PSIU - Protocolo Simples de Intercomunicação Unificado

March 6, 2012

Introdução

Um protocolo de comunicação nada mais é do que um conjunto de convenções que rege o tratamento e, especialmente, a formatação dos dados num sistema de comunicação. Seria a "gramática" de uma "linguagem" de comunicação padronizada. Conhecemos vários protocolos de comunicação e fazemos uso deles diariamente, mas não pensamos neles como protocolos de comunicação. O mais antigo deles é a língua falada: duas pessoas que emitem sons audíveis aos ouvidos humanos podem se comunicar. Neste exemplo, o protocolo de comunicação é a emissão de sons numa dada faixa de frequência, o código utilizado é a língua falada e a mensagem é o conteúdo do que se fala. Conforme o site Informática na Aldeia [1],

Na robótica recorremos o tempo todo a protocolos de comunicação para enviar comandos à robôs, obter informação de sensores, efetuar telemetria de dados, etc. De uma forma não unificada, na maioria dos casos, cada projetista cria seus próprios comandos e desenvolvem programas para controlar seus robôs a partir de computadores, tabletes ou outros dispositivos ou mesmos para robôs trocar informações entre si. Como esses comandos não são normatizados, eles acabam ficando limitados ao controle de poucos robôs desenvolvidos pela mesma pessoa ou empresa e não permite que os robôs se comuniquem entre si.

A proposta do PSIU é normatizar comandos, informações e formas de comunicação para que a conversa ocorra entre dispositivos independente dos programas de controle utilizados.

Entre inúmeros protocolos de comunicação que já foram criados, o modelo OSI é universalmente adotado e utilizado inclusive no TCP/IP (protocolo da Internet). De forma a ilustrar a abrangência do PSIU, ele representaria uma única camada do modelo OSI, a de aplicação. Isso quer dizer que o protocolo PSIU não normaliza as camadas mais baixas que definem o canal de comunicação e a forma que a mensagem chega até o destinatário. Ele apenas define palavras formadas em um padrão já consolidado, chamado ASCII, e essas palavras podem ser traduzidas em ações ou informações para robôs independente de sua arquitetura.

Sendo assim o protocolo PSIU pode ser utilizado em um canal serial (UART232, comN) consolidado no Sistema operacional de um PC, por exemplo, a partir de um driver específico (FTDI) e com comunicação por cabo. Como pode utilizar um driver similar consolidado através de uma comunicação bluetooth. Os mesmos comandos também podem trafegar em textos dentro de mensagens HTML ou sobre qualquer outro protocolo. O que o PSIU vem normatizar é o padrão em que o texto dos comandos são formatados.

Para garantir o envio de comandos em dispositivos com processamento limitado também existe a versão simplificada do PSIU que descreve comandos a partir de códigos hexadecimais de 8 bits sem a utilização da codificação ASCII.

[1] Informática na Aldeia, http://www.numaboa.com.br/informatica/internet/526-protocolos, acessado em fevereiro 2012

PSIU ASCII

O PSIU procura definir uma forma de comunicação para diferentes tipos de aplicação e volume de dados e diferentes topologias. Todas os frames (mensagens de comandos ou de respostas) enviadas no PSIU ASCII são escritos em caractere ASCII. Um pacote básico de informação neste protocolo é mostrado abaixo:

Destinatário	Tamanho	Mensagem	Remetente	CheckSum

O campo destinatário é o nome de identificação contido em cada robô. O próprio robô conhece seu nome e portanto sabe se o destinatário está correto ou não.
O tamanho é o tamanho do pacote, ou seja, a quantidade total de bytes contido na mensagem.
O campo mensagem será ou um comando ou uma resposta. Se o comando/resposta tiver parâmetros eles devem estar na mensagem, caso contrário, será mandada uma resposta de erro.
CheckSum é a soma de todos os bytes da mensagem.

