da última posição válida, e o *Internal Data Time*, do momento em que o pacote foi gerado.

Se o rastreador estiver em área de “sombra” de sinal GPRS sem comunicação como servidor, os dados de localização e alarmes serão armazenados e enviados posteriormente.

Os pacotes armazenados serão enviados em ordem do mais recente para o mais antigo, intercalado com uma nova posição. Portanto é aconselhável usar o *Internal Data Time* para organizar a sequência dos pacotes na visualização.

7) Exemplo de código em C# frame 0x22

Exemplo de código apenas para referencia, os dados

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApp1
{
    class Program
    {
        static void Main(string[] args)
        {
            //Neste exemplo o frame string dados é um pacote 0x22 sem o 0x78,0x78
inicial.
            string dados = "<\"\x01\x03e\x11\x90h9tQ-
\x13\x07\x19\r\x1c\x0b\x13\x07\x19\r\x1c\r\xc9\x02\x84p\xe6\x05\x07\x8b\xcc\x009\
t\t\x02\xd4\n$\xc3\x00\x0c\'F\x05\x14)Q\x00\x02\x00\x00\x00\x00\x00\x00\x00\x8b\x
f4\x80\r\n";

            string auxiliar = "";

            for (int C = 0; C < dados.Length; C++)
            {
                auxiliar = auxiliar + "" +
Convert.ToByte(dados[C]).ToString("X2");
            }
            Console.WriteLine(auxiliar);

            //auxiliar.Substring(2, 2) == 22

            // Location Data do tipo 0x22
            //3C 22 01 0365118127604386
130709031F02130709031F02CA028470B805078AC80038C60902D40A24C3000C274604A12A4C00020
0000000000001656537

            int LOCATION_TYPE = System.Convert.ToInt32(auxiliar.Substring(4, 2),
16);
            Console.WriteLine(LOCATION_TYPE);

            //0365118127604386    Terminal ID                6,16 Não tratado

            //130709031F02      Internal Date Time
            string INTERNAL_DATA = "20" +
Convert.ToByte(System.Convert.ToUInt32(auxiliar.Substring(22, 2), 16)).ToString()
            + Convert.ToByte(System.Convert.ToUInt32(auxiliar.Substring(24, 2),
16)).ToString().PadLeft(2, '0')
            + Convert.ToByte(System.Convert.ToUInt32(auxiliar.Substring(26, 2),
16)).ToString().PadLeft(2, '0') + " "
            + Convert.ToByte(System.Convert.ToUInt32(auxiliar.Substring(28, 2),
16)).ToString() + ":"
            + Convert.ToByte(System.Convert.ToUInt32(auxiliar.Substring(30, 2),
16)).ToString().ToString().PadLeft(2, '0') + ":"
            + Convert.ToByte(System.Convert.ToUInt32(auxiliar.Substring(32, 2),
16)).ToString().ToString().PadLeft(2, '0');

            Console.WriteLine(INTERNAL_DATA);

            //130709031F02        Data Time GPS
```

```
            string GPSD_DATABR = "20" +
Convert.ToByte(System.Convert.ToUInt32(auxiliar.Substring(34, 2), 16)).ToString()
            + Convert.ToByte(System.Convert.ToUInt32(auxiliar.Substring(36, 2),
16)).ToString().PadLeft(2, '0')
            + Convert.ToByte(System.Convert.ToUInt32(auxiliar.Substring(38, 2),
16)).ToString().PadLeft(2, '0')
            + " " + Convert.ToByte(System.Convert.ToUInt32(auxiliar.Substring(40,
2), 16)).ToString() + ":"
            + Convert.ToByte(System.Convert.ToUInt32(auxiliar.Substring(42, 2),
16)).ToString().ToString().PadLeft(2, '0') + ":"
            + Convert.ToByte(System.Convert.ToUInt32(auxiliar.Substring(44, 2),
16)).ToString().ToString().PadLeft(2, '0');

