PSS

2.1

Generated by Doxygen 1.7.1

Wed Apr 6 2011 22:09:34

Contents

1 Main Page 1

2 Namespace Index 2

2.1 Namespace List . . . 2

3 Class Index 2

3.1 Class List . . . 2

4 File Index 3

4.1 File List . . . 3

5 Namespace Documentation 4

5.1 pss Namespace Reference . . . 4

5.2 pss::configuration Namespace Reference . . . 4

5.3 pss::serialcomm Namespace Reference . . . 4

5.4 pss::server Namespace Reference . . . 4

5.5 pss::server::database Namespace Reference . . . 4

5.6 pss::server::scheduling Namespace Reference . . . 4

5.7 pss::server::test Namespace Reference . . . 5

6 Class Documentation 5

6.1 pss::server::Bot Class Reference . . . 5

6.1.1 Detailed Description . . . 6

6.1.2 Constructor & Destructor Documentation . . . 6

6.1.3 Member Function Documentation . . . 6

6.1.4 Member Data Documentation . . . 8

6.2 pss::server::test::BotMotionTester1 Class Reference . . . 10

6.2.1 Detailed Description . . . 11

6.2.2 Member Function Documentation . . . 11

6.2.3 Member Data Documentation . . . 11

6.3 pss::server::test::BotMotionTester2 Class Reference . . . 12

6.3.1 Detailed Description . . . 12

6.3.2 Member Function Documentation . . . 12

6.3.3 Member Data Documentation . . . 12

6.4 pss::serialcomm::CommunicationAPI Class Reference . . . 13

6.4.1 Detailed Description . . . 14

6.4.2 Constructor & Destructor Documentation . . . 14

6.4.3 Member Function Documentation . . . 14
6.4.4 Member Data Documentation . . . 15
6.5 pss::configuration::Configure Class Reference . . . 16
6.5.1 Detailed Description . . . 17
6.5.2 Member Function Documentation . . . 17
6.5.3 Member Data Documentation . . . 18
6.6 pss::server::database::DBHandler Class Reference . . . 20
6.6.1 Detailed Description . . . 21
6.6.2 Member Function Documentation . . . 21
6.6.3 Member Data Documentation . . . 25
6.7 pss::server::Graph Class Reference . . . 25
6.7.1 Detailed Description . . . 26
6.7.2 Constructor & Destructor Documentation . . . 26
6.7.3 Member Function Documentation . . . 27
6.7.4 Member Data Documentation . . . 30
6.8 pss::server::test::GraphTester Class Reference . . . 33
6.8.1 Detailed Description . . . 33
6.8.2 Member Function Documentation . . . 33
6.9 pss::server::test::Patient_simulator Class Reference . . . 33
6.9.1 Detailed Description . . . 33
6.9.2 Member Function Documentation . . . 34
6.10 pss::server::scheduling::PollingThread Class Reference . . . 34
6.10.1 Detailed Description . . . 34
6.10.2 Constructor & Destructor Documentation . . . 34
6.10.3 Member Function Documentation . . . 35
6.10.4 Member Data Documentation . . . 35
6.11 pss::server::Position Class Reference . . . 36
6.11.1 Detailed Description . . . 36
6.11.2 Constructor & Destructor Documentation . . . 37
6.11.3 Member Data Documentation . . . 37
6.12 pss::serialcomm::CommunicationAPI::Receiver Class Reference . . . 37
6.12.1 Detailed Description . . . 38
6.12.2 Constructor & Destructor Documentation . . . 38
6.12.3 Member Function Documentation . . . 38
6.13 pss::server::RequestHandler Class Reference . . . 38
6.13.1 Detailed Description . . . 39

6.13.2 Constructor & Destructor Documentation . . . 39
6.13.3 Member Function Documentation . . . 40
6.13.4 Member Data Documentation . . . 43
6.14 pss::serialcomm::CommunicationAPI::Sender Class Reference . . . 44
6.14.1 Detailed Description . . . 45
6.14.2 Constructor & Destructor Documentation . . . 45
6.14.3 Member Function Documentation . . . 45
6.14.4 Member Data Documentation . . . 45
6.15 pss::server::test::SimpleRead Class Reference . . . 45
6.15.1 Detailed Description . . . 46
6.15.2 Constructor & Destructor Documentation . . . 46
6.15.3 Member Function Documentation . . . 46
6.15.4 Member Data Documentation . . . 46
6.16 pss::server::test::SimpleWrite Class Reference . . . 47
6.16.1 Detailed Description . . . 47
6.16.2 Member Function Documentation . . . 48
6.16.3 Member Data Documentation . . . 48
6.17 pss::server::Utils Class Reference . . . 49
6.17.1 Detailed Description . . . 49
6.17.2 Member Function Documentation . . . 49

7 File Documentation **50**

7.1 src/BOT_CODE/bot.c File Reference . . . 50
7.1.1 Define Documentation . . . 51
7.1.2 Function Documentation . . . 53
7.2 src/BOT_CODE/bot.c . . . 53
7.3 src/BOT_CODE/bot.d File Reference . . . 56
7.4 src/BOT_CODE/bot.d . . . 56
7.5 src/BOT_CODE/bot_2.c File Reference . . . 56
7.5.1 Define Documentation . . . 56
7.5.2 Function Documentation . . . 58
7.6 src/BOT_CODE/bot_2.c . . . 58
7.7 src/BOT_CODE/bot_motion.h File Reference . . . 61
7.7.1 Define Documentation . . . 62
7.7.2 Function Documentation . . . 64
7.7.3 Variable Documentation . . . 67
7.8 src/BOT_CODE/bot_motion.h . . . 68

7.9 src/BOT_CODE/IR.c File Reference 74
7.9.1 Define Documentation 75
7.9.2 Function Documentation 76
7.9.3 Variable Documentation 77
7.10 src/BOT_CODE/IR.c 77
7.11 src/BOT_CODE/lcd.h File Reference 80
7.11.1 Define Documentation 81
7.11.2 Function Documentation 82
7.11.3 Variable Documentation 83
7.12 src/BOT_CODE/lcd.h 84
7.13 src/SERVER_CODE/pss/configuration/Configure.java File Reference 88
7.14 src/SERVER_CODE/pss/configuration/Configure.java 88
7.15 src/SERVER_CODE/pss/serialcomm/CommunicationAPI.java File Reference 89
7.16 src/SERVER_CODE/pss/serialcomm/CommunicationAPI.java 89
7.17 src/SERVER_CODE/pss/server/Bot.java File Reference 93
7.18 src/SERVER_CODE/pss/server/Bot.java 93
7.19 src/SERVER_CODE/pss/server/database/DBHandler.java File Reference 96
7.20 src/SERVER_CODE/pss/server/database/DBHandler.java 96
7.21 src/SERVER_CODE/pss/server/Graph.java File Reference 102
7.22 src/SERVER_CODE/pss/server/Graph.java 102
7.23 src/SERVER_CODE/pss/server/Position.java File Reference 109
7.24 src/SERVER_CODE/pss/server/Position.java 109
7.25 src/SERVER_CODE/pss/server/RequestHandler.java File Reference 109
7.26 src/SERVER_CODE/pss/server/RequestHandler.java 110
7.27 src/SERVER_CODE/pss/server/scheduling/PollingThread.java File Reference 116
7.28 src/SERVER_CODE/pss/server/scheduling/PollingThread.java 116
7.29 src/SERVER_CODE/pss/server/test/BotMotionTester1.java File Reference 117
7.30 src/SERVER_CODE/pss/server/test/BotMotionTester1.java 117
7.31 src/SERVER_CODE/pss/server/test/BotMotionTester2.java File Reference 119
7.32 src/SERVER_CODE/pss/server/test/BotMotionTester2.java 120
7.33 src/SERVER_CODE/pss/server/test/GraphTester.java File Reference 121
7.34 src/SERVER_CODE/pss/server/test/GraphTester.java 122
7.35 src/SERVER_CODE/pss/server/test/Patient_simulator.java File Reference 122
7.36 src/SERVER_CODE/pss/server/test/Patient_simulator.java 123
7.37 src/SERVER_CODE/pss/server/test/SimpleRead.java File Reference 123
7.38 src/SERVER_CODE/pss/server/test/SimpleRead.java 123

7.39 src/SERVER_CODE/pss/server/test/SimpleWrite.java File Reference 126
7.40 src/SERVER_CODE/pss/server/test/SimpleWrite.java 126
7.41 src/SERVER_CODE/pss/server/Utils.java File Reference 128
7.42 src/SERVER_CODE/pss/server/Utils.java 128

1 Main Page

See the file list to find the bot_code files.

PROJECT DESCTIPTION

The project aims at helping the patients and hostipal management by providing automated service bots, who can be called by the patient simply by using a cheap TV remote, and who provide a guarantee of no-deadlock/race conditions. The server uses a full fledged database (MySQL) to maintain the queue and manages the bot wisely using a perfectly scalable design. The bots are fair to all the patients. The bot notifies the guards if it is blocked by an SMS.

For more details, see the presentation

TEAM MEMBERS

Pritish Kamath
Rohit Saraf
Ashish Mathew
Vivek Madan

Execution Instructions (Ubuntu)

To execute the code for the first time run the install script as

sudo ./install

(requires proxy settings)

This will install all the dependencies and drivers required for the project and generate a Makefile.

Note: 1.The properties file config/config.properties might require a change. (e.g. Zigbee COM port, Database settings)

Then

To compile the server code	**make server**
To run the server code	**make run**
To compile and program the serving bot no 1.	**make bot**
To compile and program the serving bot no 2.	**make bot1**
To compile and program the patient_IR bot	**make patient**
To clean	**make clean**
To open the documentation	**make doc**

2 Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

pss 4

pss::configuration 4

pss::serialcomm 4

pss::server 4

pss::server::database 4

pss::server::scheduling 4

pss::server::test 5

3 Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

pss::server::Bot 5

pss::server::test::BotMotionTester1 10

pss::server::test::BotMotionTester2 12

pss::serialcomm::CommunicationAPI 13

pss::configuration::Configure 16

pss::server::database::DBHandler 20

pss::server::Graph 25

pss::server::test::GraphTester 33

pss::server::test::Patient_simulator 33

pss::server::scheduling::PollingThread 34

pss::server::Position 36

pss::serialcomm::CommunicationAPI::Receiver 37

pss::server::RequestHandler 38

pss::serialcomm::CommunicationAPI::Sender 44

pss::server::test::SimpleRead 45

pss::server::test::SimpleWrite 47

pss::server::Utils 49

4 File Index

4.1 File List

Here is a list of all files with brief descriptions:

src/BOT_CODE/bot.c 50

src/BOT_CODE/bot.d 56

src/BOT_CODE/bot_2.c 56

src/BOT_CODE/bot_motion.h 61

src/BOT_CODE/IR.c 74

src/BOT_CODE/lcd.h 80

src/SERVER_CODE/pss/configuration/Configure.java 88

src/SERVER_CODE/pss/serialcomm/CommunicationAPI.java 89

src/SERVER_CODE/pss/server/Bot.java 93

src/SERVER_CODE/pss/server/Graph.java 102

src/SERVER_CODE/pss/server/Position.java 109

src/SERVER_CODE/pss/server/RequestHandler.java 109

src/SERVER_CODE/pss/server/Utils.java 128

src/SERVER_CODE/pss/server/database/DBHandler.java 96

src/SERVER_CODE/pss/server/scheduling/PollingThread.java 116

src/SERVER_CODE/pss/server/test/BotMotionTester1.java 117

src/SERVER_CODE/pss/server/test/BotMotionTester2.java 119

src/SERVER_CODE/pss/server/test/GraphTester.java 121

src/SERVER_CODE/pss/server/test/Patient_simulator.java 122

src/SERVER_CODE/pss/server/test/SimpleRead.java 123

src/SERVER_CODE/pss/server/test/SimpleWrite.java 126

5 Namespace Documentation

5.1 pss Namespace Reference

Namespaces

- namespace configuration
- namespace serialcomm
- namespace server

5.2 pss::configuration Namespace Reference

Classes

- class Configure

5.3 pss::serialcomm Namespace Reference

Classes

- class CommunicationAPI

5.4 pss::server Namespace Reference

Namespaces

- namespace database
- namespace scheduling
- namespace test

Classes

- class Bot
- class Graph
- class Position
- class RequestHandler
- class Utils

5.5 pss::server::database Namespace Reference

Classes

- class DBHandler

5.6 pss::server::scheduling Namespace Reference

Classes

- class PollingThread

5.7 pss::server::test Namespace Reference

Classes

- class BotMotionTester1
- class BotMotionTester2
- class GraphTester
- class Patient_simulator
- class SimpleRead
- class SimpleWrite

6 Class Documentation

6.1 pss::server::Bot Class Reference

Public Member Functions

- Boolean isOriginalOrientation ()
- Boolean isBotIdle ()
- Boolean isAtHome ()
- Boolean isSafePos (Position pos)
- Bot (DBHandler dbis, int id, CommunicationAPI capi)
- void setPosition (Position x)
- void gotoNextCross ()
- void printBotPos ()
- void turnRight ()
- void turnBack ()
- void turnLeft ()

Static Public Member Functions

- static int getIdMess (char mess)

Static Public Attributes

- static final Boolean DEBUG = false
- static final int running = 0
- static final int stationary = 1
- static final int obstruction = 3
- static int bot1 = 0
- static int bot2 = 1

Package Attributes

- boolean isInUse = false
- int id
- int status
- Position currpos
- DBHandler dbh

- CommunicationAPI capi
- int pid
- Graph g = new Graph()

6.1.1 Detailed Description

Definition at line 29 of file Bot.java.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 pss::server::Bot::Bot (DBHandler *dbis*, int *id*, CommunicationAPI *capi*) [inline]

Bot Object Constructor

Parameters

dbis Database Handler

id Unique ID for the bot.

capi Comunication API Object.

See also

CommunicationAPI
DBHandler

Definition at line 171 of file Bot.java.

6.1.3 Member Function Documentation

6.1.3.1 static int pss::server::Bot::getIdMess (char *mess*) [inline, static]

All messages between bot and server are 8 bits long. In Each message the ID of the bot which sent the message and the message itself is encoded.

Parameters

mess 8 bit message.

Returns

bot ID of the bot which sent the message.

Definition at line 193 of file Bot.java.

6.1.3.2 void pss::server::Bot::gotoNextCross ( ) [inline]

Send an instruction to the bot to go forward upto the next cross (intersection) and update the new position of the bot.

Definition at line 210 of file Bot.java.

6.1.3.3 Boolean pss::server::Bot::isAtHome ( ) [inline]

this function checks if the bot is at server (origional starting position).

Returns

True if bot is at the server; false otherwise.

Definition at line 126 of file Bot.java.

6.1.3.4 Boolean pss::server::Bot::isBotIdle ( ) [inline]

This function checks if the bot is currently servicing a request or not.

Returns

True if it is **NOT** servicing a request. False otherwise.

Definition at line 108 of file Bot.java.

6.1.3.5 Boolean pss::server::Bot::isOriginalOrientation ( ) [inline]

Is the bot in its Origional Starting Orientation. The origional orientation is fixed based on the bot ID.

Returns

True if it was in origional position. Flase otherwise.

See also

Position

Definition at line 87 of file Bot.java.

6.1.3.6 Boolean pss::server::Bot::isSafePos (Position *pos*) [inline]

this function checks if the position pos is safe to visit i.e. there willl not be any collision/deadlock if the other bot also deceides the visit the node pos at the same time

Parameters

pos Position that the bot is about to visit.

Returns

True if the position is safe false otherwise.

See also

Position

Definition at line 149 of file Bot.java.

6.1.3.7 void pss::server::Bot::printBotPos ( ) [inline]

For debugging purposes

Print the current position of the bot

Definition at line 229 of file Bot.java.

6.1.3.8 void pss::server::Bot::setPosition (Position *x*) [inline]

given a position x, set the bot's current position to be x

Parameters

x

See also

Position

Definition at line 203 of file Bot.java.

6.1.3.9 void pss::server::Bot::turnBack ( ) [inline]

Sends an instruction to the bot to turn back and set the position of the bot to the new position after turning back

Definition at line 251 of file Bot.java.

6.1.3.10 void pss::server::Bot::turnLeft ( ) [inline]

Sends an instruction to the bot to turn left and set the position of the bot to the new position after turning left

Definition at line 265 of file Bot.java.

6.1.3.11 void pss::server::Bot::turnRight ( ) [inline]

Sends an instruction to the bot to turn right and set the position of the bot to the new position after turning right

Definition at line 236 of file Bot.java.

6.1.4 Member Data Documentation

6.1.4.1 int pss::server::Bot::bot1 = 0 [static]

ID of the first bot

Definition at line 76 of file Bot.java.

6.1.4.2 int pss::server::Bot::bot2 = 1 [static]

ID of the second bot

Definition at line 80 of file Bot.java.

6.1.4.3 CommunicationAPI pss::server::Bot::capi [package]

Definition at line 70 of file Bot.java.

6.1.4.4 Position pss::server::Bot::currpos [package]

Current position of the bot.

See also

Position

Definition at line 68 of file Bot.java.

6.1.4.5 DBHandler pss::server::Bot::dbh [package]

Definition at line 69 of file Bot.java.

6.1.4.6 final Boolean pss::server::Bot::DEBUG = false [static]

Boolean flag for debugging purposes

Definition at line 34 of file Bot.java.

6.1.4.7 Graph pss::server::Bot::g = new Graph() [package]

Definition at line 72 of file Bot.java.

6.1.4.8 int pss::server::Bot::id [package]

Integer id of a bot

Definition at line 56 of file Bot.java.

6.1.4.9 boolean pss::server::Bot::isInUse = false [package]

Boolean flag to indicate if the bot is currently servicing a request or not

Definition at line 51 of file Bot.java.

6.1.4.10 final int pss::server::Bot::obstruction = 3 [static]

Status flag for the bot.

Bot is obstructed while carrying out current instruction.

Definition at line 46 of file Bot.java.

6.1.4.11 int pss::server::Bot::pid [package]

Definition at line 71 of file Bot.java.

6.1.4.12 final int pss::server::Bot::running = 0 [static]

Status flag for the bot.

Bot is currently executing an instruction issued by the server

Definition at line 38 of file Bot.java.

6.1.4.13 final int pss::server::Bot::stationary = 1 [static]

Status flag for the bot.

Bot has finished executing current instruction and is idle.

Definition at line 42 of file Bot.java.

6.1.4.14 int pss::server::Bot::status [package]

Status of bot as given by status flags.

See also

running
stationary
obstruction

Definition at line 63 of file Bot.java.

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

- src/SERVER_CODE/pss/server/Bot.java

6.2 pss::server::test::BotMotionTester1 Class Reference

Static Public Member Functions

- static void printMess (Character message)
- static void main (String[ ] args)

Static Public Attributes

- static final Boolean DEBUG = false
- static final Boolean mode1 = false
- static final Boolean mess_debug = true

6.2.1 Detailed Description

Simulates bot by following instructions transmitted to it by the server

Definition at line 37 of file BotMotionTester1.java.

6.2.2 Member Function Documentation

6.2.2.1 static void pss::server::test::BotMotionTester1::main (String[ ] *args*) [inline, static]

This main function sends an initial request to the server an then simulates the behaviour of bot number 1 as it carries out the instructions to service the request as it receives them from the server This test is done as follows

1. Send an initial message to server indicating that patient 3 wants water.

2. Keep sending acks

Parameters

args

Definition at line 97 of file BotMotionTester1.java.

6.2.2.2 static void pss::server::test::BotMotionTester1::printMess (Character *message*) [inline, static]

Prints the 8 bit message and all interpretations of the information encoded in the message

Parameters

message Message of the bot

Definition at line 56 of file BotMotionTester1.java.

6.2.3 Member Data Documentation

6.2.3.1 final Boolean pss::server::test::BotMotionTester1::DEBUG = false [static]

Debugging flag

Definition at line 42 of file BotMotionTester1.java.

6.2.3.2 final Boolean pss::server::test::BotMotionTester1::mess_debug = true [static]

Debugging flag

Definition at line 50 of file BotMotionTester1.java.

6.2.3.3 final Boolean pss::server::test::BotMotionTester1::mode1 = false [static]

Debugging flag

Definition at line 46 of file BotMotionTester1.java.

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

- src/SERVER_CODE/pss/server/test/BotMotionTester1.java

6.3 pss::server::test::BotMotionTester2 Class Reference

Static Public Member Functions

- static void print_mess (Character message)
- static void main (String[ ] args)

Static Public Attributes

- static final Boolean DEBUG = false
- static final Boolean mode1 = false
- static final Boolean mess_debug = true

6.3.1 Detailed Description

Author

ashish

Definition at line 34 of file BotMotionTester2.java.

6.3.2 Member Function Documentation

6.3.2.1 static void pss::server::test::BotMotionTester2::main (String[ ] *args*) [inline, static]

Definition at line 71 of file BotMotionTester2.java.

6.3.2.2 static void pss::server::test::BotMotionTester2::print_mess (Character *message*) [inline, static]

Definition at line 40 of file BotMotionTester2.java.