Um exemplo de uma mensagem de comando no PSIU está descrita abaixo:

MNERIM031parafrente 100 PC02024

Onde: Destinatário = MNERIM , Tamanho = 029 , Comando = parafrente, Parâmetro = 100 , Remetente = PC, CheckSum = 02031

Campos do pacote

Destinatário

Destinatário é o campo onde vai conter o nome do dispositivo a ser enviado a mensagem. Este nome pode conter letra maísculas, minúsculas, numerés, espaços e caracteres especiais, desde que sejam caracteres ASCII.
Cada dispositivo sabe seu nome e com isso ele só irá ler as mensagens que sejam mandadas diretamente para eles.
A quantidade de bytes é definido pelo tamanho do nome do dispositivo.
O número de caracteres do nome também pode variar entre os dispositivos, mas como cada um sabe seu próprio nome é possível indicar onde começa o próximo campo (tamanho da mensagem).

Tamanho

Tamanho é um campo de 3 bytes fixo que vai conter o tamanho total de bytes da mensagem. Sendo assim, o tamanho varia de 0 até 999. Se uma mensagem contém apenas dezenas de bytes, o tamanho de 3 bytes se mantem sempre fixo e no campo da centena será colocado um 0.

Exemplo: 033 (Mensagem de 33 bytes)

Mensagem

O campo mensagem pode ser um comando ou uma resposta, incluido com um novo campo chamado parâmetro, que existirá se o comando ou resposta conter parâmetros.

Comando

O campo comando vai conter o nome do comando a ser executado pelo dispositivo. Propriamente dito, é uma palavra chave que deve indicar intuitivamente a ação a ser tomada pelo destinatário.
No final de cada string de nome terá que existir um espaço " " (0x20). De forma ao despositivo determinar o final de um comando.

Resposta

O campo resposta é uma mensagem de retorno definida a um comando executado.
No final de cada string de resposta terá que existir um espaço " " (0x20). De forma ao despositivo determinar o final de uma resposta.

Parâmetro

Parâmetro não é um campo obrigatório. Ele depende se o comando contém parâmetros ou não. Caso um comando ou resposta não possua parâmetro este campo deve ser deixado em branco. Um parâmetro assim como o comando, deve conter no final um espaço em branco para determinar o final do parâmetro. Se o comando tiver dois ou mais parâmetros a separação entre eles é dada por espaço " " (0x20). Caracteres do tipo float têm a parte inteira separada da fracionária por ponto (1.25)

CheckSum

CheckSum é um campo de 5 bytes fixos de tamanho e é a soma em uma variável inteira de 16 bits (desprezando o transporte "vai um") de todos os bytes da mensagem (ele não incluido). Essa soma é representada em decimal. Como os 5 bytes são fixos, se a soma dos bytes der um número de 4 bytes, deverá ser acrescentado um 0 na frente, assim formando um CheckSum de 5 bytes.

Exemplo:

```
int checksum = 0;
char mensagem[] = "MNERIM029parafrente 100 ";
for (x = 0; x < strlen(mensagem) ; x++)
      checksum = checksum + mensagem[x];
printf("%s%d", mensagem, checksum);
```

Respostas

Para um controle mais eficaz nas respostas dos comandos no protocolo PSIU ASCII, antes de cada resposta (sucesso, falha, distância calculada por um sensor, etc) será mandado também o comando que foi executado.

Dest.	Tam.	ComandoExecutado	Resposta	Remetente	CheckSum

Isso facilita o entendimento das respostas pelos dispositivos.

Exemplo:

Envio do Comando "parafrente":

```
MNERIM029parafrente 100 PC02031
```

Resposta (Sucesso):

```
PC035parafrente sucesso MNERIM02656
```

Respostas PSIU ASCII

sucesso

Uma mensagem de sucesso é enviada se um comando válido foi executado.
Exemplo:

Envio da resposta:

```
PCXXX"nomedocomando" sucesso MNERIMXXXXX
```

falha

Uma mensagem de falha é enviada se a execução se um comando falhou. Exemplo:

Envio da resposta:

```
PCXXX"nomedocomando" falha MNERIMXXXXX
```

Comandos

parafrente

O comando *parafrente* possui um parâmetro inteiro que varia de 0 a 999. Este comando diz ao robô a quantidade em *cm* que ele irá andar para frente. Se o comando for executado corretamente será recebido uma resposta de OK, caso contrário será recebido uma resposta de FALHA.

