Apêndice B

Programar com o CARMEN

Este anexo descreverá os passos básico para se criar um novo módulo utilizando CARMEN. O documento de referência utilizado, contendo mais informações sobre estilos de programação e padrões a seguir pode ser encontrado em `http://carmen.sourceforge.net/program_carmen.html`. Lembre-se sempre que você não será a única pessoa que irá ler, entender e modificar o seu código.

B.1 Introdução

Nunca inicialize o IPC você mesmo, sempre utilize `carmen_ipc_initialize`. Para escrever um novo módulo CARMEN, deverão ser criados três ficheiros:

- `{module}_messages.h` - contendo as definições da mensagem IPC.
- `{module}_interface.h` - contendo os protótipos das funções para a comunicação com o novo módulo.
- `{module}_interface.c` - contendo as funções para a comunicação com o módulo. Este ficheiro deve ser compilado em uma biblioteca.

A seguir será mostrado um módulo criado para a comunicação de baixo nível com o robô Atlas, e que servirá de exemplo para a criação destes três ficheiros. É importante salientar que este módulo não segue a padronização exigida pelo CARMEN no que diz respeito às unidades e aos tipos de mensagem, mas ainda assim serve como uma referência.

B.2 Ficheiro *messages.h*

A Figura B.1 mostra o modelo de código para o ficheiro que contém as definições da mensagem IPC. Onde lê-se `{module}` deverá ser substituído pelo nome do novo módulo, seguindo o padrão de letras minúsculas e maiúsculas de acordo com o modelo. Onde está escrito `{modulo_msg}` deverá ser escrito o nome do novo módulo mais o nome da mensagem específica.

Um módulo pode conter uma série de mensagens, e este último parâmetro é que fará a distinção entre elas. Como exemplo o nome do módulo pode ser `ATLAS` e o nome da mensagem `DIR_AND_SPEED`, sendo que o nome da estrutura da mensagem resultaria `carmen_atlas_dir_and_speed_message`.

```
#ifndef CARMEN_{MODULE}_MESSAGES_H
#define CARMEN_{MODULE}_MESSAGES_H

#ifdef __cplusplus
extern "C" {
#endif

// These messages are generic format messages

#define CARMEN_{MODULE}_RESET_OCCURRED_NAME "carmen_{module}_reset_occurred"
typedef carmen_default_message carmen_{module}_reset_occurred_message;
#define CARMEN_{MODULE}_RESET_COMMAND_NAME "carmen_{module}_reset_command"
typedef carmen_default_message carmen_{module}_reset_command_message;


typedef struct {
  int dir;
  int speed;
  int comport;
  double timestamp;
  char *host;
} carmen_{module_msg}_message;

#define      CARMEN_{MODULE_MSG}_NAME       "carmen_{module_msg}"
#define      CARMEN_{MODULE_MSG}_FMT        "{int,int,int,double,string}"

#ifdef __cplusplus
}
#endif

#endif
```

Figura B.1: Código modelo para o ficheiro *messages.h*

Na estrutura da mensagem em si, é obrigatório a criação dos campos *timestamp* e *host*, seguindo a formatação do exemplo. Os outros campos da mensagem podem ser criados de acordo com a necessidade. Neste caso existe um campo para enviar a informação de direção, outro de velocidade e outro que especifica a porta de comunicação. Os tipos destes dados devem constar também na última linha *DEFINE* de cada mensagem criada.

B.3 Ficheiro *interface.h*

Este ficheiro contém o protótipo das funções para comunicação com o novo módulo. Novamente onde lê-se `{module}` deverá ser substituído pelo nome do

novo módulo, sempre de acordo com o ficheiro anterior. Neste exemplo o nome da função protótipo pode ser qualquer um, mas convém utilizar um padrão conforme o apresentado, onde consta o nome do novo módulo e o nome da mensagem. Assim o nome completo da função seria: `carmen_atlas_subscribe_dir_and_speed_message`.

```
#ifndef CARMEN_{MODULE}_INTERFACE_H
#define CARMEN_{MODULE}_INTERFACE_H

#include <carmen/{module}_messages.h>


#ifdef __cplusplus
extern "C" {
#endif

void carmen_{module}_subscribe_{msg}_message(carmen_{module_msg}_message
                                             *dir_and_speed,
                                             carmen_handler_t handler,
                                             carmen_subscribe_t subscribe_how);

void carmen_{module}_reset(void);
#ifdef __cplusplus
}
#endif

#endif
```