6.3.3 Member Data Documentation

6.3.3.1 final Boolean pss::server::test::BotMotionTester2::DEBUG = false [static]

Definition at line 36 of file BotMotionTester2.java.

6.3.3.2 final Boolean pss::server::test::BotMotionTester2::mess_debug = true [static]

Definition at line 38 of file BotMotionTester2.java.

6.3.3.3 final Boolean pss::server::test::BotMotionTester2::mode1 = false [static]

Definition at line 37 of file BotMotionTester2.java.

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

- src/SERVER_CODE/pss/server/test/BotMotionTester2.java

6.4 pss::serialcomm::CommunicationAPI Class Reference

Classes

- class Receiver
- class Sender

Public Member Functions

- CommunicationAPI (String port)
- void close ()
- void open ()
- void send (String messageString)
- void receive ()
- void getNextCharFromBufferIfPresent ()
- Character next_char_in_buffer ()

Static Public Member Functions

- static void main (String[ ] args) throws InterruptedException

Static Public Attributes

- static final Boolean DEBUG = false

Package Attributes

- InputStream inputStream
- SerialPort serialPort
- String defaultPort
- Thread readThread

Static Package Attributes

- static CommPortIdentifier portId
- static Enumeration portList
- static OutputStream outputStream
- static boolean outputBufferEmptyFlag = false
- static ConcurrentLinkedQueue< Character > in_buffer = new ConcurrentLinkedQueue<Character>()

6.4.1 Detailed Description

Since

22 Mar RXTXComm

Author

rohit Provides a multithreaded API for accessing serial port

Definition at line 46 of file CommunicationAPI.java.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 pss::serialcomm::CommunicationAPI::CommunicationAPI (String *port*) [inline]

Definition at line 59 of file CommunicationAPI.java.

6.4.3 Member Function Documentation

6.4.3.1 void pss::serialcomm::CommunicationAPI::close ( ) [inline]

Close the serial port

Definition at line 67 of file CommunicationAPI.java.

6.4.3.2 void pss::serialcomm::CommunicationAPI::getNextCharFromBufferIfPresent ( ) [inline]

Definition at line 171 of file CommunicationAPI.java.

6.4.3.3 static void pss::serialcomm::CommunicationAPI::main (String[ ] *args*) throws InterruptedException [inline, static]

Definition at line 233 of file CommunicationAPI.java.

6.4.3.4 Character pss::serialcomm::CommunicationAPI::next_char_in_buffer ( ) [inline]

Definition at line 175 of file CommunicationAPI.java.

6.4.3.5 void pss::serialcomm::CommunicationAPI::open ( ) [inline]

Definition at line 74 of file CommunicationAPI.java.

6.4.3.6 void pss::serialcomm::CommunicationAPI::receive ( ) [inline]

A receiver thread is always on!

Definition at line 165 of file CommunicationAPI.java.

6.4.3.7 void pss::serialcomm::CommunicationAPI::send (String *messageString*) [inline]

Definition at line 157 of file CommunicationAPI.java.

6.4.4 Member Data Documentation

6.4.4.1 final Boolean pss::serialcomm::CommunicationAPI::DEBUG = false [static]

Definition at line 57 of file CommunicationAPI.java.

6.4.4.2 String pss::serialcomm::CommunicationAPI::defaultPort [package]

Definition at line 52 of file CommunicationAPI.java.

6.4.4.3 ConcurrentLinkedQueue<Character> pss::serialcomm::CommunicationAPI::in_buffer = new ConcurrentLinkedQueue<Character>() [static, package]

Definition at line 56 of file CommunicationAPI.java.

6.4.4.4 InputStream pss::serialcomm::CommunicationAPI::inputStream [package]

Definition at line 50 of file CommunicationAPI.java.

6.4.4.5 boolean pss::serialcomm::CommunicationAPI::outputBufferEmptyFlag = false [static, package]

Definition at line 55 of file CommunicationAPI.java.

6.4.4.6 OutputStream pss::serialcomm::CommunicationAPI::outputStream [static, package]

Definition at line 54 of file CommunicationAPI.java.

6.4.4.7 CommPortIdentifier pss::serialcomm::CommunicationAPI::portId [static, package]

Definition at line 48 of file CommunicationAPI.java.

6.4.4.8 Enumeration pss::serialcomm::CommunicationAPI::portList [static, package]

Definition at line 49 of file CommunicationAPI.java.

6.4.4.9 Thread pss::serialcomm::CommunicationAPI::readThread [package]

Definition at line 53 of file CommunicationAPI.java.

6.4.4.10 SerialPort pss::serialcomm::CommunicationAPI::serialPort [package]

Definition at line 51 of file CommunicationAPI.java.

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

- src/SERVER_CODE/pss/serialcomm/CommunicationAPI.java

6.5 pss::configuration::Configure Class Reference

Static Public Member Functions

- static synchronized Configure createInstance () throws IOException
- static Configure getInstance () throws IOException
- static void setValues (Properties p)
- static Properties loadProperties () throws IOException

Static Public Attributes

- static String ZIGBEE_PORT
- static String DB_UN
- static String DB_PASS
- static String DB_DRIVER
- static String DB_URL
- static String SMS_UN
- static String SMS_PASS
- static String SMS_MSG
- static String SMS_NUM
- static Integer NUM_BOTS
- static Boolean CLEAR_DB
- static String TEST_PORT1
- static String TEST_PORT2

Static Package Attributes

- static Configure instance = null

6.5.1 Detailed Description

This is a standard class in most Java programs for storing and retreiving parameters and values which would have otherwise been hardcoded. This class reads a text file config.properties which is simply a list of paramenters and their properties. Everywhere in the code these parameters are accessed through this class only. If any of these paraameters have to be changed we only need to do it in this tex file.

Author

ashish, rohit

Definition at line 36 of file Configure.java.

6.5.2 Member Function Documentation

6.5.2.1 static synchronized Configure pss::configuration::Configure::createInstance ( ) throws IOException [inline, static]

Initialization

Returns

Exceptions

IOException

Definition at line 101 of file Configure.java.

6.5.2.2 static Configure pss::configuration::Configure::getInstance ( ) throws IOException [inline, static]

Get an instance

Returns

Exceptions

IOException

Definition at line 114 of file Configure.java.

6.5.2.3 static Properties pss::configuration::Configure::loadProperties ( ) throws IOException [inline, static]

Loads properties and values from configuration file

Returns

Exceptions

IOException

Definition at line 147 of file Configure.java.

6.5.2.4 static void pss::configuration::Configure::setValues (Properties *p*) [inline, static]

Set values of a property

Parameters

p

See also

Properties

Definition at line 126 of file Configure.java.

6.5.3 Member Data Documentation

6.5.3.1 Boolean pss::configuration::Configure::CLEAR_DB [static]

Flag indicating whether the existing database should be completely cleared and a new one created from start. Can be true or false

Definition at line 83 of file Configure.java.

6.5.3.2 String pss::configuration::Configure::DB_DRIVER [static]

Address of the MySQL JDBC driver.

Definition at line 53 of file Configure.java.

6.5.3.3 String pss::configuration::Configure::DB_PASS [static]

Password for accessing MySQL database

Definition at line 49 of file Configure.java.

6.5.3.4 String pss::configuration::Configure::DB_UN [static]

Username for accessing MySQL database

Definition at line 45 of file Configure.java.

6.5.3.5 String pss::configuration::Configure::DB_URL [static]

URL of the SQL database

Definition at line 57 of file Configure.java.

6.5.3.6 Configure pss::configuration::Configure::instance = null [static, package]

Definition at line 94 of file Configure.java.

6.5.3.7 Integer pss::configuration::Configure::NUM_BOTS [static]

Number of serving (nurse) bots

Definition at line 78 of file Configure.java.

6.5.3.8 String pss::configuration::Configure::SMS_MSG [static]

Message to be sent by SMS

Definition at line 70 of file Configure.java.

6.5.3.9 String pss::configuration::Configure::SMS_NUM [static]

Phone Number to which the SMS must be sent.

Definition at line 74 of file Configure.java.

6.5.3.10 String pss::configuration::Configure::SMS_PASS [static]

Password for the SMS Gateway

Definition at line 66 of file Configure.java.

6.5.3.11 String pss::configuration::Configure::SMS_UN [static]

Username for the SMS Gateway

Definition at line 62 of file Configure.java.

6.5.3.12 String pss::configuration::Configure::TEST_PORT1 [static]

USB Serial Port used for Simulating Bot 1 during testing using simulation . Same as PORT above.

Definition at line 89 of file Configure.java.

6.5.3.13 String pss::configuration::Configure::TEST_PORT2 [static]

USB Serial Port used for Simulating Bot 2 during testing using simulation. Same as PORT above.

Definition at line 93 of file Configure.java.

6.5.3.14 String pss::configuration::Configure::ZIGBEE_PORT [static]

Address of Serial USB PORT on server to which the Zigbee module has been connected

Example : /dev/tty0

Definition at line 41 of file Configure.java.

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

- src/SERVER_CODE/pss/configuration/Configure.java

6.6 pss::server::database::DBHandler Class Reference

Public Member Functions

- void closeConnection ()
- void createConnection ()
- void createDatabase ()
- void deleteDatabase ()
- ResultSet executeStatement (String cmd)
- void executeUpdate (String cmd)
- boolean addPatient (int patient_ID, int Pos)
- boolean deletePatient (int id)
- boolean addRequest (int request_id, int patient_id, String item, String status)
- boolean deleteRequest (int request_id)
- boolean updateRequestStatus (int request_id, String status)
- boolean addAssignment (int request_id, int bot_ID)
- boolean deleteAssignment (int id, boolean is_request_id)
- int getNextRequestFIFO ()
- int getAssignedRequest (int bot_ID)
- int getPatientOfRequest (int req_id)

Static Public Member Functions

- static void main (String[ ] args) throws SQLException

Static Public Attributes

- static final Boolean DEBUG = false

6.6.1 Detailed Description

Handler functions for interaction with MySQL database

Definition at line 37 of file DBHandler.java.

6.6.2 Member Function Documentation

6.6.2.1 boolean pss::server::database::DBHandler::addAssignment (int *request_id*, int *bot_ID*) [inline]

Add record of bot being assigned to service a particular request

Parameters

request_id Request ID

bot_ID Bot ID

Returns

True if update is successful false otherwise

Definition at line 314 of file DBHandler.java.

6.6.2.2 boolean pss::server::database::DBHandler::addPatient (int *patient_ID*, int *Pos*) [inline]

Add a patient record

Parameters

patient_ID Patient ID

Pos Position on graph (Room number)

Returns

True if update is successful false otherwise

Definition at line 211 of file DBHandler.java.

6.6.2.3 boolean pss::server::database::DBHandler::addRequest (int *request_id*, int *patient_id*, String *item*, String *status*) [inline]

Add record of a request

Parameters

request_id Unique Request ID

patient_id Patient ID

item Item requested for

status Status of request

Returns

True if update is successful false otherwise

Definition at line 257 of file DBHandler.java.

6.6.2.4 void pss::server::database::DBHandler::closeConnection ( ) [inline]

Safely close database connection

Definition at line 76 of file DBHandler.java.

6.6.2.5 void pss::server::database::DBHandler::createConnection ( ) [inline]

Create a connection to the database the following parameters must be filled in here.

(1) Path to JAVA JDBC Driver

(2) URL to MySQL database

(3) Username to access MySQL database

(4) Password to access MySQL database

Definition at line 91 of file DBHandler.java.

6.6.2.6 void pss::server::database::DBHandler::createDatabase ( ) [inline]

Erases all existing tables and creates a fresh empty tables

Definition at line 123 of file DBHandler.java.

6.6.2.7 boolean pss::server::database::DBHandler::deleteAssignment (int *id*, boolean *is_request_id*) [inline]

Delete record of an assignment

Parameters

id BotID or Request ID

is_request_id True if parameter id is requestID false if it is botID

Returns

True if update is successful false otherwise

Definition at line 333 of file DBHandler.java.

6.6.2.8 void pss::server::database::DBHandler::deleteDatabase ( ) [inline]

Completely delete the existing database as well as all the information stored in it

Definition at line 133 of file DBHandler.java.

6.6.2.9 boolean pss::server::database::DBHandler::deletePatient (int *id*) [inline]

Deletes patient record

Parameters

id Patient ID

Returns

True if update is successful false otherwise

Definition at line 237 of file DBHandler.java.

6.6.2.10 boolean pss::server::database::DBHandler::deleteRequest (int *request_id*) [inline]

Delete record of a request

Parameters

request_id Request ID

Returns

True if update is successful false otherwise

Definition at line 277 of file DBHandler.java.

6.6.2.11 ResultSet pss::server::database::DBHandler::executeStatement (String *cmd*) [inline]

Execute an SQL Query on the database

Parameters

cmd SQL Query

Returns

ResultSet

See also

ResultSet

Definition at line 148 of file DBHandler.java.

6.6.2.12 void pss::server::database::DBHandler::executeUpdate (String *cmd*) [inline]

Execute SQL Update on the database

Parameters

cmd SQL Update statement

Definition at line 166 of file DBHandler.java.

6.6.2.13 int pss::server::database::DBHandler::getAssignedRequest (int *bot_ID*) [inline]

Gets the request assigned to a particular bot

Parameters

bot_ID Bot ID

Returns

Request ID or -1 if no such bot exists

Definition at line 376 of file DBHandler.java.

6.6.2.14 int pss::server::database::DBHandler::getNextRequestFIFO ( ) [inline]

Get next request based on FIFIO scheduling policy

Returns

request ID of next request that has to be assigned to a bot for servicing or -1 if no pending requests.

Definition at line 354 of file DBHandler.java.

6.6.2.15 int pss::server::database::DBHandler::getPatientOfRequest (int *req_id*) [inline]

Get ID of patient who made this request

Parameters

req_id Request ID

Returns

Patient ID or -1 if no such request is present in the system

Definition at line 398 of file DBHandler.java.

6.6.2.16 static void pss::server::database::DBHandler::main (String[ ] *args*) throws SQLException [inline, static]

Simple main to test correctness of any of the functions above

Parameters

args

Exceptions

SQLException

Definition at line 420 of file DBHandler.java.

6.6.2.17 boolean pss::server::database::DBHandler::updateRequestStatus (int *request_id*, String *status*) [inline]

Update status of a request

Parameters

request_id Request ID

status New status of request

Returns

True if update is successful false otherwise

Definition at line 295 of file DBHandler.java.

6.6.3 Member Data Documentation

6.6.3.1 final Boolean pss::server::database::DBHandler::DEBUG = false [static]

Debugging flags

Definition at line 203 of file DBHandler.java.

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

- src/SERVER_CODE/pss/server/database/DBHandler.java

6.7 pss::server::Graph Class Reference

Public Member Functions

- void init_graph ()
- Position Left_turn (Position p)
- Position Back (Position p)
- Position Right_turn (Position p)
- Position Straight (Position p)
- Position pos_after_action (Position curr, int action)
- void move_the_bot (Position curr, int patient_id)
- int find_distance (Position curr, int final_pos, int counter)
- Graph ()
- int search (int patient_id, Position curr)

Public Attributes

- int[ ][ ][ ] grph = new int[16][4][4]
- int[ ][ ][ ] distance_pos_ori_pos = new int[16][4][16]

Static Public Attributes

- static final int RIGHT = 0
- static final int LEFT = 1
- static final int STRAIGHT = 2
- static final int BACKWARD = 3
- static final int FINISH = 5
- static final int NOPATH = -2
- static final int BOT_POLLING = 5
- static final int PATIENT_POLLING = 6
- static final int NORTH = 0
- static final int SOUTH = 1
- static final int EAST = 2
- static final int WEST = 3
- static final int INFINTY = 25
- static final int SERVER_ID1_position = 14
- static final int SERVER_ID2_position = 15
- static final int SERVER_ID1_orientation = Graph.SOUTH
- static final int SERVER_ID2_orientation = Graph.NORTH
- static final int SERVER_ID1 = -2
- static final int SERVER_ID2 = -3

Package Attributes

- boolean DEBUG = false
- int no_calls = 0

Static Package Attributes

- static Map< Integer, Integer > patientPos = new HashMap<Integer, Integer>()

6.7.1 Detailed Description

This class contains all information about the floor plan of the hospital. In the present implementation the arena has been hardcoded in this class. If the floor plan or bot routes are modified then this is the only class that needs to be changed. All distance orientation and path finding functions are part of this graph. Currently the arena is as follows.

Definition at line 34 of file Graph.java.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 pss::server::Graph::Graph ( ) [inline]

Default Constructor

Definition at line 406 of file Graph.java.

6.7.3 Member Function Documentation

6.7.3.1 Position pss::server::Graph::Back (Position *p*) [inline]

This function computes the new position after a 180 degree turn

Parameters

p Current Position

Returns

the new Postion after a 180 degree turn

See also

Position

Definition at line 210 of file Graph.java.

6.7.3.2 int pss::server::Graph::find_distance (Position *curr*, int *final_pos*, int *counter*) [inline]

Recurrsive function to compute distance (in terms of number of edges left to traverse), Memoization used for optimization

Parameters

curr Current position

final_pos Final position

counter Countdown to avoid getting stuck in cycles during recursive calls. Initialized to "infinity"

Returns

Distance to the destination

See also

Position

Definition at line 333 of file Graph.java.

6.7.3.3 void pss::server::Graph::init_graph ( ) [inline]

initialize the distance from one position to another position to infinity

initialize the position of the patients

initialize graph and set the respective neighbours

Definition at line 82 of file Graph.java.

6.7.3.4 Position pss::server::Graph::Left_turn (Position *p*) [inline]

THis function calculate the new position after a left turn.

Parameters

p Current Position

Returns

the new Postion after a left turn

See also

Position

Definition at line 185 of file Graph.java.

6.7.3.5 void pss::server::Graph::move_the_bot (Position *curr*, int *patient_id*) [inline]

Tester function to be used to print the path in the Graph Tester module

Parameters

curr Current bot position

patient_id Destination patient id

See also

Patient

Definition at line 308 of file Graph.java.

6.7.3.6 Position pss::server::Graph::pos_after_action (Position *curr*, int *action*) [inline]

Compute the new Position of the bot after a specific action FORWARD, LEFT_TURN, RIGHT_TURN or BACKWRD (encoded as an int)

Parameters

curr Current position

action Next action

Returns

New postion

See also

Position

Definition at line 273 of file Graph.java.

6.7.3.7 Position pss::server::Graph::Right_turn (Position *p*) [inline]

This function computes the new position after a right turn.

Parameters

p Current Position

Returns

the new Postion after a right turn

See also

Position

Definition at line 235 of file Graph.java.

6.7.3.8 int pss::server::Graph::search (int *patient_id*, Position *curr*) [inline]

Seraches the graph and find out the action to be taken on the path to reach the patient.

Parameters

patient_id Destination patient id

curr Current position

Returns

Next action to be taken on the path to reach the patient.

See also

Position

check if same as well

Definition at line 442 of file Graph.java.

6.7.3.9 Position pss::server::Graph::Straight (Position *p*) [inline]

This function computes the new position if no turn is taken and the bot just moves straight upto the next cross.

Parameters

p Current Position

Returns

the new Postion after moving straight in the direction it is facing

See also

Position

Definition at line 260 of file Graph.java.

6.7.4 Member Data Documentation

6.7.4.1 final int pss::server::Graph::BACKWARD = 3 [static]

Definition at line 53 of file Graph.java.

6.7.4.2 final int pss::server::Graph::BOT_POLLING = 5 [static]

Definition at line 53 of file Graph.java.

6.7.4.3 boolean pss::server::Graph::DEBUG = false [package]

Debugging flags.

Definition at line 71 of file Graph.java.

6.7.4.4 int [ ][ ][ ] pss::server::Graph::distance_pos_ori_pos = new int[16][4][16]

Array used for storing the memoization results of the distance calculation algorithm

The statement

distance_pos_ori_pos[a][b][c] = d

is to be interpreted as follows.

starting at "a" facing "b"; in order to reach "c" you need to make "d" moves.

Definition at line 67 of file Graph.java.

6.7.4.5 final int pss::server::Graph::EAST = 2 [static]

Definition at line 54 of file Graph.java.

6.7.4.6 final int pss::server::Graph::FINISH = 5 [static]

Definition at line 53 of file Graph.java.

6.7.4.7 int [ ][ ][ ] pss::server::Graph::grph = new int[16][4][4]

first co-ordinate denotes the position of the bot.

Second denote the orientation

0 = right,1=left,2=up,3=down

third index denotes the diretcion in which bot will turn.

0=right,1=left,2=straight,3=backward.

(A straight turn means no turn)

The statement

grph[a][b][c] = d

is to be interpreted as follows.

at "a" facing "b", if you turn "c" you will be facing "d"

Definition at line 52 of file Graph.java.

6.7.4.8 final int pss::server::Graph::INFINTY = 25 [static]

Definition at line 59 of file Graph.java.

6.7.4.9 final int pss::server::Graph::LEFT = 1 [static]

Definition at line 53 of file Graph.java.