Exemplo:

Envio do Comando:

```
MNERIM031parafrente 100 PC02024
```

MNERIM	031	parafrente	100	PC	02024
Destinatário	Tam.	Comando	Parâmetro	Rem.	CheckSum

Resposta:

```
PC035parafrente sucesso MNERIM02656
```
(SUCESSO)

PC	035	parafrente sucesso	MNERIM	02656
Dest.	Tam.	Resposta	Rem.	CheckSum

```
PC033parafrente falha MNERIM02389
```
(FALHA)

PC	033	parafrente falha	MNERIM	02389
Dest.	Tam.	Resposta	Rem.	CheckSum

paratras

O comando *paratras* possui um parâmetro inteiro que varia de 0 a 999. Este comando diz ao robô a quantidade em *cm* que ele irá andar para trás.
Se o comando for executado corretamente será recebido uma resposta de OK, caso contrário será recebido uma resposta de FALHA.

Exemplo:

Envio do Comando:

```
MNERIM029paratras 100 PC01829
```

MNERIM	029	paratras	100	PC	01829
Destinatário	Tam.	Comando	Parâmetro	Rem.	CheckSum

Resposta:

```
PC033paratras sucesso MNERIM02452
```
(SUCESSO)

PC	033	paratras sucesso	MNERIM	02452
Dest.	Tam.	Resposta	Rem.	CheckSum

```
PC031paratras falha MNERIM02185
```
(FALHA)

PC	031	paratras falha	MNERIM	02185
Dest.	Tam.	Resposta	Rem.	CheckSum

giradireita

O comando *giradireita* possui um parâmetro inteiro que varia de 0 a 360. Este comando diz ao robô a quantidade em *graus* que ele irá girar para direita.
Se o comando for executado corretamente será recebido uma resposta de OK, caso contrário será recebido uma resposta de FALHA.

Exemplo:

Envio do Comando:

```
MNERIM032giradireita 100 PC02118
```

MNERIM	032	giradireita	100	PC	02118
Destinatário	Tam.	Comando	Parâmetro	Rem.	CheckSum

Resposta:

```
PC036giradireita sucesso MNERIM02750
```
(SUCESSO)

PC	036	giradireita sucesso	MNERIM	02750
Dest.	Tam.	Resposta	Rem.	CheckSum

```
PC034giradireita falha MNERIM02483
```
(FALHA)

PC	034	giradireita falha	MNERIM	02483
Dest.	Tam.	Resposta	Rem.	CheckSum

giraesquerda

O comando *giraesquerda* possui um parâmetro inteiro que varia de 0 a 360. Este comando diz ao robô a quantidade em *graus* que ele irá girar para esquerda.
Se o comando for executado corretamente será recebido uma resposta de OK, caso contrário será recebido uma resposta de FALHA.

Exemplo:

Envio do Comando:

```
MNERIM033giraesquerda 100 PC02239
```

MNERIM	033	giraesquerda	100	PC	02239
Destinatário	Tam.	Comando	Parâmetro	Rem.	CheckSum

Resposta:

```
PC037giraesquerda sucesso MNERIM02871
```
(SUCESSO)

PC	037	giraesquerda sucesso	MNERIM	02871
Dest.	Tam.	Resposta	Rem.	CheckSum

```
PC035giraesquerda falha MNERIM02604
```
(FALHA)

PC	035	giraesquerda falha	MNERIM	02604
Dest.	Tam.	Resposta	Rem.	CheckSum

quantoscomandos

O comando *quantoscomandos* pergunta ao dispositivo a quantidade de funções disponíveis. O retorno dessa função é um número com representando a quantidade de comandos do dispositivo.

