Track Sense

Generated by Doxygen 1.9.8

Chapter 1

Class Documentation

1.1 GPSSim Class Reference

Simulador do módulo GPS que gera frases no padrão NMEA.

`#include <GPSSim.hpp>`

Public Member Functions

- GPSSim (double latitude_inicial_graus, double longitude_inicial_graus, double altitude_metros=10.0, double frequencia_atualizacao_hz=1.0, double velocidade_nos=0.0)

  Construtor do GPSSim.
- ~GPSSim ()

  Destrutor do GPSSim.
- std::string obter_caminho_terminal_filho () const

  Obtém o caminho do terminal que estamos executando de forma filial.
- void init ()

  Inicia a geração de frases no padrão NMEA em uma thread separada.
- void stop ()

  Encerrará a geração de frases NMEA e aguardará a finalização da thread.
- void config_traj (double raio_metros=20.0, double periodo_segundos=120.0)

  Configura a trajetória circular para a simulação.
- void config_traj ()

  Configura a trajetória estática para simulação.

1.1.1 Detailed Description

Simulador do módulo GPS que gera frases no padrão NMEA.

Cria um par de pseudo-terminais (PTY) para simular um o módulo GPS real. Gera frases no padrão NMEA (RMC e GGA) em intervalos regulares, permitindo configuração de posição inicial, altitude, velocidade e padrão de movimento.

1.1.2 Constructor & Destructor Documentation

1.1.2.1 GPSSim()

```
GPSSim::GPSSim (
            double latitude_inicial_graus,
            double longitude_inicial_graus,
            double altitude_metros = 10.0,
            double frequencia_atualizacao_hz = 1.0,
            double velocidade_nos = 0.0 )  [inline], [explicit]
```

Construtor do GPSSim.

Inicializa alguns parâmetros de posição simulada e, criando os pseudo-terminais, configura-os para o padrão do módulo real.

Utilizou-se o termo `explict` para impedir que futuras conversões implicitas sejam impedidas. Além disso, foi pensado em utilizar o padrão de `Singleton` para manter apenas uma instância da classe. Entretanto, após outras reuniões, percebeu-se a falta de necessidade.

Parameters

latitude_inicial_graus	Latitude inicial em graus decimais
longitude_inicial_graus	Longitude inicial em graus decimais
altitude_metros	Altitude inicial em metros, setada para 10.
frequencia_atualizacao_hz	Frequência de atualização das frases NMEA em Hertz, setada para 1
velocidade_nos	Velocidade sobre o fundo em nós (para frase RMC), setada para 0

1.1.2.2 ~GPSSim()

```
GPSSim::~GPSSim ( )  [inline]
```

Destrutor do GPSSim.

Chama a função `stop()` e, após verificar existência de terminal Pai, fecha-o. Here is the call graph for this function:

1.1.3 Member Function Documentation

1.1.3.1 config_traj() [1/2]

```
void GPSSim::config_traj ( )  [inline]
```

Configura a trajetória estática para simulação.

1.1.3.2 config_traj() [2/2]

```
void GPSSim::config_traj (
            double raio_metros = 20.0,
            double periodo_segundos = 120.0 )  [inline]
```

Configura a trajetória circular para a simulação.

1.1.3.3 init()

```
void GPSSim::init ( )  [inline]
```

Inicia a geração de frases no padrão NMEA em uma thread separada.

Garante a criação de apenas uma thread utilizando uma variável atômica. O padrão NMEA é o protocolo padrão usado por módulos GPS, cada mensagem começa com $ e termina com \r
. Exemplo:

- $<origem><codificacao_usada>,<dados1>,<dados2>,...*<paridade>

1.1.3.4 obter_caminho_terminal_filho()

```
std::string GPSSim::obter_caminho_terminal_filho ( ) const  [inline]
```

Obtém o caminho do terminal que estamos executando de forma filial.

Returns

string correspondendo ao caminho do dispositivo.

1.1.3.5 stop()

```
void GPSSim::stop ( )  [inline]
```

Encerrará a geração de frases NMEA e aguardará a finalização da thread.