6.7.4.10 int pss::server::Graph::no_calls = 0 [package]

Debugging variable for number of recursive calls of distance computing function

Definition at line 75 of file Graph.java.

6.7.4.11 final int pss::server::Graph::NOPATH = -2 [static]

Definition at line 53 of file Graph.java.

6.7.4.12 final int pss::server::Graph::NORTH = 0 [static]

Definition at line 54 of file Graph.java.

6.7.4.13 final int pss::server::Graph::PATIENT_POLLING = 6 [static]

Definition at line 53 of file Graph.java.

6.7.4.14 Map<Integer, Integer> pss::server::Graph::patientPos = new HashMap<Integer, Integer>() [static, package]

store the node location (room no.) of patients according to patient id.

Definition at line 39 of file Graph.java.

6.7.4.15 final int pss::server::Graph::RIGHT = 0 [static]

Definition at line 53 of file Graph.java.

6.7.4.16 final int pss::server::Graph::SERVER_ID1 = -2 [static]

ID of server location for bot 1

Definition at line 430 of file Graph.java.

6.7.4.17 final int pss::server::Graph::SERVER_ID1_orientation = Graph.SOUTH [static]

Initial orientation of bot 1.

Definition at line 421 of file Graph.java.

6.7.4.18 final int pss::server::Graph::SERVER_ID1_position = 14 [static]

Initial location of bot 1.

Definition at line 413 of file Graph.java.

6.7.4.19 final int pss::server::Graph::SERVER_ID2 = -3 [static]

ID of server location for bot 2

Definition at line 434 of file Graph.java.

6.7.4.20 final int pss::server::Graph::SERVER_ID2_orientation = Graph.NORTH [static]

Initial orientation of bot 1.

Definition at line 425 of file Graph.java.

6.7.4.21 final int pss::server::Graph::SERVER_ID2_position = 15 [static]

Initial location of bot 2.

Definition at line 417 of file Graph.java.

6.7.4.22 final int pss::server::Graph::SOUTH = 1 [static]

Definition at line 54 of file Graph.java.

6.7.4.23 final int pss::server::Graph::STRAIGHT = 2 [static]

Definition at line 53 of file Graph.java.

6.7.4.24 final int pss::server::Graph::WEST = 3 [static]

Definition at line 54 of file Graph.java.

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

- src/SERVER_CODE/pss/server/Graph.java

6.8 pss::server::test::GraphTester Class Reference

Static Public Member Functions

- static void main (String[ ] args)

6.8.1 Detailed Description

Module to test correctness of pathfinding algorithm on graph. Specify an initial position and the destination patient id (Hardcoded in main). Prints out the sequence of action bot must take to follow the computed path.

Definition at line 33 of file GraphTester.java.

6.8.2 Member Function Documentation

6.8.2.1 static void pss::server::test::GraphTester::main (String[ ] *args*) [inline, static]

Definition at line 35 of file GraphTester.java.

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

- src/SERVER_CODE/pss/server/test/GraphTester.java

6.9 pss::server::test::Patient_simulator Class Reference

Static Public Member Functions

- static void main (String[ ] args)

6.9.1 Detailed Description

Simulates a patient sending requests via zigbee to the main server

Definition at line 32 of file Patient_simulator.java.

6.9.2 Member Function Documentation

6.9.2.1 static void pss::server::test::Patient_simulator::main (String[] *args*) [inline, static]

Definition at line 34 of file Patient_simulator.java.

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

- src/SERVER_CODE/pss/server/test/Patient_simulator.java

6.10 pss::server::scheduling::PollingThread Class Reference

Public Member Functions

- PollingThread (CommunicationAPI capi)
- Character poll_next (int poll_bot_or_patient)

Public Attributes

- int poll_id = Bot.bot1
- int number_bots = 0

Static Public Attributes

- static final int poll_patient = 0
- static final int poll_bot = 1
- static int bot_or_patient = poll_bot
- static final Boolean DEBUG = false
- static int BOT_POLLING = 5
- static int PATIENT_POLLING = 128

Package Attributes

- CommunicationAPI capi

6.10.1 Detailed Description

This thread cyclically polls all the entities in the system (bots/ patients) asking about their status and returning their current status which the provide over Zigbee encoded in an 8 bit character

Definition at line 37 of file PollingThread.java.

6.10.2 Constructor & Destructor Documentation

6.10.2.1 pss::server::scheduling::PollingThread::PollingThread (CommunicationAPI *capi*) [inline]

Constructor

Parameters

capi Communication API

See also

CommunicationAPI

Definition at line 88 of file PollingThread.java.

6.10.3 Member Function Documentation

6.10.3.1 Character pss::server::scheduling::PollingThread::poll_next (int *poll_bot_or_patient*) [inline]

Poll the next entity bot or patient

Parameters

poll_bot_or_patient

Returns

reponse from the entity encoded as a character

Definition at line 104 of file PollingThread.java.

6.10.4 Member Data Documentation

6.10.4.1 int pss::server::scheduling::PollingThread::bot_or_patient = poll_bot [static]

Indicates who (patient or bot) must be polled in the next turn

Definition at line 64 of file PollingThread.java.

6.10.4.2 int pss::server::scheduling::PollingThread::BOT_POLLING = 5 [static]

8 bit encoding of "POLL" message for a bot

Definition at line 72 of file PollingThread.java.

6.10.4.3 CommunicationAPI pss::server::scheduling::PollingThread::capi [package]

Communication API Object

See also

CommunicationAPI

Definition at line 43 of file PollingThread.java.

6.10.4.4 final Boolean pss::server::scheduling::PollingThread::DEBUG = false [static]

Debugging flag

Definition at line 68 of file PollingThread.java.

6.10.4.5 int pss::server::scheduling::PollingThread::number_bots = 0

Number of serving bots

Definition at line 51 of file PollingThread.java.

6.10.4.6 int pss::server::scheduling::PollingThread::PATIENT_POLLING = 128 [static]

8 bit encoding of "POLL" message for a patient (0x80)

Definition at line 76 of file PollingThread.java.

6.10.4.7 final int pss::server::scheduling::PollingThread::poll_bot = 1 [static]

Flag indicating that in the next turn bot must be polled

Definition at line 59 of file PollingThread.java.

6.10.4.8 int pss::server::scheduling::PollingThread::poll_id = Bot.bot1

ID of the bot to be polled in the next loop

Definition at line 47 of file PollingThread.java.

6.10.4.9 final int pss::server::scheduling::PollingThread::poll_patient = 0 [static]

Flag indicating that in the next turn patient must be polled

Definition at line 55 of file PollingThread.java.

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

- src/SERVER_CODE/pss/server/scheduling/PollingThread.java

6.11 pss::server::Position Class Reference

Public Member Functions

- Position (int present, int orientation)
- Position ()

Public Attributes

- int present
- int orientation

6.11.1 Detailed Description

This class encodes the information regarding the position of the bot

present : Last node on which the bot was present

orientation : it is facing NORTH, EAST,SOUTH, WEST

Definition at line 29 of file Position.java.

6.11.2 Constructor & Destructor Documentation

6.11.2.1 pss::server::Position::Position (int *present,* int *orientation*) [inline]

Constructor

Parameters

present Location (Integer as seen on the Arena map)

orientation Orientation (NORTH, SOUTH, EAST or WEST) encoded as an integer

See also

Graph

Definition at line 46 of file Position.java.

6.11.2.2 pss::server::Position::Position ( ) [inline]

Default constructor

Definition at line 54 of file Position.java.

6.11.3 Member Data Documentation

6.11.3.1 int pss::server::Position::orientation

Orientation (NORTH, SOUTH, EAST or WEST) encoded as an integer

Definition at line 38 of file Position.java.

6.11.3.2 int pss::server::Position::present

Location (Integer as seen on the Arena map)

Definition at line 34 of file Position.java.

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

- src/SERVER_CODE/pss/server/Position.java

6.12 pss::serialcomm::CommunicationAPI::Receiver Class Reference

Inherits gnu::io::SerialPortEventListener.

Public Member Functions

- Receiver ()
- void run ()
- void serialEvent (SerialPortEvent event)

6.12.1 Detailed Description

Implementation of the thread which receives. Interrupt Based to provide maximum concurrency.

Definition at line 197 of file CommunicationAPI.java.

6.12.2 Constructor & Destructor Documentation

6.12.2.1 pss::serialcomm::CommunicationAPI::Receiver::Receiver ( ) [inline]

Definition at line 199 of file CommunicationAPI.java.

6.12.3 Member Function Documentation

6.12.3.1 void pss::serialcomm::CommunicationAPI::Receiver::run ( ) [inline]

Definition at line 202 of file CommunicationAPI.java.

6.12.3.2 void pss::serialcomm::CommunicationAPI::Receiver::serialEvent (SerialPortEvent *event*) [inline]

Other events are not interesting.

Definition at line 211 of file CommunicationAPI.java.

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

- src/SERVER_CODE/pss/serialcomm/CommunicationAPI.java

6.13 pss::server::RequestHandler Class Reference

Public Member Functions

- RequestHandler ()
- void execute_bot_Command (char command)
- void bot_blocked (int bot_id)
- void buzzer_off (int bot_id)
- void assign_if_request_available (int bot_id)
- void assign_bot_req (int bot_id, int req_id)
- void request_completed (int bot_id)
- int get_new_request_id ()
- int get_a_free_bot ()
- void print_patient_command (char comm)
- void execute_patient_request (char comm)

Static Public Member Functions

- static void main (String args[ ])

Static Public Attributes

- static final Boolean DEBUG = false
- static int no_bots = 0
- static CommunicationAPI capi
- static final int BLOCKED = 0
- static final int ACK = 1
- static final int IN_PROGRESS = 2
- static final int WATER = 0
- static final int MEDICINE = 1
- static int no_request = 0
- static int request_absent = 127

Package Attributes

- DBHandler dbh

Static Package Attributes

- static Map< Integer, Bot > botList = new HashMap<Integer, Bot>()
- static Graph g = new Graph()

6.13.1 Detailed Description

The main method of this class listens for the request from patients and bots and performs the neccesary book-keeping (Adding/Modifying database). Then, in case of patients request, it assigns them to a free bot if such a bot is available. When bot finished serving a patient, it returns to the server and statys there until it gets another request from the server. In case of bot's response, server computes the next action bot has to perform and send the appropriate message.

Definition at line 101 of file RequestHandler.java.

6.13.2 Constructor & Destructor Documentation

6.13.2.1 pss::server::RequestHandler::RequestHandler ( ) [inline]

This constructor initializes database, opens the communication interface and call functions to add patients into the database and bots into local variables.

See also

DBHandler
CommunicationAPI

Definition at line 130 of file RequestHandler.java.

6.13.3 Member Function Documentation

6.13.3.1 void pss::server::RequestHandler::assign_bot_req (int *bot_id*, int *req_id*) [inline]

Assign request to the bot

Set the use_or_not bit of the bot to be true.

Make the bot turn by 180 degree

Set the use_or_not bit of the bot to be true

Set the pid field of the bot to the id of the patient who made the request

Add the request id,bot id tuple to the assignment table of the database and update the status of the request in the database to serving

Parameters

bot_id bot which is free

req_id pending request

Definition at line 424 of file RequestHandler.java.

6.13.3.2 void pss::server::RequestHandler::assign_if_request_available (int *bot_id*) [inline]

When the bot is free, search if there is any pending request in the database. If more than one request are pending use FIFO and assign that request to the bot with id bot_id If no such request are pending then set the use_or_not flag of bot to be false

Parameters

bot_id bot which is free

Definition at line 393 of file RequestHandler.java.

6.13.3.3 void pss::server::RequestHandler::bot_blocked (int *bot_id*) [inline]

Sends a command to the bot with id bot_id to start a buzzer

Parameters

bot_id

Definition at line 373 of file RequestHandler.java.

6.13.3.4 void pss::server::RequestHandler::buzzer_off (int *bot_id*) [inline]

Sends a command to the bot with id bot_id to stop the buzzer

Parameters

bot_id

Definition at line 385 of file RequestHandler.java.

6.13.3.5 void pss::server::RequestHandler::execute_bot_Command (char *command*) [inline]

Command message has the bot id encoded in the first three bits and message in rest of the firve bits

We have maintained the current position and the the patient it is searching.

Based on the status message returned from the bot following actions are performed

BLOCKED : send a command to bot to start a buzzer.

ACK : Compute the next immediate hop to be taken by the bot and send the command for that. If, bot is at the destination then start return trip to the server

IN_PROGRESS : Don't do anything

If the bot has reached the server after serving the request, then assign it a new request if available else

If it is not at the server then search for the action(LEFT,RIGHT,STRAIGHT) the bot should take.

If the bot is at the destination then signal that the request is completed and start returning to the server

If the action is right then ask the bot to turn right

If the action is left then ask the bot to turn left

If the action is straight, then ask the bot to go straight

If the action is turn_back then ask to bot to turn back

If the bot is in middle of executing the previous instruction then do nothing

Parameters

command message from bot

Definition at line 283 of file RequestHandler.java.

6.13.3.6 void pss::server::RequestHandler::execute_patient_request (char *comm*) [inline]

This function processes the request received from the patient in the form of a char

Extracts the information about what item was requested for from the last 5 bits of comm.

Obtains a new unique id for this particular request

Adds a new entry to the request table containing request id, patient_id,item requested for and the status of the request as pending

Also check if a bot is free. If so then the id of the free bot is returned and is assigned to this request

Otherwise the function exits.

Parameters

comm character encoding request from the patient

Definition at line 542 of file RequestHandler.java.

6.13.3.7 int pss::server::RequestHandler::get_a_free_bot ( ) [inline]

Scans the BotList hashmap for an free bot.

Returns

If any free bot is present then return its ID
If no such bot is present then return -1

Definition at line 498 of file RequestHandler.java.

6.13.3.8 int pss::server::RequestHandler::get_new_request_id ( ) [inline]

This function returns a unique number as request ID each time it is called

Returns

New Request ID

Definition at line 490 of file RequestHandler.java.

6.13.3.9 static void pss::server::RequestHandler::main (String *args[ ]*) [inline, static]

Sends a polling message to the next bot. This message requests the bot to tell the server it's current status. This will keep running and will recieve commands from Communication API in the form of characters. It will decode the command in the following manner.First three bits denote the id of the sender. .If ID is either 0 or 1, its a bot else it is a patient In case of bot, the control is transfered to execute_bot_command otherwise the control is transfered to execute_patient_request

Definition at line 218 of file RequestHandler.java.

6.13.3.10 void pss::server::RequestHandler::print_patient_command (char *comm*) [inline]

Prints ID of patient who issued the command for debugging purposes

Parameters

comm Command issued by patient (8 bits) typecasted as a char

Definition at line 525 of file RequestHandler.java.

6.13.3.11 void pss::server::RequestHandler::request_completed (int *bot_id*) [inline]

Request is finished

Delete the request bot was serving from the assignment table

Update its status in the request table to done.

Set the destination of the bot to server

Parameters

bot_id

Definition at line 454 of file RequestHandler.java.

6.13.4 Member Data Documentation

6.13.4.1 final int pss::server::RequestHandler::ACK = 1 [static]

Flag for returning the status of the bot. Tells if the previous instruction assigned to the bot is complete.

Definition at line 260 of file RequestHandler.java.

6.13.4.2 final int pss::server::RequestHandler::BLOCKED = 0 [static]

Flag for returning the status of the bot. Tells if some external object is obstructing the path of bot

Definition at line 256 of file RequestHandler.java.

6.13.4.3 Map<Integer, Bot> pss::server::RequestHandler::botList = new HashMap<Integer, Bot>() [static, package]

A map between bot id and Bot object

Definition at line 112 of file RequestHandler.java.

6.13.4.4 CommunicationAPI pss::server::RequestHandler::capi [static]

Xbee functions including for sending and reciving messages is implemented in this class

See also

CommunicationAPI

Definition at line 122 of file RequestHandler.java.

6.13.4.5 DBHandler pss::server::RequestHandler::dbh [package]

Database Handler as defined in DBHandler

See also

DBHandler

Definition at line 108 of file RequestHandler.java.

6.13.4.6 final Boolean pss::server::RequestHandler::DEBUG = false [static]

Definition at line 103 of file RequestHandler.java.

6.13.4.7 Graph pss::server::RequestHandler::g = new Graph() [static, package]

Floor lay-out

See also

Graph

Definition at line 117 of file RequestHandler.java.

6.13.4.8 final int pss::server::RequestHandler::IN_PROGRESS = 2 [static]

Flag for returning the status of the bot. Tells if the previous instruction is still being executed.

Definition at line 264 of file RequestHandler.java.

6.13.4.9 final int pss::server::RequestHandler::MEDICINE = 1 [static]

Integer representing request for medicine.

Definition at line 480 of file RequestHandler.java.

6.13.4.10 int pss::server::RequestHandler::no_bots = 0 [static]

Definition at line 104 of file RequestHandler.java.

6.13.4.11 int pss::server::RequestHandler::no_request = 0 [static]

Global counter to assign unique request ID's

Definition at line 484 of file RequestHandler.java.

6.13.4.12 int pss::server::RequestHandler::request_absent = 127 [static]

Definition at line 531 of file RequestHandler.java.

6.13.4.13 final int pss::server::RequestHandler::WATER = 0 [static]

Integer representing request for water.

Definition at line 476 of file RequestHandler.java.

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

- src/SERVER_CODE/pss/server/RequestHandler.java

6.14 pss::serialcomm::CommunicationAPI::Sender Class Reference

Public Member Functions

- Sender (String messString)
- void run ()

Package Attributes

- String messageString

6.14.1 Detailed Description

To send a message, it creates a new thread which waits for the message to be sent. This has been done for maximum concurrency

Definition at line 130 of file CommunicationAPI.java.

6.14.2 Constructor & Destructor Documentation

6.14.2.1 pss::serialcomm::CommunicationAPI::Sender::Sender (String *messString*) [inline]

Definition at line 134 of file CommunicationAPI.java.

6.14.3 Member Function Documentation

6.14.3.1 void pss::serialcomm::CommunicationAPI::Sender::run ( ) [inline]

Definition at line 138 of file CommunicationAPI.java.

6.14.4 Member Data Documentation

6.14.4.1 String pss::serialcomm::CommunicationAPI::Sender::messageString [package]

Definition at line 132 of file CommunicationAPI.java.

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

- src/SERVER_CODE/pss/serialcomm/CommunicationAPI.java

6.15 pss::server::test::SimpleRead Class Reference

Inherits gnu::io::SerialPortEventListener.

Public Member Functions

- SimpleRead ()
- void run ()
- void serialEvent (SerialPortEvent event)

Static Public Member Functions

- static void main (String[ ] args)

Package Attributes

- InputStream inputStream
- SerialPort serialPort
- Thread readThread

Static Package Attributes

- static CommPortIdentifier portId
- static Enumeration portList

6.15.1 Detailed Description

Definition at line 29 of file SimpleRead.java.

6.15.2 Constructor & Destructor Documentation

6.15.2.1 pss::server::test::SimpleRead::SimpleRead ( ) [inline]

Definition at line 65 of file SimpleRead.java.

6.15.3 Member Function Documentation

6.15.3.1 static void pss::server::test::SimpleRead::main (String[] *args*) [inline, static]

Definition at line 38 of file SimpleRead.java.

6.15.3.2 void pss::server::test::SimpleRead::run ( ) [inline]

Definition at line 93 of file SimpleRead.java.

6.15.3.3 void pss::server::test::SimpleRead::serialEvent (SerialPortEvent *event*) [inline]

Definition at line 102 of file SimpleRead.java.

6.15.4 Member Data Documentation

6.15.4.1 InputStream pss::server::test::SimpleRead::inputStream [package]

Definition at line 33 of file SimpleRead.java.

6.15.4.2 CommPortIdentifier pss::server::test::SimpleRead::portId [static, package]

Definition at line 31 of file SimpleRead.java.

6.15.4.3 Enumeration pss::server::test::SimpleRead::portList [static, package]

Definition at line 32 of file SimpleRead.java.

6.15.4.4 Thread pss::server::test::SimpleRead::readThread [package]

Definition at line 35 of file SimpleRead.java.

6.15.4.5 SerialPort pss::server::test::SimpleRead::serialPort [package]

Definition at line 34 of file SimpleRead.java.

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

- src/SERVER_CODE/pss/server/test/SimpleRead.java

6.16 pss::server::test::SimpleWrite Class Reference

Static Public Member Functions

- static void main (String[ ] args)

Static Package Attributes

- static Enumeration portList
- static CommPortIdentifier portId
- static String messageString = "rohit kumar saraf"
- static SerialPort serialPort
- static OutputStream outputStream
- static boolean outputBufferEmptyFlag = false

6.16.1 Detailed Description

Definition at line 29 of file SimpleWrite.java.

6.16.2 Member Function Documentation

6.16.2.1 static void pss::server::test::SimpleWrite::main (String[] *args*) [inline, static]

Definition at line 38 of file SimpleWrite.java.