            Console.WriteLine(GPSD_DATABR);

            float dirLat = -1;
            float dirLon = -1;


            int auxVal = System.Convert.ToInt32(auxiliar.Substring(66, 4), 16) &
0x800; //Bit3
            if (auxVal == 0)
                dirLon = 1;

            auxVal = System.Convert.ToInt32(auxiliar.Substring(66, 4), 16) &
0x400; //Bit2
            if (auxVal != 0)
                dirLat = 1;

            //CA                                Quantity of GPS
            float GPSD_NSATELITE = System.Convert.ToUInt32(auxiliar.Substring(46,
2), 16) & 0x0F; ;

            //028470B8                    Latitude
            float GPSD_LATITUDE =
((float)System.Convert.ToUInt32(auxiliar.Substring(48, 8), 16) / 30000 / 60 *
dirLat);

            //05078AC8                    Longitude
            float GPSD_LONGITUDE =
((float)System.Convert.ToUInt32(auxiliar.Substring(56, 8), 16) / 30000 / 60 *
dirLon);

            //00                                Speed
            float GPSD_VELOCIDADE =
System.Convert.ToUInt32(auxiliar.Substring(64, 2), 16);

            //38C6                        Course, Status
            float GPSD_DIRECAO = System.Convert.ToUInt32(auxiliar.Substring(66,
4), 16) & 0x3FF;
            float GPSD_COMU = System.Convert.ToUInt32(auxiliar.Substring(66, 4),
16) & 0x1000;

            float Real_time_GPS = System.Convert.ToUInt32(auxiliar.Substring(66,
4), 16) & 0x2000;


            //02D4                        MCC
            float MCC = System.Convert.ToUInt32(auxiliar.Substring(72, 4), 16);

            //0A                                MNC
```

```
            float MNC = System.Convert.ToUInt32(auxiliar.Substring(76, 2), 16);

            //24C3 LAC
            float LAC = System.Convert.ToUInt32(auxiliar.Substring(78, 4), 16);

            //000C27                    Cell ID
            float CellIDE = System.Convert.ToUInt32(auxiliar.Substring(82, 6),
16);

            //46                                Term Inf Content    88,2
            float ALARME = System.Convert.ToInt32(auxiliar.Substring(88, 2), 16)
& 0x38;
            float ACC_HIGH = System.Convert.ToInt32(auxiliar.Substring(88, 2),
16) & 0x02;
            float CHARGE_ON = System.Convert.ToInt32(auxiliar.Substring(88, 2),
16) & 0x04;
            float BLOQUEIO = System.Convert.ToInt32(auxiliar.Substring(88, 2),
16) & 0x80;

            //04A1                      Power Voltage             90,4
            float Power_Voltage =
(float)System.Convert.ToInt32(auxiliar.Substring(90, 4), 16) / 100;

            //2A                                Battery Voltage           94,2
            float Battery_Voltage =
(float)System.Convert.ToInt32(auxiliar.Substring(94, 2), 16) / 10;

            //4C                                GSM Signal Strength 96,2
            float GSM_Signal = System.Convert.ToInt32(auxiliar.Substring(96, 2),
16);

            //0002                      Alarm/Language            98,4

            //000000                    Milleage                  102,6
            float Milleage = System.Convert.ToInt32(auxiliar.Substring(102, 6),
16);

            //000000                    TotalHoursSum       108,6
            float TotalHoursSum = System.Convert.ToInt32(auxiliar.Substring(108,
6), 16);

            //0165                      Serial Number       114,4
        }
    }
}
```

Documentação de referência necessária para implementação:

- Manual do protocolo:

  NT20 Protocol.pdf

- Descrição dos comandos:

  NT20 Command List.pdf

- Lista de comandos SMS, GPRS e SSCOM:

  Manual basico NEW TRACKER NT20 - X3Tech_rev2.pdf