Verifica a execução da thread e encerra-a caso exista. Força a finalização da thread geradora de mensagens. Here is the caller graph for this function:

The documentation for this class was generated from the following file:

- src/GPSSim.hpp

Chapter 2

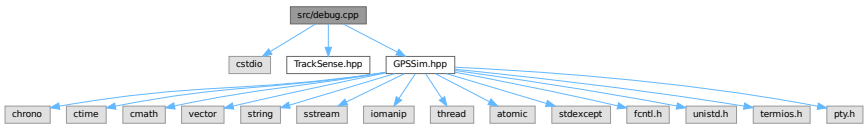
File Documentation

2.1 src/debug.cpp File Reference

Responsável por prover ferramentas de debug.

```
#include <cstdio>
#include "TrackSense.hpp"
#include "GPSSim.hpp"
```

Include dependency graph for debug.cpp:



Functions

- int main ()

2.1.1 Detailed Description

Responsável por prover ferramentas de debug.

Já que a aplicação deve ser executada dentro da placa, não conseguiríamos executá-la no DeskTop. Para tanto, fez-se necessário o desenvolvimento de ferramentas que possibilitam a debugação de nosso código.

2.1.2 Function Documentation

2.1.2.1 main()

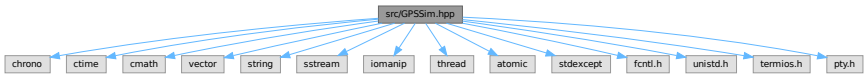
```
int main ( )
```

2.2 src/GPSSim.hpp File Reference

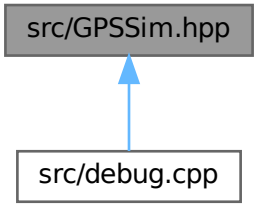
Implementação da Classe Simuladora do GPS6MV2.

```
#include <chrono>
#include <ctime>
#include <cmath>
#include <vector>
#include <string>
#include <sstream>
#include <iomanip>
#include <thread>
#include <atomic>
#include <stdexcept>
#include <fcntl.h>
#include <unistd.h>
#include <termios.h>
#include <pty.h>
```

Include dependency graph for GPSSim.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class GPSSim

  Simulador do módulo GPS que gera frases no padrão NMEA.

2.2.1 Detailed Description

Implementação da Classe Simuladora do GPS6MV2.

Supondo que o módulo GPS6MV2 não esteja disponível, a classe implementada neste arquivo tem como objetivo simular todas as funcionalidades do mesmo.

2.3 GPSSim.hpp

Go to the documentation of this file.