6.16.3 Member Data Documentation

6.16.3.1 String pss::server::test::SimpleWrite::messageString = "rohit kumar saraf" [static, package]

Definition at line 33 of file SimpleWrite.java.

6.16.3.2 boolean pss::server::test::SimpleWrite::outputBufferEmptyFlag = false [static, package]

Definition at line 36 of file SimpleWrite.java.

6.16.3.3 OutputStream pss::server::test::SimpleWrite::outputStream [static, package]

Definition at line 35 of file SimpleWrite.java.

6.16.3.4 CommPortIdentifier pss::server::test::SimpleWrite::portId [static, package]

Definition at line 32 of file SimpleWrite.java.

6.16.3.5 Enumeration pss::server::test::SimpleWrite::portList [static, package]

Definition at line 31 of file SimpleWrite.java.

6.16.3.6 SerialPort pss::server::test::SimpleWrite::serialPort [static, package]

Definition at line 34 of file SimpleWrite.java.

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

- src/SERVER_CODE/pss/server/test/SimpleWrite.java

6.17 pss::server::Utils Class Reference

Static Public Member Functions

- static char left_message (int bot_id)
- static char right_message (int bot_id)
- static char straight_message (int bot_id)
- static char back_message (int bot_id)

6.17.1 Detailed Description

Actual encoding of messages for each basic instruction LEFT,RIGHT,FORWARD and BACKWARD along with ID of bot which should perform this action into an into 8 bit number

Author

rohit

Definition at line 32 of file Utils.java.

6.17.2 Member Function Documentation

6.17.2.1 static char pss::server::Utils::back_message (int *bot_id*) [inline, static]

Gives the char representing the instruction to tell bot with id bot_is to turn backward 180 degrees

Parameters

bot_id

Returns

char encoding of the instruction

See also

Bot

Definition at line 73 of file Utils.java.

6.17.2.2 static char pss::server::Utils::left_message (int *bot_id*) [inline, static]

Gives the char representing the instruction to tell bot with id bot_is to turn left

Parameters

bot_id

Returns

char encoding of the instruction

See also

Bot

Definition at line 40 of file Utils.java.

6.17.2.3 static char pss::server::Utils::right_message (int *bot_id*) [inline, static]

Gives the char representing the instruction to tell bot with id bot_is to turn right

Parameters

bot_id

Returns

char encoding of the instruction

See also

Bot

Definition at line 51 of file Utils.java.

6.17.2.4 static char pss::server::Utils::straight_message (int *bot_id*) [inline, static]

Gives the char representing the instruction to tell bot with id bot_is to move straight

Parameters

bot_id

Returns

char encoding of the instruction

See also

Bot

Definition at line 62 of file Utils.java.

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

- src/SERVER_CODE/pss/server/Utils.java

7 File Documentation

7.1 src/BOT_CODE/bot.c File Reference

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include "bot_motion.h"
```

Defines

- #define BIT_MACROS

- #define SetBit(x, b) ((x)|=(b))
- #define GetBit(x, b) ((x)&(b))
- #define ResetBit(x, b) ((x)&=(~(b)))
- #define NOTHING 10
- #define GO_UPTO_CROSS 2
- #define TURN_RIGHT 0
- #define TURN_LEFT 1
- #define TURN_AROUND 3
- #define POLLING 5
- #define ID_MASK 0xE0
- #define INST_MASK 0x1F
- #define MY_ID 0x00

Functions

- void USART_Init (void)
- SIGNAL (SIG_USART0_RECV)
- void init_devices_1 ()
- int main ()

7.1.1 Define Documentation

7.1.1.1 #define BIT_MACROS

Definition at line 29 of file bot.c.

7.1.1.2 #define GetBit(*x*, *b*) ((x)&(b))

Definition at line 31 of file bot.c.

7.1.1.3 #define GO_UPTO_CROSS 2

Definition at line 36 of file bot.c.

7.1.1.4 #define ID_MASK 0xE0

Definition at line 42 of file bot.c.

7.1.1.5 #define INST_MASK 0x1F

Definition at line 43 of file bot.c.

7.1.1.6 #define MY_ID 0x00

Definition at line 45 of file bot.c.

7.1.1.7 #define NOTHING 10

Definition at line 35 of file bot.c.

7.1.1.8 #define POLLING 5

Definition at line 40 of file bot.c.

7.1.1.9 #define ResetBit(*x, b*) ((x)&=(~(b)))

Definition at line 32 of file bot.c.

7.1.1.10 #define SetBit(*x, b*) ((x)|=(b))

Definition at line 30 of file bot.c.

7.1.1.11 #define TURN_AROUND 3

Definition at line 39 of file bot.c.

7.1.1.12 #define TURN_LEFT 1

Definition at line 38 of file bot.c.

7.1.1.13 #define TURN_RIGHT 0

Definition at line 37 of file bot.c.

7.1.2 Function Documentation

7.1.2.1 void init_devices_1 ( )

Inits the Zigbee module

Definition at line 116 of file bot.c.

7.1.2.2 int main (void)

Processes the requests given by the server in an infinite loop. Sets the status at appropriate times (which are later sent back to the server during polling).

Definition at line 126 of file bot.c.

7.1.2.3 SIGNAL (SIG_USART0_RECV)

ISR for receive complete interrupt Replies back to the polling/server instruction

Definition at line 74 of file bot.c.

7.1.2.4 void USART_Init (void)

USART0 initialization for Zigbee communication. desired baud rate:9600 actual baud rate:9600 (0.0%) char size: 8 bit parity: Disabled

Definition at line 60 of file bot.c.

7.2 src/BOT_CODE/bot.c

```
00001 /*
00002  * ============================================================================
       =======
00003  *
00004  *       Filename:  bot.c
00005  *
00006  *           Date:  31st March, 2010
00007  *
00008  *        Version:  2.1
00009  *       Revision:  2.1
00010  *       Compiler:  gcc-avr
00011  *
00012  *        Authors:  Pritish Kamath, pritish.kamath@iitb.ac.in
00013  *                  Rohit Saraf   , rohitsaraf@iitb.ac.in
00014  *                  Ashish Mathew , ashishmathew@iitb.ac.in
00015  *                  Vivek Madan   , vivekmadan@iitb.ac.in
00016  *
00017  *        Company:  IIT Bombay
00018  *      Copyright:  ERTS Lab, IIT Bombay
00019  *
00020  * ============================================================================
       =======
00021  */
00022
00023 #include <avr/io.h>
00024 #include <avr/interrupt.h>
00025 #include <util/delay.h>
00026 #include "bot_motion.h"
00027
```

```
00028 #ifndef BIT_MACROS
00029 #define BIT_MACROS
00030 #define SetBit(x,b) ((x)|=(b))
00031 #define GetBit(x,b) ((x)&(b))
00032 #define ResetBit(x,b) ((x)&=(~(b)))
00033 #endif
00034
00035 #define NOTHING         10
00036 #define GO_UPTO_CROSS   2
00037 #define TURN_RIGHT      0
00038 #define TURN_LEFT       1
00039 #define TURN_AROUND     3
00040 #define POLLING         5
00041
00042 #define ID_MASK         0xE0
00043 #define INST_MASK       0x1F
00044
00045 #define MY_ID           0x00  //Specific for each bot.
00046
00047 static volatile char data = 0;
00048 static volatile char instruction = 0;
00049
00050 static volatile ACTION = NOTHING;
00051
00060 void USART_Init(void)
00061 {
00062   UCSR0B = 0x00;
00063   UCSR0A = 0x00;
00064   UCSR0C = 0x06;
00065   UBRR0L = 0x47;
00066   UBRR0H = 0x00;
00067   UCSR0B = 0x98;
00068 }
00069
00074 SIGNAL(SIG_USART0_RECV)
00075 {
00076    data = UDR0;
00077    _delay_ms(10);
00078    if(GetBit(data,ID_MASK) == MY_ID)
00079    {
00080       PORTJ = 0xff;
00081       instruction = GetBit(data,INST_MASK);
00082       if(instruction == POLLING){
00083          PORTJ = 5;
00084          UDR0 = (MY_ID | status);
00085       }
00086       else if(instruction==GO_UPTO_CROSS)
00087       {
00088          PORTJ = 1;
00089          status = PROCESSING;
00090          ACTION = GO_UPTO_CROSS;
00091       }
00092       else if(instruction==TURN_RIGHT)
00093       {
00094          PORTJ = 2;
00095          status = PROCESSING;
00096          ACTION = TURN_RIGHT;
00097       }
00098       else if(instruction==TURN_LEFT)
00099       {
00100          PORTJ = 3;
00101          status = PROCESSING;
00102          ACTION = TURN_LEFT;
00103       }
00104       else if(instruction==TURN_AROUND)
00105       {
00106          PORTJ = 4;
```

```
00107           status = PROCESSING;
00108           ACTION = TURN_AROUND;
00109       }
00110    }
00111 }
00112
00116 void init_devices_1(){
00117    cli();
00118    USART_Init();
00119    sei();
00120 }
00121
00126 int main()
00127 {
00128    init_devices_1();
00129    init_devices();
00130
00131    lcd_set_4bit();
00132    lcd_init();
00133    DDRJ = 0xff;
00134    PORTJ = 0xf0;
00135
00136    left_position_encoder_interrupt_init();
00137    right_position_encoder_interrupt_init();
00138
00139    unsigned char q = 0;
00140    for(q = 0;q<10;q++){
00141       Left_white_line = ADC_Conversion(3);    //Getting data of Left WL Sensor
00142       Center_white_line = ADC_Conversion(2); //Getting data of Center WL Sensor
00143       Right_white_line = ADC_Conversion(1);  //Getting data of Right WL Sensor
00144    }
00145
00146    while(1)
00147    {
00148       if(ACTION == GO_UPTO_CROSS)
00149       {
00150          status = PROCESSING;
00151          go_upto_next_cross();
00152          instruction = 0;
00153          status = IDLE;
00154          ACTION = NOTHING;
00155       }
00156       else if(ACTION == TURN_RIGHT)
00157       {
00158          status = PROCESSING;
00159          turn_right();
00160          instruction = 0;
00161          status = IDLE;
00162          ACTION = NOTHING;
00163       }
00164       else if(ACTION == TURN_LEFT)
00165       {
00166          status = PROCESSING;
00167          turn_left();
00168          instruction = 0;
00169          status = IDLE;
00170          ACTION = NOTHING;
00171       }
00172       else if(ACTION == TURN_AROUND)
00173       {
00174          status = PROCESSING;
00175          turn_left();
00176          instruction = 0;
00177          status = IDLE;
00178          ACTION = NOTHING;
00179       }
00180    }
```

```
00181     return 0;
00182 }
```

7.3 src/BOT_CODE/bot.d File Reference

7.4 src/BOT_CODE/bot.d

```
00001 bot.o bot.d : src/BOT_CODE/bot.c src/BOT_CODE/bot_motion.h src/BOT_CODE/lcd.h
```

7.5 src/BOT_CODE/bot_2.c File Reference

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include "bot_motion.h"
```

Defines

- #define SetBit(x, b) ((x)|=(b))
- #define GetBit(x, b) ((x)&(b))
- #define ResetBit(x, b) ((x)&=(~(b)))
- #define NOTHING 10
- #define GO_UPTO_CROSS 2
- #define TURN_RIGHT 0
- #define TURN_LEFT 1
- #define TURN_AROUND 3
- #define POLLING 5
- #define ID_MASK 0xE0
- #define INST_MASK 0x1F
- #define MY_ID 0x20

Functions

- void USART_Init (void)
- SIGNAL (SIG_USART0_RECV)
- void init_devices_1 ()
- int main ()

7.5.1 Define Documentation

7.5.1.1 #define GetBit(*x, b*) ((x)&(b))

Definition at line 31 of file bot_2.c.

7.5.1.2 #define GO_UPTO_CROSS 2

Definition at line 36 of file bot_2.c.

7.5.1.3 #define ID_MASK 0xE0

Definition at line 42 of file bot_2.c.

7.5.1.4 #define INST_MASK 0x1F

Definition at line 43 of file bot_2.c.

7.5.1.5 #define MY_ID 0x20

Definition at line 45 of file bot_2.c.

7.5.1.6 #define NOTHING 10

Definition at line 35 of file bot_2.c.

7.5.1.7 #define POLLING 5

Definition at line 40 of file bot_2.c.

7.5.1.8 #define ResetBit(*x*, *b*) ((x)&=(~(b)))

Definition at line 32 of file bot_2.c.

7.5.1.9 #define SetBit(*x*, *b*) ((x)|=(b))

Definition at line 30 of file bot_2.c.

7.5.1.10 #define TURN_AROUND 3

Definition at line 39 of file bot_2.c.

7.5.1.11 #define TURN_LEFT 1

Definition at line 38 of file bot_2.c.

7.5.1.12 #define TURN_RIGHT 0

Definition at line 37 of file bot_2.c.

7.5.2 Function Documentation

7.5.2.1 void init_devices_1 ( )

Inits the Zigbee module

Definition at line 116 of file bot_2.c.

7.5.2.2 int main (void)

Processes the requests given by the server in an infinite loop. Sets the status at appropriate times (which are later sent back to the server during polling).

Definition at line 126 of file bot_2.c.

7.5.2.3 SIGNAL (SIG_USART0_RECV)

ISR for receive complete interrupt Replies back to the polling/server instruction

Definition at line 74 of file bot_2.c.

7.5.2.4 void USART_Init (void)

USART0 initialization for Zigbee communication. desired baud rate:9600 actual baud rate:9600 (0.0%) char size: 8 bit parity: Disabled

Definition at line 60 of file bot_2.c.

7.6 src/BOT_CODE/bot_2.c

```
00001 /*
00002  * ============================================================================
      =======
00003  *
00004  *       Filename:  bot_2.c
00005  *
00006  *           Date:  31st March, 2010
00007  *
00008  *        Version:  2.1
00009  *       Revision:  2.1
00010  *       Compiler:  gcc-avr
00011  *
00012  *        Authors:  Pritish Kamath, pritish.kamath@iitb.ac.in
```

```
00013  *                  Rohit Saraf   , rohitsaraf@iitb.ac.in
00014  *                  Ashish Mathew , ashishmathew@iitb.ac.in
00015  *                  Vivek Madan   , vivekmadan@iitb.ac.in
00016  *
00017  *        Company:  IIT Bombay
00018  *      Copyright:  ERTS Lab, IIT Bombay
00019  *
00020  * ================================================================================
       =======
00021  */
00022
00023 #include <avr/io.h>
00024 #include <avr/interrupt.h>
00025 #include <util/delay.h>
00026 #include "bot_motion.h"
00027
00028 #ifndef BIT_MACROS
00029 #define BIT_MACROS
00030 #define SetBit(x,b) ((x)|=(b))
00031 #define GetBit(x,b) ((x)&(b))
00032 #define ResetBit(x,b) ((x)&=(~(b)))
00033 #endif
00034
00035 #define NOTHING         10
00036 #define GO_UPTO_CROSS   2
00037 #define TURN_RIGHT      0
00038 #define TURN_LEFT       1
00039 #define TURN_AROUND     3
00040 #define POLLING         5
00041
00042 #define ID_MASK         0xE0
00043 #define INST_MASK       0x1F
00044
00045 #define MY_ID           0x20  //Specific for each bot.
00046
00047 static volatile char data = 0;
00048 static volatile char instruction = 0;
00049
00050 static volatile ACTION = NOTHING;
00051
00060 void USART_Init(void)
00061 {
00062   UCSR0B = 0x00;
00063   UCSR0A = 0x00;
00064   UCSR0C = 0x06;
00065   UBRR0L = 0x47;
00066   UBRR0H = 0x00;
00067   UCSR0B = 0x98;
00068 }
00069
00074 SIGNAL(SIG_USART0_RECV)
00075 {
00076    data = UDR0;
00077    _delay_ms(10);
00078    if(GetBit(data,ID_MASK) == MY_ID)
00079    {
00080       PORTJ = 0xff;
00081       instruction = GetBit(data,INST_MASK);
00082       if(instruction == POLLING){
00083          PORTJ = 5;
00084          UDR0 = (MY_ID | status);
00085       }
00086       else if(instruction==GO_UPTO_CROSS)
00087       {
00088          PORTJ = 1;
00089          status = PROCESSING;
00090          ACTION = GO_UPTO_CROSS;
```

```
00091       }
00092       else if(instruction==TURN_RIGHT)
00093       {
00094          PORTJ = 2;
00095          status = PROCESSING;
00096          ACTION = TURN_RIGHT;
00097       }
00098       else if(instruction==TURN_LEFT)
00099       {
00100          PORTJ = 3;
00101          status = PROCESSING;
00102          ACTION = TURN_LEFT;
00103       }
00104       else if(instruction==TURN_AROUND)
00105       {
00106          PORTJ = 4;
00107          status = PROCESSING;
00108          ACTION = TURN_AROUND;
00109       }
00110    }
00111 }
00112
00116 void init_devices_1(){
00117    cli();
00118    USART_Init();
00119    sei();
00120 }
00121
00126 int main()
00127 {
00128    init_devices_1();
00129    init_devices();
00130
00131    lcd_set_4bit();
00132    lcd_init();
00133    DDRJ = 0xff;
00134    PORTJ = 0xf0;
00135
00136    left_position_encoder_interrupt_init();
00137    right_position_encoder_interrupt_init();
00138
00139    unsigned char q = 0;
00140    for(q = 0;q<10;q++){
00141       Left_white_line = ADC_Conversion(3);   //Getting data of Left WL Sensor
00142       Center_white_line = ADC_Conversion(2); //Getting data of Center WL Sensor
00143       Right_white_line = ADC_Conversion(1);  //Getting data of Right WL Sensor
00144    }
00145
00146    while(1)
00147    {
00148       if(ACTION == GO_UPTO_CROSS)
00149       {
00150          status = PROCESSING;
00151          go_upto_next_cross();
00152          instruction = 0;
00153          status = IDLE;
00154          ACTION = NOTHING;
00155       }
00156       else if(ACTION == TURN_RIGHT)
00157       {
00158          status = PROCESSING;
00159          turn_right();
00160          instruction = 0;
00161          status = IDLE;
00162          ACTION = NOTHING;
00163       }
00164       else if(ACTION == TURN_LEFT)
```

```
00165       {
00166          status = PROCESSING;
00167          turn_left();
00168          instruction = 0;
00169          status = IDLE;
00170          ACTION = NOTHING;
00171       }
00172       else if(ACTION == TURN_AROUND)
00173       {
00174          status = PROCESSING;
00175          turn_left();
00176          instruction = 0;
00177          status = IDLE;
00178          ACTION = NOTHING;
00179       }
00180    }
00181    return 0;
00182 }
```