Exemplo:

Envio do Comando:

```
MNERIM032quantoscomandos PC02415
```

MNERIM	032	quantoscomandos	PC	02415
Destinatário	Tam.	Comando	Rem.	CheckSum

Resposta:

```
PC032quantoscomandos 6 MNERIM02501
```

PC	032	quantoscomandos 6	MNERIM	02501
Dest.	Tam.	Resposta	Rem.	CheckSum

exibecomandos

O comando *exibecomandos* pede ao dispositivo para exibir o comando desejado. Essa função tem como parâmetro um numero inteiro.
O retorno dessa função é o nome do comando seguido da quantidade de parâmetros inteiro, float e char que a função recebe.

Exemplo:

Envio do Comando:

```
MNERIM031exibecomando 1 PC02126
```

MNERIM	031	exibecomandos	PC	02126
Destinatário	Tam.	Comando	Rem.	CheckSum

Resposta:

```
PC044exibecomando parafrente 1 0 0 MNERIM03386
```

PC	038	exibecomando parafrente 1 0 0	MNERIM	03148
Dest.	Tam.	Resposta	Rem.	CheckSum

Comandos de Controle

qualseunome

O comando *qualseunome* pergunta ao dispositivo o seu nome e recebe como resposta o nome do dispositivo.
Se houver interesse em se comunicar com o dispositivo e caso não se saiba o nome do dispositivo usa-se esse comando.
No campo *DESTINATÁRIO* no lugar do nome irá ter duas interrogações "??" (0x3F3F).

Exemplo:

Envio do Comando:

```
??024qualseunome PC01654
```

??	024	qualseunome	PC	01654
Destinatário	Tam.	Comando	Rem.	CheckSum

Resposta:

```
PC028qualseunome MNERIM01988
```

PC	028	qualseunome	MNERIM	01988
Dest.	Tam.	Resposta	Rem.	CheckSum

Table 1: Comandos do PSIU

Comando	Parâmetro	Exemplo de comando
qualseunome	–	??024qualseunome PC01654
quantoscomandos	–	PC032quantoscomandos 7 MNERIM02502
exibecomandos	1 int	PC038exibecomando parafrente MNERIM03148
parafrente	1 int (0 : 999)	MNERIM031parafrente 100 PC02024
paratras	1 int (0 : 999)	MNERIM029paratras 100 PC01829
giradireita	1 int (0 : 360)	MNERIM032giradireita 100 PC02118
giraesquerda	1 int (0 : 360)	MNERIM033giraesquerda 100 PC02239

PSIU HEXA

Um pacote básico de informação neste protocolo é mostrado abaixo, onde os campos remetente, destinatário, tamanho e CRC são de 1 byte cada, o que facilita a implementação deste protocolo por dispositivos pouco complexos.

Remetente	destinatário	tamanho	carga	CRC

Os campos de remetente e destinatário dependem dos níveis mais baixos da comunicação e da topologia da rede: Uma comunicação ponto-a-ponto não requer nenhum dos 2 campos e portanto começaria a partir do tamanho. Uma rede com um mestre e vários escravos precisa apenas do endereço do escravo (seja como remetente ou destinatário). Numa rede ad-hoc ambos os campos são necessários.

O tamanho é o tamanho do pacote, ou seja: número de bytes de carga +4. Como o tamanho é um byte a carga pode ser de 0 até 251 bytes dependendo do tipo de pacote.

1 Tipos de pacote

Tipo	tamanho (bytes)	carga (bytes)
ping	4	0
comando	5	1
nó	6	2
complexo	7 a 255	3 a 251

Um pacote do tipo ping não tem carga nenhuma. Normalmente é usado para saber se o dispositivo está vivo: A manda um ping para B e B responde com um ping para A. Para os dispositivos mais burros, o tamanho é sempre 4, 5 ou 6. Apenas dispositivos com maior capacidade de processamento reconhecem um pacote completo.