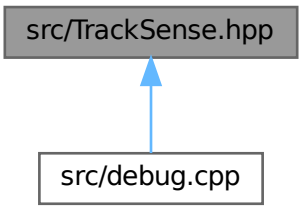
```
00001
00008 #ifndef GPSSim_HPP
00009 #define GPSSim_HPP
00010
00011 //------------------------------------------------------------
00012
00013 // Para manipulações de Tempo e de Data
00014 #include <chrono>
00015 #include <ctime>
00016
00017 #include <cmath>
00018 #include <vector>
00019
00020 #include <string>
00021 #include <sstream>
00022 #include <iomanip>
00023
00024 // Para threads e sincronizações
00025 #include <thread>
00026 #include <atomic>
00027
00028 // Para tratamento de erros
00029 #include <stdexcept>
00030
00031 // As seguintes bibliotecas possuem relevância superior
00032 // Por se tratarem de bibliotecas C, utilizaremos o padrão de `::` para explicitar
00033 // que algumas funções advém delas.
00034 /*
00035 Fornece constantes e funções de controle de descritores de arquivos,
00036 operações de I/O de baixo nível e manipulação de flags de arquivos.
00037 */
00038 #include <fcntl.h>
00039 /*
00040 Fornece acesso a chamadas do OS de baixo nível, incluindo manipulação
00041 de processos, I/O de arquivos, controle de descritores e operações do
00042 sistema de arquivos.
00043 */
00044 #include <unistd.h>
00045 /*
00046 Fornece estruturas e funções para configurar a comunicação
00047 em sistemas Unix. Ele permite o controle detalhado sobre interfaces
00048 de terminal (TTY)
00049 */
00050 #include <termios.h>
00051 /*
00052 Fornece funções para criação e manipulação de pseudo-terminais (PTYs),
00053 um mecanismo essencial em sistemas Unix para emular terminais virtuais.
00054 */
00055 #include <pty.h>
00056
00064 class GPSSim {
00065 private:
00066
00067 public:
00068
00085     explicit
00086     GPSSim(
00087         double latitude_inicial_graus,
00088         double longitude_inicial_graus,
00089         double altitude_metros = 10.0,
00090         double frequencia_atualizacao_hz = 1.0,
00091         double velocidade_nos = 0.0
00092     ) : lat_(latitude_inicial_graus),
00093         lon_(longitude_inicial_graus),
00094         alt_(altitude_metros),
00095         periodo_atualizacao_((frequencia_atualizacao_hz) > 0 ?
00096                              chrono::milliseconds((int)std::llround(1000.0/frequencia_atualizacao_hz))
    :
00097                              chrono::milliseconds(1000)),
00098         velocidade_nos_(velocidade_nos)
00099     {
00100
00101         // Cria o par de pseudo-terminais
00102         if(
00103             openpty( &_fd_pai, &_fd_filho, _nome_pt_filho, nullptr, nullptr ) != 0
00104         ){
00105             throw std::runtime_error("Falha ao criar pseudo-terminal");
00106         }
00107
00108         // Configura o terminal filho para simular o módulo real (9600 8N1)
00109         termios config_com{};              // Cria a estrutura vazia
00110         tcgetattr(_fd_filho, &config_com); // Lê as configurações atuais e armazena na struct
```

```
00111         cfsetispeed(&config_com, B9600);   // Definimos velocidade de entrada e de saída
00112         cfsetospeed(&config_com, B9600);   // Essa constante está presente dentro do termios.h
00113         // Diversas operações bits a bits
00114         config_com.c_cflag = (config_com.c_cflag & ~CSIZE) | CS8;
00115         config_com.c_cflag |= (CLOCAL | CREAD);
00116         config_com.c_cflag &= ~(PARENB | CSTOPB);
00117         config_com.c_iflag = IGNPAR;
00118         config_com.c_oflag = 0;
00119         config_com.c_lflag = 0;
00120         tcsetattr(_fd_filho, TCSANOW, &config_com);  // Aplicamos as configurações
00121
00122         // Fecha o filho - será aberto pelo usuário no caminho correto
00123         // Mantemos o Pai aberto para procedimentos posteriores
00124         ::close(_fd_filho);
00125     }
00126
00132     ~GPSSim(){ stop(); if( _fd_pai >= 0 ){ ::close(_fd_pai); } }
00133
00138     std::string
00139     obter_caminho_terminal_filho() const { return std::string(_nome_pt_filho); }
00140
00149     void
00150     init(){
00151         if(
00152             // Variáveis atômicas possuem esta funcionalidade
00153             _is_exec.exchange(true)
00154         ){
00155
00156             return;
00157         }
00158
00159         _thread_geradora = std::thread(
00160                                         // Função Lambda que será executada pela thread
00161                                         // Observe que os argumentos utilizados serão obtidos pelo
00162                                         // this inserido nos colchetes
00163                                         [this]{ loop(); }
00164                                       );
00165     }
00166
00173     void
00174     stop(){
00175
00176         if(
00177             !_is_exec.exchange(false)
00178         ){
00179
00180             return;
00181         }
00182
00183         if(
00184             _thread_geradora.joinable()
00185         ){
00186
00187             _thread_geradora.join();
00188         }
00189     }
00190
00194     void
00195     config_traj(
00196         double raio_metros = 20.0,
00197         double periodo_segundos = 120.0
00198     ){
00199
00200         _raio         = raio_metros;
00201         _periodo_circ = periodo_segundos;
00202         _mov_circ     = true;
00203     }
00204
00208     void
00209     config_traj(){
00210
00211         _mov_circ = false;
00212     }
00213
00214
00215
00216
00217
00218
00219
00220
00221
00222
00223
00224
00225
00226
```

```
00227
00228
00229
00230 };
00231
00232 #endif // GPSSim_HPP
```

2.4 src/TrackSense.hpp File Reference

This graph shows which files directly or indirectly include this file:



2.5 TrackSense.hpp

Go to the documentation of this file.

```
00001 #ifndef TRACKSENSE_HPP
00002 #define TRACKSENSE_HPP
00003
00004
00005
00006
00007
00008
00009
00010
00011
00012
00013
00014
00015
00016
00017
00018
00019 #endif // TRACKSENSE_HPP
```

Index