7.7 src/BOT_CODE/bot_motion.h File Reference

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <math.h>
#include "lcd.h"
```

Defines

- #define SetBit(x, b) ((x)|=(b))
- #define GetBit(x, b) ((x)&(b))
- #define ResetBit(x, b) ((x)&=(~(b)))
- #define FCPU 11059200ul
- #define W_THRESHOLD 0x0f
- #define W_THRESHOLD_STOP 0x08
- #define ROTATE_THRESHOLD 0x0f
- #define LEFT_SENSOR 3
- #define CENTER_SENSOR 2
- #define RIGHT_SENSOR 1
- #define FRONT_IR_SENSOR 6
- #define CONT_BLACK 5
- #define IDLE 1
- #define PROCESSING 2
- #define BLOCKED 0

Functions

- void port_init ()
- void timer5_init ()
- void velocity (unsigned char, unsigned char)
- void motors_delay ()
- unsigned char ADC_Conversion (unsigned char)

- void lcd_port_config (void)
- void left_position_encoder_interrupt_init (void)
- void right_position_encoder_interrupt_init (void)
- ISR (INT4_vect)
- ISR (INT5_vect)
- void reset_shaft_counters ()
- void adc_pin_config (void)
- void motion_pin_config (void)
- void adc_init ()
- void print_sensor (char row, char coloumn, unsigned char channel)
- void motion_set (unsigned char Direction)
- void forward (void)
- void stop (void)
- void init_devices (void)
- void print_sensor_data ()
- void read_sensors ()
- void buzzer_on (void)
- void buzzer_off (void)
- void turn_right ()
- void turn_left ()
- void go_distance (unsigned char x)
- void go_upto_next_cross ()

Variables

- unsigned char ADC_Value
- unsigned char flag = 0
- unsigned char Left_white_line = 0
- unsigned char Center_white_line = 0
- unsigned char Right_white_line = 0
- unsigned char Front_IR_Sensor = 0

7.7.1 Define Documentation

7.7.1.1 #define BLOCKED 0

Definition at line 49 of file bot_motion.h.

7.7.1.2 #define CENTER_SENSOR 2

Definition at line 41 of file bot_motion.h.

7.7.1.3 #define CONT_BLACK 5

Definition at line 45 of file bot_motion.h.

7.7.1.4 #define FCPU 11059200ul

Definition at line 34 of file bot_motion.h.

7.7.1.5 #define FRONT_IR_SENSOR 6

Definition at line 43 of file bot_motion.h.

7.7.1.6 #define GetBit(*x*, *b*) ((x)&(b))

Definition at line 28 of file bot_motion.h.

7.7.1.7 #define IDLE 1

Definition at line 47 of file bot_motion.h.

7.7.1.8 #define LEFT_SENSOR 3

Definition at line 40 of file bot_motion.h.

7.7.1.9 #define PROCESSING 2

Definition at line 48 of file bot_motion.h.

7.7.1.10 #define ResetBit(*x*, *b*) ((x)&=(~(b)))

Definition at line 29 of file bot_motion.h.

7.7.1.11 #define RIGHT_SENSOR 1

Definition at line 42 of file bot_motion.h.

7.7.1.12 #define ROTATE_THRESHOLD 0x0f

Definition at line 38 of file bot_motion.h.

7.7.1.13 #define SetBit(*x*, *b*) ((x)|=(b))

Definition at line 27 of file bot_motion.h.

7.7.1.14 #define W_THRESHOLD 0x0f

Definition at line 36 of file bot_motion.h.

7.7.1.15 #define W_THRESHOLD_STOP 0x08

Definition at line 37 of file bot_motion.h.

7.7.2 Function Documentation

7.7.2.1 unsigned char ADC_Conversion (unsigned char *Ch*)

ADC Conversion

Definition at line 200 of file bot_motion.h.

7.7.2.2 void adc_init ( )

Initialize the ADC module.

Definition at line 188 of file bot_motion.h.

7.7.2.3 void adc_pin_config (void)

Set ADC pin configuration

Definition at line 133 of file bot_motion.h.

7.7.2.4 void buzzer_off (void)

Switches the buzzer off.

Definition at line 315 of file bot_motion.h.

7.7.2.5 void buzzer_on (void)

Switches the buzzer on, to signal a block on the path. Changes status of the bot to BLOCKED, which is sent to the server (which in turn sends an SMS to inform the guards to take appropriate actions.)

Definition at line 303 of file bot_motion.h.

7.7.2.6 void forward (void)

Set bot direction forward.

Definition at line 253 of file bot_motion.h.

7.7.2.7 void go_distance (unsigned char *x*)

Go forward by a certain specified number of steps.

Definition at line 359 of file bot_motion.h.

7.7.2.8 void go_upto_next_cross ( )

Go forward upto the next intersection, while following a white line. Uses 7-fold scheme : (left, center, right) - Action (0,1,0) - Go Forward. (1,1,0) - Turn right (slightly) (1,0,0) - Turn right (hard) (0,1,1) - Turn left (slightly) (0,0,1) - Turn left (hard) (1,1,1) - Reached the intersection (0,0,0) - Recovery mode. Move in the direction of the last sensor that was on white line

Definition at line 396 of file bot_motion.h.

7.7.2.9 void init_devices (void)

Calls the init methods for all required devices.

Definition at line 269 of file bot_motion.h.

7.7.2.10 ISR (INT4_vect)

Interrupt handler for left shaft count change.

Definition at line 106 of file bot_motion.h.

7.7.2.11 ISR (INT5_vect)

Interrupt handler for right shaft count change.

Definition at line 115 of file bot_motion.h.

7.7.2.12 void lcd_port_config (void)

Function to configure LCD port all the LCD pin's direction set as output all the LCD pins are set to logic 0 except PORTC 7

Definition at line 74 of file bot_motion.h.

7.7.2.13 void left_position_encoder_interrupt_init (void)

Left shaft encoder init.

Definition at line 83 of file bot_motion.h.

7.7.2.14 void motion_pin_config (void)

Function to configure ports to enable robot's motion

Definition at line 144 of file bot_motion.h.

7.7.2.15 void motion_set (unsigned char *Direction*)

Function used for setting motor's direction

Definition at line 239 of file bot_motion.h.

7.7.2.16 void motors_delay ( )

7.7.2.17 void port_init ( )

Function to Initialize PORTS

Definition at line 155 of file bot_motion.h.

7.7.2.18 void print_sensor (char *row*, char *coloumn*, unsigned char *channel*)

Print Sensor Values At Desired Row And Coloumn Location on LCD

Definition at line 220 of file bot_motion.h.

7.7.2.19 void print_sensor_data ( )

Prints White line sensor values on the screen

Definition at line 281 of file bot_motion.h.

7.7.2.20 void read_sensors ( )

Reads all relevant sensor values and stores it in appropriate global variables.

Definition at line 291 of file bot_motion.h.

7.7.2.21 void reset_shaft_counters ( )

Reset shaft counters.

Definition at line 123 of file bot_motion.h.

7.7.2.22 void right_position_encoder_interrupt_init (void)

Right shaft encoder init.

Definition at line 94 of file bot_motion.h.

7.7.2.23 void stop (void)

Stop the bot

Definition at line 261 of file bot_motion.h.

7.7.2.24 void timer5_init ( )

Timer 5 initialised in PWM mode for velocity control Prescale:64 PWM 8bit fast, TOP=0x00FF Timer Frequency:674.988Hz

Definition at line 168 of file bot_motion.h.

7.7.2.25 void turn_left ( )

Turn left at an intersection.

Definition at line 343 of file bot_motion.h.

7.7.2.26 void turn_right ( )

Turn right at an intersection.

Definition at line 327 of file bot_motion.h.

7.7.2.27 void velocity (unsigned char *left_motor,* unsigned char *right_motor*)

Set velocity

Definition at line 230 of file bot_motion.h.

7.7.3 Variable Documentation

7.7.3.1 unsigned char ADC_Value

Definition at line 60 of file bot_motion.h.

7.7.3.2 unsigned char Center_white_line = 0

Definition at line 63 of file bot_motion.h.

7.7.3.3 unsigned char flag = 0

Definition at line 61 of file bot_motion.h.

7.7.3.4 unsigned char Front_IR_Sensor = 0

Definition at line 65 of file bot_motion.h.

7.7.3.5 unsigned char Left_white_line = 0

Definition at line 62 of file bot_motion.h.

7.7.3.6 unsigned char Right_white_line = 0

Definition at line 64 of file bot_motion.h.

7.8 src/BOT_CODE/bot_motion.h

```
00001 /*
00002  * ================================================================================
      =======
00003  *
00004  *       Filename:  bot_motion.h
00005  *
00006  *           Date:  31st March, 2010
00007  *
00008  *        Version:  2.1
00009  *       Revision:  2.1
00010  *       Compiler:  gcc-avr
00011  *
00012  *        Authors:  Pritish Kamath, pritish.kamath@iitb.ac.in
00013  *                  Rohit Saraf   , rohitsaraf@iitb.ac.in
00014  *                  Ashish Mathew , ashishmathew@iitb.ac.in
00015  *                  Vivek Madan   , vivekmadan@iitb.ac.in
00016  *
00017  *        Company:  IIT Bombay
00018  *      Copyright:  ERTS Lab, IIT Bombay
00019  *
00020  * ================================================================================
      =======
00021  */
00022
00023 #include <avr/io.h>
00024 #include <avr/interrupt.h>
00025 #include <util/delay.h>
00026
00027 #define SetBit(x,b) ((x)|=(b))
00028 #define GetBit(x,b) ((x)&(b))
00029 #define ResetBit(x,b) ((x)&=(~(b)))
00030
00031 #include <math.h> //included to support power function
00032 #include "lcd.h"
00033
00034 #define FCPU 11059200ul //defined here to make sure that program works properly
00035
00036 #define W_THRESHOLD 0x0f
00037 #define W_THRESHOLD_STOP 0x08
00038 #define ROTATE_THRESHOLD 0x0f //0x41
00039
00040 #define LEFT_SENSOR 3
```

```
00041 #define CENTER_SENSOR 2
00042 #define RIGHT_SENSOR 1
00043 #define FRONT_IR_SENSOR 6
00044
00045 #define CONT_BLACK 5
00046
00047 #define IDLE            1
00048 #define PROCESSING      2
00049 #define BLOCKED         0
00050
00051
00052 void port_init();
00053 void timer5_init();
00054 void velocity(unsigned char, unsigned char);
00055 void motors_delay();
00056
00057 static volatile unsigned char ShaftCountLeft = 0;
00058 static volatile unsigned char ShaftCountRight = 0;
00059 unsigned char ADC_Conversion(unsigned char);
00060 unsigned char ADC_Value;
00061 unsigned char flag = 0;
00062 unsigned char Left_white_line = 0;
00063 unsigned char Center_white_line = 0;
00064 unsigned char Right_white_line = 0;
00065 unsigned char Front_IR_Sensor=0;
00066
00067 static volatile char status = IDLE;
00068
00074 void lcd_port_config (void)
00075 {
00076  DDRC = DDRC | 0xF7; //all the LCD pin's direction set as output
00077  PORTC = PORTC & 0x80; // all the LCD pins are set to logic 0 except PORTC 7
00078 }
00079
00083 void left_position_encoder_interrupt_init(void)
00084 {
00085    cli();
00086    SetBit(EICRB,_BV(ISC41)); //The falling edge between two samples of INTn gener
      ates an interrupt request.
00087    SetBit(EIMSK,_BV(INT4)); //INT4 enable
00088    sei();
00089 }
00090
00094 void right_position_encoder_interrupt_init(void)
00095 {
00096    cli();
00097    SetBit(EICRB,_BV(ISC51)); //The falling edge between two samples of INTn gener
      ates an interrupt request.
00098    SetBit(EIMSK,_BV(INT5)); //INT5 enable
00099    sei();
00100 }
00101
00102
00106 ISR(INT4_vect)
00107 {
00108    ShaftCountLeft++;
00109 }
00110
00111
00115 ISR(INT5_vect)
00116 {
00117    ShaftCountRight++;
00118 }
00119
00123 void reset_shaft_counters()
00124 {
00125    ShaftCountLeft = 0;
```

```
00126    ShaftCountRight = 0;
00127 }
00128
00129
00133 void adc_pin_config (void)
00134 {
00135  DDRF = 0x00;
00136  PORTF = 0x00;
00137  DDRK = 0x00;
00138  PORTK = 0x00;
00139 }
00140
00144 void motion_pin_config (void)
00145 {
00146  DDRA = DDRA | 0x0F;
00147  PORTA = PORTA & 0xF0;
00148  DDRL = DDRL | 0x18;   //Setting PL3 and PL4 pins as output for PWM generation
00149  PORTL = PORTL | 0x18; //PL3 and PL4 pins are for velocity control using PWM.
00150 }
00151
00155 void port_init()
00156 {
00157    lcd_port_config();
00158    adc_pin_config();
00159    motion_pin_config();
00160 }
00161
00168 void timer5_init()
00169 {
00170    TCCR5B = 0x00;       //Stop
00171    TCNT5H = 0xFF;       //Counter higher 8-bit value to which OCR5xH value is com
      pared with
00172    TCNT5L = 0x01;       //Counter lower 8-bit value to which OCR5xH value is comp
      ared with
00173    OCR5AH = 0x00;       //Output compare register high value for Left Motor
00174    OCR5AL = 0xFF;       //Output compare register low value for Left Motor
00175    OCR5BH = 0x00;       //Output compare register high value for Right Motor
00176    OCR5BL = 0xFF;       //Output compare register low value for Right Motor
00177    OCR5CH = 0x00;       //Output compare register high value for Motor C1
00178    OCR5CL = 0xFF;       //Output compare register low value for Motor C1
00179    TCCR5A = 0xA9;       /*{COM5A1=1, COM5A0=0; COM5B1=1, COM5B0=0; COM5C1=1 COM5C
      0=0}
00180                         For Overriding normal port functionalit to OCRnA outputs.

00181                         {WGM51=0, WGM50=1} Along With WGM52 in TCCR5B for Selecti
      ng FAST PWM 8-bit Mode*/
00182    TCCR5B = 0x0B;       //WGM12=1; CS12=0, CS11=1, CS10=1 (Prescaler=64)
00183 }
00184
00188 void adc_init()
00189 {
00190    ADCSRA = 0x00;
00191    ADCSRB = 0x00;     //MUX5 = 0
00192    ADMUX = 0x20;      //Vref=5V external --- ADLAR=1 --- MUX4:0 = 0000
00193    ACSR = 0x80;
00194    ADCSRA = 0x86;     //ADEN=1 --- ADIE=1 --- ADPS2:0 = 1 1 0
00195 }
00196
00200 unsigned char ADC_Conversion(unsigned char Ch)
00201 {
00202    unsigned char a;
00203    if(Ch>7)
00204    {
00205       ADCSRB = 0x08;
00206    }
00207    Ch = Ch & 0x07;
00208    ADMUX= 0x20| Ch;
```

```
00209    ADCSRA = ADCSRA | 0x40;    //Set start conversion bit
00210    while((ADCSRA&0x10)==0);   //Wait for conversion to complete
00211    a=ADCH;
00212    ADCSRA = ADCSRA|0x10; //clear ADIF (ADC Interrupt Flag) by writing 1 to it
00213    ADCSRB = 0x00;
00214    return a;
00215 }
00216
00220 void print_sensor(char row, char coloumn,unsigned char channel)
00221 {
00222
00223    ADC_Value = ADC_Conversion(channel);
00224    lcd_print(row, coloumn, ADC_Value, 3);
00225 }
00226
00230 void velocity (unsigned char left_motor, unsigned char right_motor)
00231 {
00232    OCR5AL = (unsigned char)left_motor;
00233    OCR5BL = (unsigned char)right_motor;
00234 }
00235
00239 void motion_set (unsigned char Direction)
00240 {
00241  unsigned char PortARestore = 0;
00242
00243  Direction &= 0x0F;              // removing upper nibbel for the protection
00244  PortARestore = PORTA;           // reading the PORTA original status
00245  PortARestore &= 0xF0;           // making lower direction nibbel to 0
00246  PortARestore |= Direction;      // adding lower nibbel for forward command and r
      estoring the PORTA status
00247  PORTA = PortARestore;           // executing the command
00248 }
00249
00253 void forward (void)
00254 {
00255   motion_set (0x06);
00256 }
00257
00261 void stop (void)
00262 {
00263   motion_set (0x00);
00264 }
00265
00269 void init_devices (void)
00270 {
00271    cli();
00272    port_init();
00273    adc_init();
00274    timer5_init();
00275    sei();
00276 }
00277
00281 void print_sensor_data()
00282 {
00283       print_sensor(1,1,3); //Prints value of White Line Sensor1
00284       print_sensor(1,5,2); //Prints Value of White Line Sensor2
00285       print_sensor(1,9,1); //Prints Value of White Line Sensor3
00286 }
00287
00291 void read_sensors()
00292 {
00293    Left_white_line = ADC_Conversion(LEFT_SENSOR);
00294    Center_white_line = ADC_Conversion(CENTER_SENSOR);
00295    Right_white_line = ADC_Conversion(RIGHT_SENSOR);
00296    Front_IR_Sensor = ADC_Conversion(FRONT_IR_SENSOR);
00297 }
00298
```

```
00303 void buzzer_on (void)
00304 {
00305  unsigned char port_restore = 0;
00306  port_restore = PINC;
00307  port_restore = port_restore | 0x08;
00308  PORTC = port_restore;
00309  status = BLOCKED;
00310 }
00311
00315 void buzzer_off (void)
00316 {
00317  unsigned char port_restore = 0;
00318  port_restore = PINC;
00319  port_restore = port_restore & 0xF7;
00320  PORTC = port_restore;
00321  status = BLOCKED;
00322 }
00323
00327 void turn_right(){
00328    buzzer_off();
00329    motion_set(0x0A);
00330    velocity(100,100);
00331    _delay_ms(1000);
00332    while(1){
00333       print_sensor_data();
00334       read_sensors();
00335       if(Center_white_line < W_THRESHOLD) break;
00336    }
00337    velocity(0,0);
00338 }
00339
00343 void turn_left(){
00344    buzzer_off();
00345    motion_set(0x05);
00346    velocity(100,100);
00347    _delay_ms(1000);
00348    while(1){
00349       print_sensor_data();
00350       read_sensors();
00351       if(Center_white_line < W_THRESHOLD) break;
00352    }
00353    velocity(0,0);
00354 }
00355
00359 void go_distance(unsigned char x)
00360 {
00361    reset_shaft_counters();
00362    forward();
00363    velocity(100,100);
00364    PORTJ = 0x00;
00365    while(1){
00366       read_sensors();
00367       print_sensor_data();
00368       if( Front_IR_Sensor<0xF0)
00369       {
00370          stop();
00371          buzzer_on();
00372       }
00373       else
00374       {
00375          forward();
00376          buzzer_off();
00377       }
00378       if((ShaftCountLeft + ShaftCountRight)*5 > x*10)
00379          break;
00380    }
00381    velocity(0,0);
```

```
00382 }
00383
00396 void go_upto_next_cross(){
00397    char last_on = LEFT_SENSOR;
00398    char black_flag = 0;
00399    while(1)
00400    {
00401       PORTJ = PORTJ+1;
00402       if(Center_white_line<W_THRESHOLD_STOP && Left_white_line<W_THRESHOLD_STOP &
      & Right_white_line<W_THRESHOLD_STOP ){
00403          PORTJ = 0xAA;
00404          break;
00405       }
00406
00407       read_sensors();
00408
00409       flag=0;
00410       print_sensor_data();
00411
00412       if( Front_IR_Sensor<0xF0)
00413       {
00414          stop();
00415          buzzer_on();
00416       }
00417       //Sensor config : 010
00418       else if(Left_white_line > W_THRESHOLD && Center_white_line < W_THRESHOLD &&
       Right_white_line > W_THRESHOLD)
00419       {
00420          forward();
00421          velocity(150,150);
00422          black_flag = 0;
00423          buzzer_off();
00424       }
00425
00426       //Sensor config : 110
00427       else if(Left_white_line < W_THRESHOLD && Center_white_line < W_THRESHOLD &&
       Right_white_line > W_THRESHOLD)
00428       {
00429          forward();
00430          velocity(120,150);
00431          black_flag = 0;
00432          buzzer_off();
00433       }
00434
00435       //Sensor config : 100
00436       else if(Left_white_line < W_THRESHOLD && Center_white_line > W_THRESHOLD &&
       Right_white_line > W_THRESHOLD)
00437       {
00438          PORTA = 0x05;
00439          velocity(50,130);
00440          last_on = LEFT_SENSOR;
00441          black_flag = 0;
00442          buzzer_off();
00443       }
00444
00445       //Sensor config : 011
00446       else if(Left_white_line > W_THRESHOLD && Center_white_line < W_THRESHOLD &&
       Right_white_line < W_THRESHOLD)
00447       {
00448          forward();
00449          velocity(150,120);
00450          black_flag = 0;
00451          buzzer_off();
00452       }
00453
00454       //Sensor config : 001
00455       else if(Left_white_line > W_THRESHOLD && Center_white_line > W_THRESHOLD &&
```

```
       Right_white_line < W_THRESHOLD)
00456       {
00457          PORTA = 0x0A;
00458          velocity(130,50);
00459          last_on = RIGHT_SENSOR;
00460          black_flag = 0;
00461          buzzer_off();
00462       }
00463       //Sensor config : 000
00464       else
00465       {
00466          buzzer_off();
00467          if(black_flag >= CONT_BLACK)  {
00468             if(last_on == LEFT_SENSOR)
00469                motion_set(0x05);
00470             else if(last_on == RIGHT_SENSOR)
00471                motion_set(0x0A);
00472             velocity(100,100);
00473             while(1){
00474                print_sensor_data();
00475                read_sensors();
00476                if(Center_white_line < W_THRESHOLD) break;
00477             }
00478          }
00479          black_flag = (black_flag < CONT_BLACK)?black_flag+1:CONT_BLACK;
00480          forward();
00481          velocity(0,0);
00482          PORTJ = 0x99;
00483       }
00484    }
00485    velocity(0,0);
00486    go_distance(8);
00487 }
```

7.9 src/BOT_CODE/IR.c File Reference

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
```

Defines

- #define SetBit(x, b) ((x)|=(b))
- #define GetBit(x, b) ((x)&(b))
- #define ResetBit(x, b) ((x)&=(~(b)))
- #define Bit(x) (1<<(x))
- #define ID_MASK 0xE0
- #define INST_MASK 0x1F
- #define MY_ID 0x80
- #define CODESIZE 5

Functions

- unsigned char num_bits_matched (unsigned char a, unsigned char b)
- unsigned char patient_id (unsigned char code1[ ])
- void init_ports ()
- void USART_Init (void)

- ISR (SIG_USART0_RECV)
- void IR_Get_Input_Vector (void)
- unsigned char IR_read ()
- void learn ()
- void IR_init_devices ()
- int main (void)

Variables

- volatile unsigned char Pat [6][CODESIZE]
- volatile unsigned char code [CODESIZE]

7.9.1 Define Documentation

7.9.1.1 #define Bit(x) (1<<(x))

Definition at line 32 of file IR.c.

7.9.1.2 #define CODESIZE 5

Definition at line 48 of file IR.c.

7.9.1.3 #define GetBit(x, b) ((x)&(b))

Definition at line 30 of file IR.c.

7.9.1.4 #define ID_MASK 0xE0

Definition at line 35 of file IR.c.