Figura B.2: Código modelo para o ficheiro *interface.h*

B.4 Ficheiro *interface.c*

Este ficheiro deve ser compilado em uma biblioteca, e contém as funções para comunicação com o módulo em criação. A nomeclatura deverá seguir o estabelecido nas subseções anteriores.

```
#include <carmen/global.h>
#include <carmen/ipc_wrapper.h>
#include <carmen/{module}_messages.h>


void
carmen_{module}_subscribe_{msg}_message(carmen_{module_msg}_message *{msg},
                                        carmen_handler_t handler,
                                        carmen_subscribe_t subscribe_how)
{
  carmen_subscribe_message(CARMEN_{MODULE_MSG}_NAME,
                           CARMEN_{MODULE_MSG}_FMT,
                           {module_msg}, sizeof(carmen_{module_msg}_message),
                           handler, subscribe_how);
}
```

Figura B.3: Código modelo para o ficheiro *interface.c*

B.5 Compilação

Estes três ficheiros devem ser colocados dentro de uma pasta que por sua vez está dentro do diretório `/src` do CARMEN. Para a compilação mais um ficheiro deve ser criado (Figura B.6), contendo as especificações da compilação. Novamente o exemplo deve ter os ficheiros renomeados para coincidirem com os da criação do módulo. Contendo estes quatro ficheiros dentro de um diretório localizado dentro da pasta `/src` do CARMEN, basta executar o comando `make`.

```
include ../Makefile.conf

CFLAGS +=
IFLAGS +=
LFLAGS +=

MODULE_NAME = EXEMPLO
MODULE_COMMENT = Interface for exemplo 2008

SOURCES = exemplo_interface.c
PUBLIC_INCLUDES = exemplo_messages.h exemplo_interface.h
PUBLIC_LIBRARIES = libexemplo_interface.a

TARGETS = libexemplo_interface.a

PUBLIC_LIBRARIES_SO = libexemplo_interface.so
ifndef NO_PYTHON
TARGETS += libexemplo_interface.so.1
endif

# rules

libexemplo_interface.a: exemplo_interface.o

libexemplo_interface.so.1:    exemplo_interface.o

include ../Makefile.rules
```

Figura B.4: Código modelo para o ficheiro *Makefile* do novo módulo

Para integrar totalmente este módulo com o CARMEN, deve-se aceder ao diretório `/src` e editar o ficheiro *Makefile*, este é um ficheiro geral que fará a compilação de todos os módulos do CARMEN. Dentro deste ficheiro está a lista dos módulos a serem compilados onde lê-se PACKAGES. Nesta lista deve ser incluído o diretório criado para abrigar os ficheiros do novo módulo.

Por exemplo, dentro do diretório `/src` cria-se um novo diretório chamado **atlas** e dentro deste são colocados os ficheiros acima descritos, la lista de **PACKAGES** acrescenta-se **ATLAS** no fim da lista. Para fins de testes, pode-se executar o comando **make** dentro de `..carmen/src/` e procurar na lista dos módulos compilados o novo módulo.

Finalmente, deve-se aceder ao diretório `..carmen/include/carmen` e abrir o ficheiro `carmen.h`. Neste deve-se incluir a seguinte linha: `#include<carmen/{module}_interface.h>`, que é o nome do ficheiro *interface.h* criado anteriormente. Desta forma qualquer programa que incluir o ficheiro `carmen.h` terá acesso aos vários módulos nele incluídos.

B.6 Programas de Teste

Com o módulo criado e compilado, pode-se criar programas que utilizem os módulos criados. Estes programas podem tanto ser novos módulos dentro do CARMEN ou programas externos que apenas utilizam as bibliotecas e funcionalidades do CARMEN.

Neste exemplo, criaremos dois programas que irão se intercomunicar através do módulo criado. Um destes como componente do novo módulo criado no CARMEN e outro como uma aplicação externa que envia mensagens a este módulo através do IPC. A ferramenta utilizada na criação do aplicativo externo foi o *Kdevelop*.