1.1 Comando

Um pacote do tipo comando manda apenas 1 byte de informação. É usado para mensagens do tipo ACK, NAK, BUSY e AVAILABLE.

Uma possibilidade de valores de carga padrão para este tipo de pacote é:

carga	comando	Significado
65 (ascii A)	ACK	Último pacote foi recebido corretamente.
78 (ascii N)	NAK	Erro no último pacote (erro de CRC). Reenvie.
66 (ascii B)	BUSY	Não consegui processar o último pacote. Não reenvie pois estou ocupado.
63 (ascii ?)	AVAIL?	Pergunta: pronto para receber um pacote?
70 (ascii F)	FREE	Resposta: pronto para receber um pacote. (Se não, manda um BUSY)
87 (ascii W)	WTF	Último comando é impossível de ser executado (CRC não deu erro e não está ocupado, mas o valor de carga não faz sentido).

1.2 Nó

No pacote tipo nó, um byte de informação é enviado a um nó específico do dispositivo. Cada nó é um atuador, um sensor ou um elemento mais abstrato mas que é modelado como um atuador ou um sensor. O valor 0 no campo nó é reservado para a resposta de sensores.

Remetente	destinatário	tamanho	nó	sinal	CRC

1.2.1 Atuadores e sensores

Nós atuadores recebem um sinal e respondem com com um ACK.

tipo do nó	sinal
DC aberto	Valor de um PWM que controla o motor DC
DC controlado	Velocidade do motor. 255 (ou 127) é a máxima vel. definida
stepper	Número de passos
servo	-128: -90o. 127: +90o
linear	valor entre 0 (mínimo) e 255 (máximo) de atuador linear
on-off	0 (desligado) ou 255 (ligado)

Quando o nó é um sensor, o campo sinal não importa. Na resposta o campo nó recebe valor 0 como indicador de que é uma resposta a um pedido de informação. Se um dispositivo recebe um pacote do tipo nó com nó igual a 0 sem ter pedido informação a nenhum outro dispositivo, ele deve responder com um WTF.

tipo do nó	sinal
Luminosidade	
ângulo	
distância (sonar)	
velocidade	
aceleração	
on-off	

1.3 Complexo

Para dispositivos inteligentes ou muitos bytes de dados, um pacote pode ter vários bytes. Cada dispositivo sabe o tamanho do seu nome, logo vai considera o campo destinatário como sendo deste tamanho.

O tamanho máximo é 255 bytes. Se porventura houver uma mensagem maior, ela deverá ser quebrada em mensagens de 255 bytes.

rem.	dest.	tamanho	nCampos	tam. campo 1	...	tam. campo n	payload	CRC

Rementente é o robô que enviou a mensagem e destinatário é para quem é a mensagem. O campo de remetente é importante pois o destinatário não precisa obedecer a todo robô. Um byte permite endereçar 256 robôs. Os nós são dispositivos de um robô: motores, sensores, etc. Em função dele o sinal pode significar diferentes coisas.

1.3.1 Atuadores

tipo do nó	sinal
DC aberto	Valor de um PWM que controla o motor DC
DC controlado	Velocidade do motor. 255 (ou 127) é a máxima vel. definida
stepper	Número de passos
servo	-128: -90o. 127: +90o
linear	valor entre 0 (mínimo) e 255 (máximo) de atuador linear
on-off	0 (desligado) ou 255 (ligado)

1.3.2 Sensores

Os sensores envolvem 2 sinais: um pedindo informação do sensor, com apenas os endereços e o nó, e o outro enviando esta informação, com o sinal relevante. Neste caso pode-se colocar um bit do nó como indicador se é um pedido ou uma resposta.

tipo do nó	sinal
Luminosidade	
ângulo	
distância (sonar)	
velocidade	
aceleração	
on-off	