7.9.1.5 #define INST_MASK 0x1F

Definition at line 36 of file IR.c.

7.9.1.6 #define MY_ID 0x80

Definition at line 38 of file IR.c.

7.9.1.7 #define ResetBit(x, b) ((x)&=(~(b)))

Definition at line 31 of file IR.c.

7.9.1.8 #define SetBit(x, b) ((x)|=(b))

Definition at line 29 of file IR.c.

7.9.2 Function Documentation

7.9.2.1 void init_ports ( )

Initialize all ports

Definition at line 95 of file IR.c.

7.9.2.2 void IR_Get_Input_Vector (void)

Function to read IR message from the detector Stores the 'IR-code' received in the code array (global).

The code received is 'Protocol Independent'. Although the TV remote being used is RC-5 based, it does not assume any specific protocol, as long as the frequency of the remote matches that of the TSOP sensor.

Definition at line 146 of file IR.c.

7.9.2.3 void IR_init_devices ( )

Initialise all the devices

Definition at line 218 of file IR.c.

7.9.2.4 unsigned char IR_read ( )

Returns the patient_ID of the last request that was received.

Definition at line 194 of file IR.c.

7.9.2.5 ISR (SIG_USART0_RECV)

ISR handler for Zigbee receive complete.

Definition at line 127 of file IR.c.

7.9.2.6 void learn ( )

Learns and remembers the patterns of TV remote buttons 1, 2, ... , 6.

Definition at line 201 of file IR.c.

7.9.2.7 int main (void)

Main Function Performs the job of receiving TV remote signals and stores it in the variable 'patient', which is later communicated to the server on request.

Definition at line 230 of file IR.c.

7.9.2.8 unsigned char num_bits_matched (unsigned char *a*, unsigned char *b*)

Number of common bits in given two bytes.

Definition at line 56 of file IR.c.

7.9.2.9 unsigned char patient_id (unsigned char *code1[]*)

Finds the patient_id for a specific IR code.

Definition at line 68 of file IR.c.

7.9.2.10 void USART_Init (void)

USART0 initialization for Zigbee communication. desired baud rate:9600 actual baud rate:9600 (0.0%) char size: 8 bit parity: Disabled

Definition at line 114 of file IR.c.

7.9.3 Variable Documentation

7.9.3.1 volatile unsigned char code[CODESIZE]

Definition at line 51 of file IR.c.

7.9.3.2 volatile unsigned char Pat[6][CODESIZE]

Definition at line 49 of file IR.c.

7.10 src/BOT_CODE/IR.c

```
00001 /*
00002  * ================================================================================
      =======
00003  *
00004  *       Filename:  IR.c
00005  *
00006  *           Date:  31st March, 2010
00007  *
00008  *        Version:  2.1
00009  *       Revision:  2.1
00010  *       Compiler:  gcc-avr
00011  *
00012  *        Authors:  Pritish Kamath, pritish.kamath@iitb.ac.in
```

```
00013  *                  Rohit Saraf   , rohitsaraf@iitb.ac.in
00014  *                  Ashish Mathew , ashishmathew@iitb.ac.in
00015  *                  Vivek Madan   , vivekmadan@iitb.ac.in
00016  *
00017  *        Company:  IIT Bombay
00018  *      Copyright:  ERTS Lab, IIT Bombay
00019  *
00020  * ================================================================================
       =======
00021  */
00022
00023 #include <avr/io.h>
00024 #include <avr/interrupt.h>
00025 #include <util/delay.h>
00026
00027 #ifndef BIT_MACROS
00028 #define BIT_MACROS
00029 #define SetBit(x,b) ((x)|=(b))
00030 #define GetBit(x,b) ((x)&(b))
00031 #define ResetBit(x,b) ((x)&=(~(b)))
00032 #define Bit(x) (1<<(x))
00033 #endif
00034
00035 #define ID_MASK         0xE0
00036 #define INST_MASK       0x1F
00037
00038 #define MY_ID           0x80  //Specific for patient polling bot.
00039
00040 static volatile unsigned char PATIENT_QUEUE[10];
00041 static volatile unsigned char PATIENT_QUEUE_CURR_PTR = 0;
00042 static volatile unsigned char PATIENT_QUEUE_TAIL_PTR = 0;
00043 static volatile unsigned char PATIENT_QUEUE_SIZE = 0;
00044
00045 static volatile char data = 0;
00046
00047 static volatile unsigned char patient = 0x07;
00048 #define CODESIZE 5
00049 volatile unsigned char Pat[6][CODESIZE];
00050 static volatile unsigned char got_intr = 0;
00051 volatile unsigned char code[CODESIZE];
00052
00056 unsigned char num_bits_matched(unsigned char a, unsigned char b)
00057 {
00058    char i;
00059    unsigned char num = 0;
00060    for(i=0; i<8; i++)
00061       if(GetBit(a,Bit(i)) == GetBit(b,Bit(i))) num++;
00062    return num;
00063 }
00064
00068 unsigned char patient_id(unsigned char code1[])
00069 {
00070    unsigned char i, j;
00071    unsigned char room = 0, max_count = 0, curr_count;
00072    for(i=0; i<6; i++)
00073    {
00074       curr_count = 0;
00075       for(j=0; j<CODESIZE; j++)
00076          {
00077             curr_count += num_bits_matched(code1[j],Pat[i][j]);
00078          }
00079       if(max_count <= curr_count)
00080          {
00081             room = i;
00082             max_count = curr_count;
00083          }
00084    }
```

```
00085    PORTJ = max_count;
00086    if(max_count > CODESIZE*6)
00087       return room+1;
00088    else
00089       return 0x07;
00090 }
00091
00095 void init_ports()
00096 {
00097    DDRA = 0x0F;
00098    PORTA = 0x00;
00099    DDRL = 0xff;
00100    PORTL = 0x00;
00101    PORTJ = 0x00;
00102    DDRJ  = 0xFF;
00103    PORTE = 0x00;
00104    DDRE  = 0x00;
00105 }
00106
00114 void USART_Init(void)
00115 {
00116   UCSR0B = 0x00; //disable while setting baud rate
00117   UCSR0A = 0x00;
00118   UCSR0C = 0x06;
00119   UBRR0L = 0x47; //set baud rate lo
00120   UBRR0H = 0x00; //set baud rate hi
00121   UCSR0B = 0x98;
00122 }
00123
00127 ISR(SIG_USART0_RECV)
00128 {
00129    data = UDR0;
00130    _delay_ms(100);
00131    if(GetBit(data,ID_MASK) == MY_ID)
00132    {
00133       if(patient == 0x07) UDR0 = 0x7f;
00134       else UDR0 = ((patient+1)<<5);
00135       patient = 0x07;
00136    }
00137 }
00138
00146 void IR_Get_Input_Vector (void)
00147 {
00148    while((PINE & 0x80) == 0x80);
00149    _delay_us(7400);
00150    unsigned char Pulse_counter=0, addr = 0;
00151
00152    while(Pulse_counter < 5)
00153    {
00154       _delay_us(1800);
00155       Pulse_counter++;
00156       if((PINE & 0x80) == 0x80)
00157       {
00158          addr = addr & ~(1 << (Pulse_counter-1));
00159       }
00160       else
00161       {
00162          addr = addr | (1 << (Pulse_counter-1));
00163       }
00164    }
00165
00166    unsigned char last = 0;
00167    unsigned char bitarray[CODESIZE];
00168    unsigned char cc = 0;
00169    while(cc < CODESIZE*8){
00170       if(GetBit(PINE,0x80)) SetBit(bitarray[cc/8],Bit(cc%8));
00171       else ResetBit(bitarray[cc/8],Bit(cc%8));
```

```
00172       last = PINE & 0x80;
00173       _delay_us(300);
00174       cc++;
00175    }
00176
00177    char j;
00178    for(j=0;j<CODESIZE;j++){
00179          code[j] = bitarray[j];
00180    }
00181
00182    char i;
00183    for(i=0; i<CODESIZE; i++)
00184    {
00185       PORTJ = bitarray[i];
00186       for(j=0; j<5; j++)
00187          _delay_ms(10);
00188    }
00189 }
00190
00194 unsigned char IR_read (){
00195    return patient_id(code);
00196 }
00197
00201 void learn(){
00202    char i,j;
00203    for(i=0;i<6;i++){
00204       //asking for value i.
00205       PORTJ = i+1;
00206       IR_Get_Input_Vector();
00207       for(j=0;j<5;j++){
00208          Pat[i][j] = code[j];
00209       }
00210       _delay_ms(1000);
00211    }
00212    PORTJ |= 0x40;
00213 }
00214
00218 void IR_init_devices(){
00219    cli();
00220    init_ports();
00221    USART_Init();
00222    sei();
00223 }
00224
00230 int main(void)
00231 {
00232    IR_init_devices();
00233    learn();
00234    while(1){
00235       IR_Get_Input_Vector();
00236       _delay_ms(500);
00237       patient = patient_id(code);
00238       PORTJ = patient;
00239    }
00240    return 0;
00241 }
```

7.11 src/BOT_CODE/lcd.h File Reference

```
#include <avr/io.h>

#include <avr/delay.h>

#include <util/delay.h>
```

Defines

- #define FCPU 11059200ul
- #define RS 0
- #define RW 1
- #define EN 2
- #define lcd_port PORTC
- #define sbit(reg, bit) reg |= (1<<bit)
- #define cbit(reg, bit) reg &= ~(1<<bit)

Functions

- void init_ports ()
- void lcd_reset_4bit ()
- void lcd_init ()
- void lcd_wr_command (unsigned char)
- void lcd_wr_char (char)
- void lcd_home ()
- void lcd_cursor (char, char)
- void lcd_print (char, char, unsigned int, int)
- void lcd_string (char *)
- void lcd_set_4bit ()

Variables

- unsigned int temp
- unsigned int unit
- unsigned int tens
- unsigned int hundred
- unsigned int thousand
- unsigned int million
- int i

7.11.1 Define Documentation

7.11.1.1 #define cbit(*reg, bit*) reg &= ~(1<<bit)

Definition at line 20 of file lcd.h.

7.11.1.2 #define EN 2

Definition at line 16 of file lcd.h.

7.11.1.3 #define FCPU 11059200ul

Definition at line 13 of file lcd.h.

7.11.1.4 #define lcd_port PORTC

Definition at line 17 of file lcd.h.

7.11.1.5 #define RS 0

Definition at line 14 of file lcd.h.

7.11.1.6 #define RW 1

Definition at line 15 of file lcd.h.

7.11.1.7 #define sbit(*reg, bit*) reg |= (1<<bit)

Definition at line 19 of file lcd.h.

7.11.2 Function Documentation

7.11.2.1 void init_ports ( )

Initialize all ports

Definition at line 95 of file IR.c.

7.11.2.2 void lcd_cursor (char *row*, char *column*)

Definition at line 167 of file lcd.h.

7.11.2.3 void lcd_home ( )

Definition at line 149 of file lcd.h.

7.11.2.4 void lcd_init ( )

Definition at line 85 of file lcd.h.

7.11.2.5 void lcd_print (char *row*, char *coloumn*, unsigned int *value*, int *digits*)

Definition at line 179 of file lcd.h.

7.11.2.6 void lcd_reset_4bit ( )

7.11.2.7 void lcd_set_4bit ( )

Definition at line 43 of file lcd.h.

7.11.2.8 void lcd_string (char * *str*)

Definition at line 156 of file lcd.h.

7.11.2.9 void lcd_wr_char (char *letter*)

Definition at line 124 of file lcd.h.

7.11.2.10 void lcd_wr_command (unsigned char *cmd*)

Definition at line 99 of file lcd.h.

7.11.3 Variable Documentation

7.11.3.1 unsigned int hundred

Definition at line 35 of file lcd.h.

7.11.3.2 int i

Definition at line 39 of file lcd.h.

7.11.3.3 unsigned int million

Definition at line 37 of file lcd.h.

7.11.3.4 unsigned int temp

Definition at line 32 of file lcd.h.

7.11.3.5 unsigned int tens

Definition at line 34 of file lcd.h.

7.11.3.6 unsigned int thousand

Definition at line 36 of file lcd.h.

7.11.3.7 unsigned int unit

Definition at line 33 of file lcd.h.

7.12 src/BOT_CODE/lcd.h

```
00001 /*
00002  * ============================================================================
      =======
00003  *       Filename:  lcd.h
00004  *       Compiler:  gcc-avr
00005  *
00006  *      Copyright:  NEX Robotics
00007  * ============================================================================
      =======
00008  */
00009 #include <avr/io.h>
00010 #include <avr/delay.h>
00011 #include <util/delay.h>
00012
00013 #define FCPU 11059200ul
00014 #define RS 0
00015 #define RW 1
00016 #define EN 2
00017 #define lcd_port PORTC
00018
00019 #define sbit(reg,bit)   reg |= (1<<bit)
00020 #define cbit(reg,bit)   reg &= ~(1<<bit)
00021
00022 void init_ports();
00023 void lcd_reset_4bit();
00024 void lcd_init();
00025 void lcd_wr_command(unsigned char);
00026 void lcd_wr_char(char);
00027 void lcd_home();
00028 void lcd_cursor(char, char);
```

```
00029 void lcd_print(char, char, unsigned int, int);
00030 void lcd_string(char*);
00031
00032 unsigned int temp;
00033 unsigned int unit;
00034 unsigned int tens;
00035 unsigned int hundred;
00036 unsigned int thousand;
00037 unsigned int million;
00038
00039 int i;
00040
00041
00042 /*****Function to Reset LCD*****/
00043 void lcd_set_4bit()
00044 {
00045    _delay_ms(1);
00046
00047    cbit(lcd_port,RS);            //RS=0 --- Command Input
00048    cbit(lcd_port,RW);            //RW=0 --- Writing to LCD
00049    lcd_port = 0x30;           //Sending 3
00050    sbit(lcd_port,EN);            //Set Enable Pin
00051    _delay_ms(5);              //Delay
00052    cbit(lcd_port,EN);            //Clear Enable Pin
00053
00054    _delay_ms(1);
00055
00056    cbit(lcd_port,RS);            //RS=0 --- Command Input
00057    cbit(lcd_port,RW);            //RW=0 --- Writing to LCD
00058    lcd_port = 0x30;           //Sending 3
00059    sbit(lcd_port,EN);            //Set Enable Pin
00060    _delay_ms(5);              //Delay
00061    cbit(lcd_port,EN);            //Clear Enable Pin
00062
00063    _delay_ms(1);
00064
00065    cbit(lcd_port,RS);            //RS=0 --- Command Input
00066    cbit(lcd_port,RW);            //RW=0 --- Writing to LCD
00067    lcd_port = 0x30;           //Sending 3
00068    sbit(lcd_port,EN);            //Set Enable Pin
00069    _delay_ms(5);              //Delay
00070    cbit(lcd_port,EN);            //Clear Enable Pin
00071
00072    _delay_ms(1);
00073
00074    cbit(lcd_port,RS);            //RS=0 --- Command Input
00075    cbit(lcd_port,RW);            //RW=0 --- Writing to LCD
00076    lcd_port = 0x20;           //Sending 2 to initialise LCD 4-bit mode
00077    sbit(lcd_port,EN);            //Set Enable Pin
00078    _delay_ms(5);              //Delay
00079    cbit(lcd_port,EN);            //Clear Enable Pin
00080
00081
00082 }
00083
00084 /*****Function to Initialize LCD*****/
00085 void lcd_init()
00086 {
00087    _delay_ms(1);
00088
00089    lcd_wr_command(0x28);         //LCD 4-bit mode and 2 lines.
00090    lcd_wr_command(0x01);
00091    lcd_wr_command(0x06);
00092    lcd_wr_command(0x0E);
00093    lcd_wr_command(0x80);
00094
00095 }
```

```
00096
00097
00098 /*****Function to Write Command on LCD*****/
00099 void lcd_wr_command(unsigned char cmd)
00100 {
00101    unsigned char temp;
00102    temp = cmd;
00103    temp = temp & 0xF0;
00104    lcd_port &= 0x0F;
00105    lcd_port |= temp;
00106    cbit(lcd_port,RS);
00107    cbit(lcd_port,RW);
00108    sbit(lcd_port,EN);
00109    _delay_ms(5);
00110    cbit(lcd_port,EN);
00111
00112    cmd = cmd & 0x0F;
00113    cmd = cmd<<4;
00114    lcd_port &= 0x0F;
00115    lcd_port |= cmd;
00116    cbit(lcd_port,RS);
00117    cbit(lcd_port,RW);
00118    sbit(lcd_port,EN);
00119    _delay_ms(5);
00120    cbit(lcd_port,EN);
00121 }
00122
00123 /*****Function to Write Data on LCD*****/
00124 void lcd_wr_char(char letter)
00125 {
00126    char temp;
00127    temp = letter;
00128    temp = (temp & 0xF0);
00129    lcd_port &= 0x0F;
00130    lcd_port |= temp;
00131    sbit(lcd_port,RS);
00132    cbit(lcd_port,RW);
00133    sbit(lcd_port,EN);
00134    _delay_ms(5);
00135    cbit(lcd_port,EN);
00136
00137    letter = letter & 0x0F;
00138    letter = letter<<4;
00139    lcd_port &= 0x0F;
00140    lcd_port |= letter;
00141    sbit(lcd_port,RS);
00142    cbit(lcd_port,RW);
00143    sbit(lcd_port,EN);
00144    _delay_ms(5);
00145    cbit(lcd_port,EN);
00146 }
00147
00148
00149 void lcd_home()
00150 {
00151    lcd_wr_command(0x80);
00152 }
00153
00154
00155 /*****Function to Print String on LCD*****/
00156 void lcd_string(char *str)
00157 {
00158    while(*str != '\0')
00159    {
00160       lcd_wr_char(*str);
00161       str++;
00162    }
```

```
00163 }
00164
00165 /*** Position the LCD cursor at "row", "column". ***/
00166
00167 void lcd_cursor (char row, char column)
00168 {
00169    switch (row) {
00170       case 1: lcd_wr_command (0x80 + column - 1); break;
00171       case 2: lcd_wr_command (0xc0 + column - 1); break;
00172       case 3: lcd_wr_command (0x94 + column - 1); break;
00173       case 4: lcd_wr_command (0xd4 + column - 1); break;
00174       default: break;
00175    }
00176 }
00177
00178 /***** Function To Print Any input value upto the desired digit on LCD *****/
00179 void lcd_print (char row, char coloumn, unsigned int value, int digits)
00180 {
00181    unsigned char flag=0;
00182    if(row==0||coloumn==0)
00183    {
00184       lcd_home();
00185    }
00186    else
00187    {
00188       lcd_cursor(row,coloumn);
00189    }
00190    if(digits==5 || flag==1)
00191    {
00192       million=value/10000+48;
00193       lcd_wr_char(million);
00194       flag=1;
00195    }
00196    if(digits==4 || flag==1)
00197    {
00198       temp = value/1000;
00199       thousand = temp%10 + 48;
00200       lcd_wr_char(thousand);
00201       flag=1;
00202    }
00203    if(digits==3 || flag==1)
00204    {
00205       temp = value/100;
00206       hundred = temp%10 + 48;
00207       lcd_wr_char(hundred);
00208       flag=1;
00209    }
00210    if(digits==2 || flag==1)
00211    {
00212       temp = value/10;
00213       tens = temp%10 + 48;
00214       lcd_wr_char(tens);
00215       flag=1;
00216    }
00217    if(digits==1 || flag==1)
00218    {
00219       unit = value%10 + 48;
00220       lcd_wr_char(unit);
00221    }
00222    if(digits>5)
00223    {
00224       lcd_wr_char('E');
00225    }
00226
00227 }
00228
```