B.6.1 Programa de Recepção

Imaginado-se que este novo módulo do CARMEN, para quais as funções de comunicação foram criadas, tem como objetivo uma comunicação de baixo nível com o robô, é normal que o programa que deve receber tais mensagens seja componente deste novo módulo. Assim poderá ser compilado sempre em conjunto com os outros módulos do CARMEN e poderá ser acessado por diferentes aplicativos externos ou internos que implementem a interface com este módulo.

O programa comentado é exibido na Figura B.5.

O último passo é incorporar as instruções para a compilação deste programa no **makefile** dentro do diretório do módulo, de acordo com a Figura B.6

B.6.2 Programa de Envio

Sugere-se criar um novo projeto utilzando o modelo *Qt3 Hello World*. Nas configurações do projeto deve-se inserir o caminho para os *headers* conforme mostra a Figura B.7, adaptando o caminho de acordo com sua instalação pessoal.

Para as bibliotecas, deve-se configurar conforme a Figura B.8. O CARMEN disponibiliza uma série de outras bibliotecas que tornam automáticas muitas operações, como conversões por exemplo. Aqui apenas as bibliotecas mais básicas são utilizadas, para maiores informações recomenda-se uma leitura mais aprofundada da documentação disponibilizada pelo CARMEN.

Finalmente, a Figura B.9 contém o código fonte do programa com os devidos comentários.

```c
#include <carmen/carmen.h>
//inclui os headers do carmen

carmen_exemplo_dirandspeed_message dirandspeed;
//declara uma variável para receber a mensagem

void exemplo_dirandspeed_handler(void)
//função invocada na recepção de uma mensagem
{
  printf("direction %d, speed %d \n",dirandspeed.dir,dirandspeed.speed);
}

void shutdown_module(int x)
//função para fechamento limpo do programa
{
  if(x == SIGINT) {
    carmen_ipc_disconnect();
    printf("Disconnected.\n");
    exit(1);
  }
}

int main(int argc, char **argv)
{
  carmen_ipc_initialize(argc, argv);
  //inicializa ipc
  carmen_param_check_version(argv[0]);
  //verifica versão do servidor de parâmetros

  signal(SIGINT, shutdown_module);

  carmen_exemplo_subscribe_dirandspeed_message(&dirandspeed, (carmen_handler_t)
                                               exemplo_dirandspeed_handler,
                                               CARMEN_SUBSCRIBE_LATEST);
  //subscrição da mensagem dir and speed

  while(1) {
    carmen_ipc_sleep(1);
  }
  return 0;
}
```

Figura B.5: Código do programa para receber mensagens

```
include ../Makefile.conf

CFLAGS +=
IFLAGS +=
LFLAGS += -lglobal -lipc -latlas_interface -lparam_interface

MODULE_NAME = EXEMPLO
MODULE_COMMENT = Interface for exemplo 2008

SOURCES = exemplo_interface.c exemplo_receptor.c
PUBLIC_INCLUDES = exemplo_messages.h exemplo_interface.h
PUBLIC_LIBRARIES = libexemplo_interface.a

TARGETS = libexemplo_interface.a exemplo_receptor

PUBLIC_LIBRARIES_SO = libexemplo_interface.so
ifndef NO_PYTHON
TARGETS += libexemplo_interface.so.1
endif

# rules

libexemplo_interface.a: exemplo_interface.o

libexemplo_interface.so.1:    exemplo_interface.o

exemplo_receptor:       exemplo_receptor.o libexemplo_interface.a

include ../Makefile.rules
```

Figura B.6: makefile para o programa de recepção e demais ficheiros do módulo

B.6.3 Testes

Para os testes deve-se abrir uma janela do terminal do linux com quatro abas, ou então quatro janelas separadas. A seguir acede-se ao diretório `..\carmen\bin` e executa-se o comando `./central`. Na aba ou janela seguinte, neste mesmo diretório executa-se `./param_daemon-rscout`. Estas duas instruções inicializam o servidor IPC e o servidor de parâmetros.

