RobotClient Documents

# 関数の解説

・RobotClient(std::string address,int port,int timeout,std::string matchCondition, boost::asio::io\_service ioService)

戻り値 無し

第一引数 サーバーのアドレス。”127.0.0.1”のような形式。std::string型

第二引数 サーバーの接続するポート。int型

第三引数 タイムアウトするまでの時間(秒単位)。int型

第四引数 通信を打ち切る文字列。std::string型

第五引数 ライブラリ内で使用するboost::asio::io\_service型の変数

コンストラクタ。

・void connect()

戻り値 無し

引数 無し

サーバーと接続する

・void set(std::map<std::string,int>)

戻り値 無し

第一引数 サーバーに保存したい変数。std::map<std::string,int>型

サーバーにstd::map<std::string,int>型の変数を自身が接続しているポート番号をkeyとして保存する。

・std::map<std::string,int> get(int port)

戻り値 サーバーに保存されていたデータ

第一引数 欲しいデータのkeyとなっているポート番号。int型

サーバーに欲しいデータのkeyを送り、そのkeyに対するvalueとして保存され

ているstd::map<std::string,int>型の変数を返す。

・void get(int port, std::map<std::string,int>& data)

戻り値 なし

第一引数 欲しいデータのkeyとなっているポート番号。int型

第二引数 サーバーに保存されていたデータを格納したい変数map型

サーバーに欲しいデータのkeyを送り、そのkeyに対するvalueとして保存され

ているデータを第二引数で与えられた変数に格納して返す。ひとつ前の関数と

較してこちらはいちいちstd::map<std::string,int>型の変数を生成しないので効率が良い。

# 例

1 #include "RobotClient.hpp"

2

3 int main(){

4 boost::asio::io\_service ioService;

5 RobotClient client("127.0.0.1",5000,5,"\n",ioService);

6 client.connect();

7 std::map<std::string, int>setData;

8 int i = 0;

9 while (1) {

10 setData["x"] = i++;

11 client.set(setData);

12 std::cout << client.get(5001)["a"] << std::endl;

13 std::cout << client.get(5002)["b"] << std::endl;

14 std::cout << client.get(5003)["c"] << std::endl;

15 }

16 return 0;

17 }

4行目 ライブラリで使用するboost::asio::io\_service型の変数を定義

5行目 サーバーのアドレスを127.0.0.1、接続先のポートを5000番、5秒でタイムアウト、”\n”で受信打ち切りとしてRobotClientのインスタンスを生成する

6行目 サーバーと接続する

11行目 サーバーに自身の持つ変数setDataを保存する

12行目 サーバーにポート番号5001番から接続している端末が保存した変数の”a”keyに対応

するvalueを出力する

13行目 サーバーにポート番号5002番から接続している端末が保存した変数の”b”keyに対応

するvalueを出力する

14行目 サーバーにポート番号5003番から接続している端末が保存した変数の”c”keyに対応

するvalueを出力する

# 使い方

・配布したヘッダファイルを全てインクルードディレクトリにコピーしRobotClient.hppをインクルードするだけ。

・ポート番号と各種端末の対応は以下のようになっている(2016/06/22現在)

|  |  |
| --- | --- |
| 5000 | RasPi1のコントローラー(サーバー) |
| 5001 | 空き |
| 5002 | 空き |
| 5003 | 空き |
| 5004 | RasPi4のコントローラー(クライアント) |
| 5005 | 空き |
| 5006 | 空き |

基本的にラベル番号+5000のポートにアクセスすると良いし、ある端末が保存したデータにアクセスしたい時はその端末のラベル番号を見れば簡単に分かるようになっている(ラベル番号1の端末が5000番のポートにアクセス(本当はローカルでデータをセットしてるだけなのでアクセスはしてない)しているのはミスです。要望があれば直します)

# 注意

・matchConditionは正しく設定しないと通信が終わらないので注意する。

　現状は”\n”をmachConditionとして使用しているのでこれに合わせて

　プログラムを設計する。

・ポート番号は被らないようにしなければならない。どのマシンにどういったポート番号を

　割り振るかは要相談。

・保存できる型はstd::map<std::string,int>型のみ。std::map<std::string,float>型や

　std::map<std::string,double>型といったものは使えない。

・サーバー側はRasPiという貧弱なマシンで複数の端末に通信しなければいけないのでできるだけ

　通信の回数を少なくして、かつ、通信の間隔をある程度開けるようにする。現状は1000μsごとにリクエストを行っている。