7.13 src/SERVER_CODE/pss/configuration/Configure.java File Reference

Classes

- class pss::configuration::Configure

Namespaces

- namespace pss::configuration

7.14 src/SERVER_CODE/pss/configuration/Configure.java

```
00001 /*
00002  * ================================================================================
       =======
00003  *
00004  *       Filename:  Configure.java
00005  *
00006  *           Date:  31st March, 2010
00007  *
00008  *        Version:  2.1
00009  *       Revision:  2.1
00010  *       Compiler:  javac
00011  *
00012  *        Authors:  Pritish Kamath, pritish.kamath@iitb.ac.in
00013  *                  Rohit Saraf   , rohitsaraf@iitb.ac.in
00014  *                  Ashish Mathew , ashishmathew@iitb.ac.in
00015  *                  Vivek Madan   , vivekmadan@iitb.ac.in
00016  *
00017  *        Company:  IIT Bombay
00018  *      Copyright:  ERTS Lab, IIT Bombay
00019  *
00020  * ================================================================================
       =======
00021  */
00022
00023 package pss.configuration;
00024
00025 import java.io.FileInputStream;
00026 import java.io.IOException;
00027 import java.util.Properties;
00028
00036 public class Configure {
00041     public static String ZIGBEE_PORT;
00045     public static String DB_UN;
00049     public static String DB_PASS;
00053     public static String DB_DRIVER;
00057     public static String DB_URL;
00058
00062     public static String SMS_UN;
00066     public static String SMS_PASS;
00070     public static String SMS_MSG;
00074     public static String SMS_NUM;
00078     public static Integer NUM_BOTS;
00083     public static Boolean CLEAR_DB;
00084
00085
00089     public static String TEST_PORT1;
00093     public static String TEST_PORT2;
00094     static Configure instance = null;
00095
00101     public static synchronized Configure createInstance() throws IOException{
00102         if(instance == null){
```

```
00103             instance = new Configure();
00104             setValues(loadProperties());
00105         }
00106         return  instance;
00107     }
00108 
00114     public static Configure getInstance() throws IOException{
00115         if(instance==null){
00116             createInstance();
00117         }
00118         return instance;
00119     }
00120 
00126     public static void setValues(Properties p){
00127         ZIGBEE_PORT = p.getProperty("zigbee_port");
00128         DB_DRIVER = p.getProperty("db_driver");
00129         DB_PASS = p.getProperty("db_password");
00130         DB_UN = p.getProperty("db_username");
00131         DB_URL = p.getProperty("db_url");
00132         NUM_BOTS = Integer.parseInt(p.getProperty("num_bots"));
00133         SMS_UN = p.getProperty("sms_un");
00134         SMS_PASS = p.getProperty("sms_pass");
00135         SMS_MSG = p.getProperty("sms_msg");
00136         CLEAR_DB = Boolean.parseBoolean(p.getProperty("clear_db"));
00137         SMS_NUM=p.getProperty("sms_num");
00138         TEST_PORT1 = p.getProperty("test_port1");
00139         TEST_PORT2 = p.getProperty("test_port2");
00140     }
00141 
00147     public static Properties loadProperties() throws IOException{
00148         Properties prop = new Properties();
00149         prop.load(new FileInputStream("config/config.properties"));
00150         return prop;
00151     }
00152 }
```

7.15 src/SERVER_CODE/pss/serialcomm/CommunicationAPI.java File Reference

Classes

- class pss::serialcomm::CommunicationAPI
- class pss::serialcomm::CommunicationAPI::Sender
- class pss::serialcomm::CommunicationAPI::Receiver

Namespaces

- namespace pss::serialcomm

7.16 src/SERVER_CODE/pss/serialcomm/CommunicationAPI.java

```
00001 /*
00002  * ================================================================================
      =======
00003  *
00004  *       Filename:  CommunicationAPI.java
00005  *
00006  *           Date:  31st March, 2010
00007  *
00008  *        Version:  2.1
```

```
00009  *       Revision:  2.1
00010  *       Compiler:  javac
00011  *
00012  *        Authors:  Pritish Kamath, pritish.kamath@iitb.ac.in
00013  *                  Rohit Saraf   , rohitsaraf@iitb.ac.in
00014  *                  Ashish Mathew , ashishmathew@iitb.ac.in
00015  *                  Vivek Madan   , vivekmadan@iitb.ac.in
00016  *
00017  *        Company:  IIT Bombay
00018  *      Copyright:  ERTS Lab, IIT Bombay
00019  *
00020  * ========================================================================
       =======
00021  */
00022
00023 package pss.serialcomm;
00024
00025 import gnu.io.CommPortIdentifier;
00026 import gnu.io.PortInUseException;
00027 import gnu.io.SerialPort;
00028 import gnu.io.SerialPortEvent;
00029 import gnu.io.SerialPortEventListener;
00030 import gnu.io.UnsupportedCommOperationException;
00031 import java.io.IOException;
00032 import java.io.InputStream;
00033 import java.io.OutputStream;
00034 import java.util.Enumeration;
00035 import java.util.TooManyListenersException;
00036 import java.util.concurrent.ConcurrentLinkedQueue;
00037
00046 public class CommunicationAPI {
00047
00048     static CommPortIdentifier portId;
00049     static Enumeration portList;
00050     InputStream inputStream;
00051     SerialPort serialPort;
00052     String defaultPort;
00053     Thread readThread;
00054     static OutputStream outputStream;
00055     static boolean outputBufferEmptyFlag = false;
00056     static ConcurrentLinkedQueue<Character> in_buffer = new ConcurrentLinkedQueue
       <Character>();
00057     public static final Boolean DEBUG = false;
00058
00059     public CommunicationAPI(String port) {
00060         this.defaultPort = port;
00061
00062     }
00063
00067     public void close() {
00068         serialPort.close();
00069     }
00070
00071       /*
00072          Open the serial port for 2-way communication
00073       */
00074     public void open() {
00075         boolean portFound = false;
00076         portList = CommPortIdentifier.getPortIdentifiers();
00077
00078         while (portList.hasMoreElements()) {
00079             portId = (CommPortIdentifier) portList.nextElement();
00080             if (portId.getPortType() == CommPortIdentifier.PORT_SERIAL) {
00081                 if (portId.getName().equals(defaultPort)) {
00082                     if (DEBUG) {
00083                         System.out.println("Found port: " + defaultPort);
00084                     }
```

```
00085                     portFound = true;
00086                     try {
00087                         serialPort = (SerialPort) portId.open("communication-api"
      , 2000000);
00088                     } catch (PortInUseException e) {
00089                         if (DEBUG) {
00090                             System.out.println("Please connect X-bee properly!");

00091                         }
00092                     }
00093                     try {
00094                         inputStream = serialPort.getInputStream();
00095                     } catch (IOException e) {
00096                     }
00097                     try {
00098                         outputStream = serialPort.getOutputStream();
00099                     } catch (IOException e) {
00100                     }
00101                     Receiver x = new Receiver();
00102                     try {
00103                         serialPort.addEventListener(x);
00104                     } catch (TooManyListenersException e) {
00105                     }
00106                     serialPort.notifyOnDataAvailable(true);
00107                     try {
00108                         serialPort.setSerialPortParams(9600,
00109                                 SerialPort.DATABITS_8,
00110                                 SerialPort.STOPBITS_1,
00111                                 SerialPort.PARITY_NONE);
00112                     } catch (UnsupportedCommOperationException e) {
00113                     }
00114                     readThread = new Thread(x);
00115                     readThread.start();
00116                 }
00117             }
00118         }
00119         if (!portFound) {
00120             if (DEBUG) {
00121                 System.out.println("port " + defaultPort + " not found.");
00122             }
00123         }
00124     }
00125
00130     class Sender implements Runnable {
00131
00132         String messageString;
00133
00134         public Sender(String messString) {
00135             this.messageString = messString;
00136         }
00137
00138         public void run() {
00139             try{
00140                try {
00141                     System.out.println("Sending \"" + (int) messageString.charAt(0
      ) + "\" to " + serialPort.getName());
00142                     outputStream.write((String.valueOf(messageString)).charAt(0));

00143                     outputStream.flush();
00144                 } catch (IOException e) {
00145                 }
00146                 try {
00147                     Thread.sleep(2000);  // Be sure data is xferred before closing

00148                 } catch (Exception e) {
00149                 }
00150             }catch(Exception e){
```

```
00151                 System.out.println("Zigbee not found\n Please check the configurat
      ion file");
00152                 System.exit(1);
00153             }
00154         }
00155     }
00156 
00157     public void send(String messageString) {
00158         Thread s = new Thread(new Sender(messageString));
00159         s.start();
00160     }
00161 
00165     public void receive() {
00166         Receiver x = new Receiver();
00167         readThread = new Thread(x);
00168         readThread.start();
00169     }
00170 
00171     public void getNextCharFromBufferIfPresent() {
00172         in_buffer.poll();
00173     }
00174 
00175     public Character next_char_in_buffer() {
00176         Character x = null;
00177         long time1 = System.currentTimeMillis();
00178         long time2 = 0;
00179         while (true) {
00180             if (x != null) {
00181                 System.out.println("received message " + (int) x);
00182                 return x;
00183             }
00184             x = in_buffer.poll();
00185             time2 = System.currentTimeMillis();
00186             if(time2 - time1 > 2000){
00187                System.out.println("Timeout!");
00188                x = 0xff;
00189                return x;
00190             }
00191         }
00192     }
00193 
00197     class Receiver implements Runnable, SerialPortEventListener {
00198 
00199         public Receiver() {
00200         }
00201 
00202         public void run() {
00203             try {
00204                 while (true) {
00205                     Thread.sleep(20000);
00206                 }
00207             } catch (InterruptedException e) {
00208             }
00209         }
00210 
00211         public void serialEvent(SerialPortEvent event) {
00212             switch (event.getEventType()) {
00214                 case SerialPortEvent.OUTPUT_BUFFER_EMPTY:
00215                     break;
00216 
00217                 case SerialPortEvent.DATA_AVAILABLE:
00218                     try {
00219                         while (inputStream.available() > 0) {
00220                             int char_bytes = inputStream.read();
00221                             in_buffer.add((char) char_bytes);
00222                             if (DEBUG) {
00223                                 System.out.println("RECIEVED : " + (int) char_byt
```

```
      es);
00224                                 }
00225                             }
00226                         } catch (IOException e) {
00227                         }
00228                         break;
00229                 }
00230             }
00231         }
00232
00233         public static void main(String[] args) throws InterruptedException {
00234             CommunicationAPI capi = new CommunicationAPI("/dev/ttyUSB1");
00235             capi.open();
00236             capi.send(Character.toString(((char) 160)));
00237             capi.next_char_in_buffer();
00238
00239             Thread.sleep(1);
00240         }
00241
00242         ;
00243 }
```

7.17 src/SERVER_CODE/pss/server/Bot.java File Reference

Classes

- class pss::server::Bot

Namespaces

- namespace pss::server

7.18 src/SERVER_CODE/pss/server/Bot.java

```
00001 /*
00002  * ================================================================================
      =======
00003  *
00004  *       Filename:  Bot.java
00005  *
00006  *           Date:  31st March, 2010
00007  *
00008  *        Version:  2.1
00009  *       Revision:  2.1
00010  *       Compiler:  javac
00011  *
00012  *        Authors:  Pritish Kamath, pritish.kamath@iitb.ac.in
00013  *                  Rohit Saraf   , rohitsaraf@iitb.ac.in
00014  *                  Ashish Mathew , ashishmathew@iitb.ac.in
00015  *                  Vivek Madan   , vivekmadan@iitb.ac.in
00016  *
00017  *        Company:  IIT Bombay
00018  *      Copyright:  ERTS Lab, IIT Bombay
00019  *
00020  * ================================================================================
      =======
00021  */
00022
00023 package pss.server;
00024
00025 import pss.server.database.DBHandler;
```

```
00026 import pss.server.test.BotMotionTester1;
00027 import pss.serialcomm.CommunicationAPI;
00028
00029 public class Bot {
00030
00034     public static final Boolean DEBUG = false;
00038     public static final int running = 0;
00042     public static final int stationary = 1;
00046     public static final int obstruction = 3;
00047
00051     boolean isInUse = false;
00052
00056     int id;
00063     int status;
00068     Position currpos;
00069     DBHandler dbh;
00070     CommunicationAPI capi;
00071     int pid;
00072     Graph g = new Graph();
00076     public static int bot1 = 0;
00080     public static int bot2 = 1;
00081
00087     public Boolean isOriginalOrientation() {
00088         if (id == bot1) {
00089             if (currpos.orientation == Graph.SERVER_ID1_orientation) {
00090                 System.out.println("Bot " + id + " is in correct orientation");
00091                 return true;
00092             }
00093         } else {
00094             if (currpos.orientation == Graph.SERVER_ID2_orientation) {
00095                 System.out.println("Bot " + id + " is in correct orientation");
00096                 return true;
00097             }
00098         }
00099         System.out.println("Bot " + id + " is not in correct orientation");
00100
00101         return false;
00102     }
00103
00108     public Boolean isBotIdle() {
00109         if (this.isInUse) {
00110             System.out.println("It is in use");
00111         }
00112         if (!this.isInUse) {
00113             System.out.println("It is not in use");
00114         }
00115         if ((isAtHome()) && (isOriginalOrientation()) && (!(isInUse))) {
00116             return true;
00117         }
00118         System.out.println("Returning false");
00119         return false;
00120     }
00121
00126     public Boolean isAtHome() {
00127         if (id == bot1) {
00128             if (currpos.present == Graph.SERVER_ID1_position) {
00129                 System.out.println("Bot " + id + " is at home");
00130                 return true;
00131             }
00132         } else {
00133             if (currpos.present == Graph.SERVER_ID2_position) {
00134                 System.out.println("bot " + id + " is at home");
00135                 return true;
00136             }
00137         }
00138         System.out.println(" bot " + id + " is not at home");
00139         return false;
```

```
00140     }
00141
00149     public Boolean isSafePos(Position pos) {
00150         int mypos = currpos.present;
00151         int otherpos = pos.present;
00152         if (mypos == otherpos) {
00153             return false;
00154         } else {
00155             if ((mypos == 0 && otherpos == 3) || (mypos == 3 && otherpos == 0)) {

00156                 return false;
00157             } else {
00158                 return true;
00159             }
00160         }
00161     }
00162
00171     public Bot(DBHandler dbis, int id, CommunicationAPI capi) {
00172         this.dbh = dbis;
00173         this.capi = capi;
00174         this.id = id;
00175         if (id == bot1) {
00176             pid = Graph.SERVER_ID1;
00177             currpos = new Position(Graph.SERVER_ID1_position, Graph.SERVER_ID1_or
      ientation);
00178         }
00179         if (id == bot2) {
00180             pid = Graph.SERVER_ID2;
00181             currpos = new Position(Graph.SERVER_ID2_position, Graph.SERVER_ID2_or
      ientation);
00182         }
00183         this.isInUse = false;
00184         this.status = stationary;
00185     }
00186
00193     public static int getIdMess(char mess) {
00194         int int_mess = (int) mess;
00195         return int_mess / 32;
00196     }
00197
00203     public void setPosition(Position x) {
00204         this.currpos = x;
00205     }
00206
00210     public void gotoNextCross() {
00211         //change bot pos.
00212         if (DEBUG) {
00213             printBotPos();
00214
00215             System.out.println("going straight");
00216         }
00217
00218         this.currpos = g.Straight(currpos);
00219         //and send command to BOT.
00220         String t = Character.toString(Utils.straight_message(id));
00221         capi.send(t);
00222         BotMotionTester1.printMess(Utils.straight_message(id));
00223     }
00224
00229     public void printBotPos() {
00230         System.out.println("Bot id: " + id + "the position is " + currpos.
      present + "orientation is " + currpos.orientation);
00231     }
00232
00236     public void turnRight() {
00237         //send command to bot.
00238         if (DEBUG) {
```

```
00239             printBotPos();
00240             System.out.println("turning right");
00241         }
00242         this.currpos = g.Right_turn(this.currpos);
00243         String t = Character.toString(Utils.right_message(id));
00244         capi.send(t);
00245         BotMotionTester1.printMess(Utils.right_message(id));
00246     }
00247
00251     public void turnBack() {
00252         if (DEBUG) {
00253             printBotPos();
00254             System.out.println("turn back message is " + (int) Utils.back_message
      (id));
00255         }
00256         String t = Character.toString(Utils.back_message(id));
00257         this.currpos = g.Back(this.currpos);
00258         capi.send(t);
00259         BotMotionTester1.printMess(Utils.back_message(id));
00260     }
00261
00265     public void turnLeft() {
00266         printBotPos();
00267         if (DEBUG) {
00268             System.out.println("turning left");
00269         }
00270         this.currpos = g.Left_turn(this.currpos);
00271         String t = Character.toString(Utils.left_message(id));
00272         capi.send(t);
00273         BotMotionTester1.printMess(Utils.left_message(id));
00274
00275     }
00276 }
```