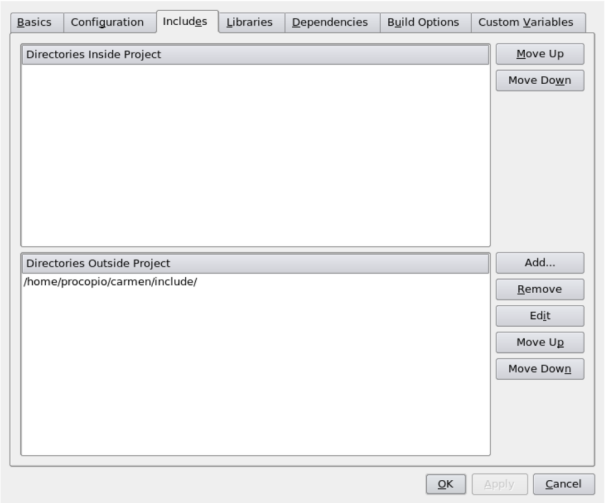

Figura B.7: Caminhos de diretórios que terão *headers* incluídos

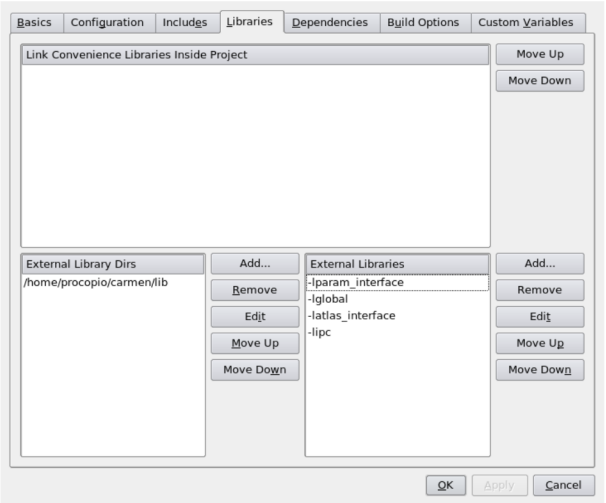

Figura B.8: Caminhos de diretórios e bibliotecas a serem incluídas

A seguir, em uma nova aba, entra-se no diretório `..\carmen\src\exemplo` (supondo que a pasta *exemplo* contém o novo módulo) e executa-se o comando `./exemplo_receptor`, este comando inicia o programa receptor. O último passo é executar o programa que recebe números do teclado e envia ao servidor IPC, para isso deve-se aceder ao diretório onde o programa foi criado e executá-lo.

Aqui conclui-se esta breve introdução sobre programação no CARMEN. Ainda existe muito a explorar.

```c
#ifdef HAVE_CONFIG_H
#include <config.h>
#endif

#include <carmen/carmen.h>
//inclusão dos headers

void shutdown_module(int x)
//função para desconexão limpa do programa
{
  if(x == SIGINT) {
    carmen_ipc_disconnect();
    printf("Disconnected.\n");
    exit(1);
  }
}

static void exemplo_dirandspeed_command(int dir, int speed)
//função que constrói e envia uma mensagem tipo dirandspeed
{
  IPC_RETURN_TYPE err;
  static carmen_exemplo_dirandspeed_message v;
  //variável com a estrutura da mensagem desejada

  v.dir = dir;
  v.speed = speed;
  v.comport = 0;
  v.host = carmen_get_host();
  v.timestamp = carmen_get_time();
  //armazenamento dos valores na mensagem

  err = IPC_publishData(CARMEN_EXEMPLO_DIRANDSPEED_NAME, &v);
  //publicação da mensagem
  carmen_test_ipc(err, "Could not publish", CARMEN_EXEMPLO_DIRANDSPEED_NAME);
  //teste de sucesso do envio
}

int main(int argc, char **argv)
{
  //conecta ao ipc
  carmen_ipc_initialize(argc, argv);
  //verifica versão do servidor de parâmetros
  carmen_param_check_version(argv[0]);

  signal(SIGINT, shutdown_module);

  while(1) {
    //recebe dois valores do teclado e invoca a função de envio
    int  direction, speed;
    printf("Type in direction 0<dir<90 and press Enter \n");
    scanf("%d", &direction);
    printf("Type in speed -1<speed<10 and press Enter\n");
    scanf("%d", &speed);
    printf("_____\n\n");
    exemplo_dirandspeed_command(direction,speed);
  }
  return 0;
}
```

Figura B.9: Código do programa para enviar mensagens