7.19 src/SERVER_CODE/pss/server/database/DBHandler.java File Reference

Classes

- class pss::server::database::DBHandler

Namespaces

- namespace pss::server::database

7.20 src/SERVER_CODE/pss/server/database/DBHandler.java

```
00001 /*
00002  * ================================================================================
      =======
00003  *
00004  *       Filename:  DBHandler.java
00005  *
00006  *           Date:  31st March, 2010
00007  *
00008  *        Version:  2.1
00009  *       Revision:  2.1
00010  *       Compiler:  javac
00011  *
00012  *        Authors:  Pritish Kamath, pritish.kamath@iitb.ac.in
00013  *                  Rohit Saraf   , rohitsaraf@iitb.ac.in
00014  *                  Ashish Mathew , ashishmathew@iitb.ac.in
```

```
00015  *                  Vivek Madan   , vivekmadan@iitb.ac.in
00016  *
00017  *        Company:  IIT Bombay
00018  *      Copyright:  ERTS Lab, IIT Bombay
00019  *
00020  * ========================================================================
       =======
00021  */
00022
00023 package pss.server.database;
00024
00025 import java.io.IOException;
00026 import java.io.PrintStream;
00027 import java.sql.*;
00028 import java.util.ArrayList;
00029 import java.util.List;
00030 import java.util.logging.Level;
00031 import java.util.logging.Logger;
00032 import pss.configuration.Configure;
00033
00037 public class DBHandler {
00038
00039     /*
00040      * System dependant parameters for setting up database connection
00041      */
00042     private static String driver_path = null;
00043     private static Connection conn = null;
00044     private static Statement stmt = null;
00045     private static String url = null;
00046     private static String username = null;
00047     private static String password = null;
00048
00052     private List<String> getCreateCommands() {
00053         List<String> a = new ArrayList<String>();
00054         a.add("create database erts");
00055         a.add("use erts");
00056         a.add("create table patient(patient_ID INT,Pos INT,primary key(patient_ID
       ))");
00057         a.add("create table request(request_ID INT,patient_ID INT,item char,statu
       s char(20),primary key (request_ID),foreign key (patient_ID) references patient)"
       );
00058         a.add("create table assignment(request_ID INT,bot_ID INT,foreign key (req
       uest_ID) references request)");
00059
00060         return a;
00061     }
00062
00063     private List<String> getDeleteCommands() {
00064         List<String> a = new ArrayList<String>();
00065         a.add("use erts");
00066         a.add("drop table assignment");
00067         a.add("drop table request");
00068         a.add("drop table patient");
00069         a.add("drop database erts");
00070         return a;
00071     }
00072
00076     public void closeConnection() {
00077         try {
00078             conn.close();
00079         } catch (SQLException ex) {
00080             System.err.println(ex.getMessage());
00081         }
00082     }
00083
00091     public void createConnection() {
00092         try {
```

```
00093             Configure.getInstance();
00094         } catch (IOException ex) {
00095             Logger.getLogger(DBHandler.class.getName()).log(Level.SEVERE, null, e
      x);
00096         }
00097         driver_path = Configure.DB_DRIVER;
00098         url = Configure.DB_URL;
00099         username = Configure.DB_UN;
00100         password = Configure.DB_PASS;
00101         try {
00102             Class.forName(driver_path).newInstance();
00103             conn = DriverManager.getConnection(url, username, password);
00104             stmt = conn.createStatement();
00105             if (conn == null) {
00106                 System.out.println("WARNING : No connection set up to create data
      base");
00107             }
00108 
00109         } catch (ClassNotFoundException ex) {
00110             System.err.println(ex.getMessage());
00111         } catch (IllegalAccessException ex) {
00112             System.err.println(ex.getMessage());
00113         } catch (InstantiationException ex) {
00114             System.err.println(ex.getMessage());
00115         } catch (SQLException ex) {
00116             System.err.println(ex.getMessage());
00117         }
00118     }
00119 
00123     public void createDatabase() {
00124         List<String> cmds = getCreateCommands();
00125         for (String cmd : cmds) {
00126             this.executeUpdate(cmd);
00127         }
00128     }
00129 
00133     public void deleteDatabase() {
00134         List<String> cmds = getDeleteCommands();
00135         for (String cmd : cmds) {
00136             try{
00137                this.executeUpdate(cmd);
00138             }catch(Exception e){}
00139         }
00140     }
00141 
00148     public ResultSet executeStatement(String cmd) {
00149         ResultSet rs = null;
00150         try {
00151             rs = stmt.executeQuery(cmd);
00152 
00153         } catch (SQLException ex) {
00154             System.err.println(ex.getMessage());
00155         }
00156         if (rs == null) {
00157             System.out.println("Warning execution unsucessful");
00158         }
00159         return rs;
00160     }
00161 
00166     public void executeUpdate(String cmd) {
00167         try {
00168             stmt.executeUpdate(cmd);
00169         } catch (SQLException ex) {
00170             System.err.println(ex.getMessage());
00171         }
00172     }
00173 
```

```
00180     private void printResultSet(ResultSet rs, PrintStream ps) {
00181         try {
00182             ResultSetMetaData rsmd = rs.getMetaData();
00183             String head = "";
00184             int colCount = rsmd.getColumnCount();
00185             for (int i = 1; i <= colCount; i++) {
00186                 head += rsmd.getColumnName(i) + "\t";
00187             }
00188             ps.println(head);
00189             while (rs.next()) {
00190                 String temp = "";
00191                 for (int i = 1; i <= colCount; i++) {
00192                     temp = temp + "\t" + rs.getString(i);
00193                 }
00194                 ps.println(temp);
00195             }
00196         } catch (SQLException e) {
00197             ps.println(e);
00198         }
00199     }
00203     public static final Boolean DEBUG = false;
00204
00211     public boolean addPatient(int patient_ID, int Pos) {
00212         if (DEBUG) {
00213             System.out.println("Adding patient");
00214         }
00215         try {
00216             PreparedStatement ps = conn.prepareStatement("insert into patient(pat
      ient_ID,Pos) values(?,?)");
00217             ps.setInt(1, patient_ID);
00218             ps.setInt(2, Pos);
00219             ps.executeUpdate();
00220             if (DEBUG) {
00221                 System.out.println("Added patient");
00222             }
00223
00224             return true;
00225         } catch (Exception e) {
00226             System.out.println(e);
00227             System.out.println("Could not add patient");
00228             return false;
00229         }
00230     }
00231
00237     public boolean deletePatient(int id) {
00238         try {
00239             PreparedStatement ps = conn.prepareStatement("delete from patient whe
      re patient_ID=?");
00240             ps.setInt(1, id);
00241             ps.executeUpdate();
00242             return true;
00243         } catch (Exception e) {
00244             System.out.println(e);
00245             return false;
00246         }
00247     }
00248
00257     public boolean addRequest(int request_id, int patient_id, String item, String
       status) {
00258         try {
00259             PreparedStatement ps = conn.prepareStatement("insert into request(req
      uest_ID,patient_ID,item,status) values(?,?,?,?)");
00260             ps.setInt(1, request_id);
00261             ps.setInt(2, patient_id);
00262             ps.setString(3, item);
00263             ps.setString(4, status);
00264             ps.executeUpdate();
```

```
00265             return true;
00266         } catch (Exception e) {
00267             System.out.println(e);
00268             return false;
00269         }
00270     }
00271
00277     public boolean deleteRequest(int request_id) {
00278         try {
00279             PreparedStatement ps = conn.prepareStatement("delete from request whe
      re request_ID=?");
00280             ps.setInt(1, request_id);
00281             ps.executeUpdate();
00282             return true;
00283         } catch (Exception e) {
00284             System.out.println(e);
00285             return false;
00286         }
00287     }
00288
00295     public boolean updateRequestStatus(int request_id, String status) {
00296         try {
00297             PreparedStatement ps = conn.prepareStatement("update request set stat
      us =? where request_ID=?");
00298             ps.setString(1, status);
00299             ps.setInt(2, request_id);
00300             ps.executeUpdate();
00301             return true;
00302         } catch (Exception e) {
00303             System.out.println(e);
00304             return false;
00305         }
00306     }
00307
00314     public boolean addAssignment(int request_id, int bot_ID) {
00315         try {
00316             PreparedStatement ps = conn.prepareStatement("insert into assignment(
      request_ID,bot_ID) values(?,?)");
00317             ps.setInt(1, request_id);
00318             ps.setInt(2, bot_ID);
00319             ps.executeUpdate();
00320             return true;
00321         } catch (Exception e) {
00322             System.out.println(e);
00323             return false;
00324         }
00325     }
00326
00333     public boolean deleteAssignment(int id, boolean is_request_id) {
00334         try {
00335             PreparedStatement ps;
00336             if (is_request_id) {
00337                 ps = conn.prepareStatement("delete from assignment where request_
      ID=?");
00338             } else {
00339                 ps = conn.prepareStatement("delete from assignment where bot_ID=?
      ");
00340             }
00341             ps.setInt(1, id);
00342             ps.executeUpdate();
00343             return true;
00344         } catch (Exception e) {
00345             System.out.println(e);
00346             return false;
00347         }
00348     }
00349
```

```
00354     public int getNextRequestFIFO() {
00355         try {
00356             PreparedStatement ps;
00357             ps = conn.prepareStatement("select request_ID from request where stat
      us=? order by request_ID ASC");
00358             ps.setString(1, "pending");
00359             ResultSet rs = ps.executeQuery();
00360             if (rs.next()) {
00361                 return rs.getInt(1);
00362             } else {
00363                 return -1;
00364             }
00365         } catch (Exception e) {
00366             System.out.println(e);
00367             return -1;
00368         }
00369     }
00370 
00376     public int getAssignedRequest(int bot_ID) {
00377         try {
00378             PreparedStatement ps;
00379             ps = conn.prepareStatement("select request_ID from assignment where b
      ot_ID=?");
00380             ps.setInt(1, bot_ID);
00381             ResultSet rs = ps.executeQuery();
00382             if (rs.next()) {
00383                 return rs.getInt(1);
00384             } else {
00385                 return -1;
00386             }
00387         } catch (Exception e) {
00388             System.out.println(e);
00389             return -1;
00390         }
00391     }
00392 
00398     public int getPatientOfRequest(int req_id) {
00399         try {
00400             PreparedStatement ps;
00401             ps = conn.prepareStatement("select patient_ID from request where requ
      est_ID=?");
00402             ps.setInt(1, req_id);
00403             ResultSet rs = ps.executeQuery();
00404             if (rs.next()) {
00405                 return rs.getInt(1);
00406             } else {
00407                 return -1;
00408             }
00409         } catch (Exception e) {
00410             System.out.println(e);
00411             return -1;
00412         }
00413     }
00414 
00420     public static void main(String[] args) throws SQLException {
00421 
00422 
00423         DBHandler dbh = new DBHandler();
00424         dbh.createConnection();
00425 
00426         System.out.println("Initializing database...");
00427         if (true) {
00428             try {
00429                 dbh.deleteDatabase();
00430             } catch (Exception e) {
00431                 System.out.println("No database to delete");
00432             }
```

```
00433         }
00434         try {
00435             dbh.createDatabase();
00436         } catch (Exception e) {
00437             dbh.executeUpdate("use erts");
00438             System.out.println("Reading Existing Database...");
00439         }
00440 
00441 
00442         String a[] = new String[1];
00443         a[0] = "request";
00444 
00445         dbh.addRequest(1, 5, "1", "pending");
00446         dbh.addRequest(2, 5, "1", "pending");
00447         dbh.addRequest(3, 5, "1", "pending");
00448         dbh.addAssignment(1, 3);
00449         dbh.addPatient(2, 5);
00450         ResultSet rs = dbh.executeStatement("select * from request");
00451         dbh.printResultSet(rs, System.out);
00452 
00453         System.out.println(dbh.getAssignedRequest(3));
00454         System.out.println(dbh.getNextRequestFIFO());
00455         System.out.println(dbh.getPatientOfRequest(3));
00456         dbh.closeConnection();
00457     }
00458 }
```

7.21 src/SERVER_CODE/pss/server/Graph.java File Reference

Classes

- class pss::server::Graph

Namespaces

- namespace pss::server

7.22 src/SERVER_CODE/pss/server/Graph.java

```
00001 /*
00002  * ================================================================================
      =======
00003  *
00004  *       Filename:  Graph.java
00005  *
00006  *           Date:  31st March, 2010
00007  *
00008  *        Version:  2.1
00009  *       Revision:  2.1
00010  *       Compiler:  javac
00011  *
00012  *        Authors:  Pritish Kamath, pritish.kamath@iitb.ac.in
00013  *                  Rohit Saraf   , rohitsaraf@iitb.ac.in
00014  *                  Ashish Mathew , ashishmathew@iitb.ac.in
00015  *                  Vivek Madan   , vivekmadan@iitb.ac.in
00016  *
00017  *        Company:  IIT Bombay
00018  *      Copyright:  ERTS Lab, IIT Bombay
00019  *
00020  * ================================================================================
      =======
```

```
00021  */
00022
00023 package pss.server;
00024
00025 import java.util.HashMap;
00026 import java.util.Map;
00027
00034 public class Graph {
00035
00039     static Map<Integer, Integer> patientPos = new HashMap<Integer, Integer>();
00052     public int[][][] grph = new int[16][4][4];
00053     public static final int RIGHT = 0, LEFT = 1, STRAIGHT = 2, BACKWARD = 3,
      FINISH = 5, NOPATH = -2, BOT_POLLING = 5, PATIENT_POLLING = 6;      // R=right, L=
      left,F=forward,B=backward. it indicates the direction in which you move.
00054     public static final int NORTH = 0, SOUTH = 1, EAST = 2, WEST = 3;
00055     /*
00056      * Place holder for a distance that cannot be acheived on any shortest path i
      n the graph and hence can be safely
00057      * treated as infinity.
00058      */
00059     public static final int INFINTY = 25;
00067     public int[][][] distance_pos_ori_pos = new int[16][4][16];
00071     boolean DEBUG = false;
00075     int no_calls = 0;
00076
00082     public void init_graph() {
00083         for (int i = 0; i < 16; i++) {
00084             for (int j = 0; j < 4; j++) {
00085                 for (int k = 0; k < 16; k++) {
00086                     distance_pos_ori_pos[i][j][k] = INFINTY;
00087                 }
00088             }
00089         }
00090
00091         patientPos.put(-1, 0);
00092         patientPos.put(2, 5);
00093         patientPos.put(3, 9);
00094         patientPos.put(4, 13);
00095         patientPos.put(5, 10);
00096         patientPos.put(6, 6);
00097         patientPos.put(7, 1);
00098         patientPos.put(-2, 14);
00099         patientPos.put(-3, 15);
00100
00101         for (int i = 0; i < 16; i++) {
00102             for (int j = 0; j < 4; j++) {
00103                 for (int k = 0; k < 4; k++) {
00104                     grph[i][j][k] = -1;
00105                 }
00106             }
00107         }
00108         grph[0][WEST][RIGHT] = 15;
00109         grph[0][WEST][LEFT] = 14;
00110         grph[0][NORTH][STRAIGHT] = 15;
00111         grph[0][NORTH][RIGHT] = 3;
00112         grph[0][SOUTH][STRAIGHT] = 14;
00113         grph[0][SOUTH][LEFT] = 3;
00114         grph[0][EAST][STRAIGHT] = 3;
00115
00116         grph[15][NORTH][BACKWARD] = 0;
00117         grph[15][SOUTH][STRAIGHT] = 0;
00118         grph[14][SOUTH][BACKWARD] = 0;
00119         grph[14][NORTH][STRAIGHT] = 0;
00120
00121         grph[3][EAST][LEFT] = 4;
00122         grph[3][NORTH][LEFT] = 0;
00123         grph[3][WEST][STRAIGHT] = 0;
```

```
00124         grph[3][NORTH][STRAIGHT] = 4;
00125
00126         grph[4][EAST][STRAIGHT] = 8;
00127         grph[4][EAST][LEFT] = 5;
00128         grph[4][SOUTH][LEFT] = 8;
00129         grph[4][NORTH][STRAIGHT] = 5;
00130         grph[4][NORTH][RIGHT] = 8;
00131
00132         grph[8][EAST][STRAIGHT] = 12;
00133         grph[8][EAST][LEFT] = 9;
00134         grph[8][SOUTH][LEFT] = 12;
00135         grph[8][NORTH][STRAIGHT] = 9;
00136         grph[8][NORTH][RIGHT] = 12;
00137
00138         grph[12][EAST][LEFT] = 13;
00139         grph[12][EAST][RIGHT] = 11;
00140         grph[12][SOUTH][STRAIGHT] = 11;
00141         grph[12][NORTH][STRAIGHT] = 13;
00142         grph[12][NORTH][BACKWARD] = 11;
00143
00144         grph[11][SOUTH][STRAIGHT] = 10;
00145         grph[11][SOUTH][RIGHT] = 7;
00146         grph[11][NORTH][LEFT] = 7;
00147         grph[11][WEST][STRAIGHT] = 7;
00148
00149         grph[7][SOUTH][STRAIGHT] = 6;
00150         grph[7][SOUTH][RIGHT] = 2;
00151         grph[7][NORTH][LEFT] = 2;
00152         grph[7][WEST][STRAIGHT] = 2;
00153         grph[7][WEST][LEFT] = 6;
00154
00155         grph[2][SOUTH][STRAIGHT] = 1;
00156         grph[2][SOUTH][BACKWARD] = 3;
00157         grph[2][NORTH][STRAIGHT] = 3;
00158         grph[2][WEST][LEFT] = 1;
00159         grph[2][WEST][RIGHT] = 3;
00160
00161         grph[5][NORTH][BACKWARD] = 4;
00162         grph[5][SOUTH][STRAIGHT] = 4;
00163
00164         grph[9][NORTH][BACKWARD] = 8;
00165         grph[9][SOUTH][STRAIGHT] = 8;
00166
00167         grph[13][NORTH][BACKWARD] = 12;
00168         grph[13][SOUTH][STRAIGHT] = 12;
00169
00170         grph[10][NORTH][STRAIGHT] = 11;
00171         grph[10][SOUTH][BACKWARD] = 11;
00172
00173         grph[6][NORTH][STRAIGHT] = 7;
00174         grph[6][SOUTH][BACKWARD] = 7;
00175
00176         grph[1][NORTH][STRAIGHT] = 2;
00177         grph[1][SOUTH][BACKWARD] = 2;
00178     }
00179
00185     public Position Left_turn(Position p) {
00186         Position t = new Position();
00187         t.present = p.present;
00188         switch (p.orientation) {
00189             case WEST:
00190                 t.orientation = SOUTH;
00191                 break;
00192             case EAST:
00193                 t.orientation = NORTH;
00194                 break;
00195             case NORTH:
```

```
00196                 t.orientation = WEST;
00197                 break;
00198             case SOUTH:
00199                 t.orientation = EAST;
00200                 break;
00201         } //send command to bot.
00202         return t;
00203     }
00204
00210     public Position Back(Position p) {
00211         Position t = new Position();
00212         t.present = p.present;
00213         switch (p.orientation) {
00214             case WEST:
00215                 t.orientation = EAST;
00216                 break;
00217             case EAST:
00218                 t.orientation = WEST;
00219                 break;
00220             case NORTH:
00221                 t.orientation = SOUTH;
00222                 break;
00223             case SOUTH:
00224                 t.orientation = NORTH;
00225                 break;
00226         }
00227         return t;
00228     }
00229
00235     public Position Right_turn(Position p) {
00236         Position t = new Position();
00237         t.present = p.present;
00238         switch (p.orientation) {
00239             case WEST:
00240                 t.orientation = NORTH;
00241                 break;
00242             case EAST:
00243                 t.orientation = SOUTH;
00244                 break;
00245             case NORTH:
00246                 t.orientation = EAST;
00247                 break;
00248             case SOUTH:
00249                 t.orientation = WEST;
00250                 break;
00251         }
00252         return t;
00253     }
00254
00260     public Position Straight(Position p) {
00261         Position t = new Position();
00262         t.present = grph[p.present][p.orientation][STRAIGHT];
00263         t.orientation = p.orientation;
00264         return t;
00265     }
00266
00273     public Position pos_after_action(Position curr, int action) {
00274         if (action == LEFT) {
00275             if (DEBUG) {
00276                 System.out.println("Turn left");
00277             }
00278             return Left_turn(curr);
00279         } else if (action == RIGHT) {
00280             if (DEBUG) {
00281                 System.out.println(" Turn right");
00282             }
00283             return Right_turn(curr);
```

```
00284         }
00285         if (action == STRAIGHT) {
00286             if (DEBUG) {
00287                 System.out.println("Move Straight");
00288             }
00289             return Straight(curr);
00290         }
00291         if (action == BACKWARD) {
00292             if (DEBUG) {
00293                 System.out.println("Reverse");
00294             }
00295             return Back(curr);
00296         } else {
00297             System.exit(0);
00298             return null;
00299         }
00300     }
00301
00308     public void move_the_bot(Position curr, int patient_id) {
00309         boolean success = true;
00310
00311         while (curr.present != patientPos.get(patient_id)) {
00312             System.out.println("Position is " + curr.present + " orientation is
      " + curr.orientation);
00313             int action = search(patient_id, curr);
00314             curr = pos_after_action(curr, action);
00315             if (curr == null) {
00316                 success = false;
00317                 break;
00318             }
00319         }
00320         if (success == false) {
00321             System.out.println("path does not exist");
00322         }
00323     }
00324
00333     public int find_distance(Position curr, int final_pos, int counter) {
00334         int min_distance = INFINTY;
00335         no_calls++;
00336         if (DEBUG) {
00337             System.out.println("Number of calls is " + no_calls + "counter is " +
       counter);
00338             System.out.println("Position is : " + curr.present + " Orientation :
      " + curr.orientation + " Counter : " + counter);
00339         }
00340         if (distance_pos_ori_pos[curr.present][curr.orientation][final_pos] <
      INFINTY) {
00341             return distance_pos_ori_pos[curr.present][curr.orientation][final_pos
      ];
00342         }
00343         if (curr.present == final_pos) {
00344             return 0;
00345         }
00346         if (counter == 0) {
00347             return INFINTY + 1;
00348         }
00349         if (DEBUG) {
00350             System.out.println("Number of calls is " + no_calls + "counter is " +
       counter);
00351         }
00352         if (grph[curr.present][curr.orientation][LEFT] != -1) {
00353             if (DEBUG) {
00354                 System.out.println("Searching in left for position : " + curr.
      present + " orientation " + curr.orientation);
00355                 System.out.println(" " + Left_turn(curr).present + " orientation
      " + Left_turn(curr).orientation);
00356             }
```

```
00357             int d1 = find_distance(Left_turn(curr), final_pos, counter - 1);
00358             if (d1 < min_distance) {
00359                 min_distance = d1;
00360             }
00361         }
00362         if (grph[curr.present][curr.orientation][RIGHT] != -1) {
00363             if (DEBUG) {
00364                 System.out.println("Searching in right for position : " + curr.
      present + " orientation " + curr.orientation);
00365                 System.out.println(" " + Right_turn(curr).present + " orientation
       " + Right_turn(curr).orientation);
00366             }
00367             int d1 = find_distance(Right_turn(curr), final_pos, counter - 1);
00368             if (d1 < min_distance) {
00369                 min_distance = d1;
00370             }
00371         }
00372         if (grph[curr.present][curr.orientation][STRAIGHT] != -1) {
00373             if (DEBUG) {
00374                 System.out.println("Searching in forward for position : " + curr.
      present + " orientation " + curr.orientation);
00375             }
00376             int d1 = find_distance(Straight(curr), final_pos, counter - 1);
00377             if (d1 < min_distance) {
00378                 min_distance = d1;
00379             }
00380         }
00381         if (grph[curr.present][curr.orientation][BACKWARD] != -1) {
00382             if (DEBUG) {
00383                 System.out.println("Searching in backward for position : " + curr
      .present + " orientation " + curr.orientation);
00384             }
00385             int d1 = find_distance(Back(curr), final_pos, counter - 1);
00386             if (d1 < min_distance) {
00387                 min_distance = d1;
00388             }
00389         }
00390         if (min_distance == INFINTY) {
00391             if (DEBUG) {
00392                 System.out.println("MAX_DISTANCE No path: " + curr.present + " or
      ientation " + curr.orientation);
00393             }
00394             distance_pos_ori_pos[curr.present][curr.orientation][final_pos] =
      INFINTY;
00395             return INFINTY;
00396         } else {
00397             distance_pos_ori_pos[curr.present][curr.orientation][final_pos] = min
      _distance + 1;
00398             return min_distance + 1;
00399         }
00400
00401     }
00402
00406     public Graph() {
00407         this.init_graph();
00408     }
00409
00413     public static final int SERVER_ID1_position = 14;
00417     public static final int SERVER_ID2_position = 15;
00421     public static final int SERVER_ID1_orientation = Graph.SOUTH;
00425     public static final int SERVER_ID2_orientation = Graph.NORTH;
00426
00430     public static final int SERVER_ID1 = -2;
00434     public static final int SERVER_ID2 = -3;
00435
00442     public int search(int patient_id, Position curr) {
00443         int final_pos = patientPos.get(patient_id);
```

```
00445         int min_distance = INFINTY;
00446         int action = 13;
00447         int max_counter = INFINTY;
00448         if (final_pos == curr.present) {
00449             return FINISH;
00450         }
00451         if (grph[curr.present][curr.orientation][LEFT] != -1) {
00452             if (DEBUG) {
00453                 System.out.println("Searching in left for position : " + curr.
      present + " orientation " + curr.orientation);
00454                 System.out.println("Sending in left position : " + Left_turn(curr
      ).present + " orientation " + Left_turn(curr).orientation);
00455             }
00456             int d1 = find_distance(Left_turn(curr), final_pos, max_counter);
00457             if (d1 < min_distance) {
00458                 action = LEFT;
00459                 min_distance = d1;
00460             }
00461         }
00462         if (grph[curr.present][curr.orientation][RIGHT] != -1) {
00463             if (DEBUG) {
00464                 System.out.println("Searching in right for position : " + curr.
      present + " orientation " + curr.orientation);
00465             }
00466             int d1 = find_distance(Right_turn(curr), final_pos, max_counter);
00467             if (d1 < min_distance) {
00468                 action = RIGHT;
00469                 min_distance = d1;
00470             }
00471         }
00472         if (grph[curr.present][curr.orientation][STRAIGHT] != -1) {
00473             if (DEBUG) {
00474                 System.out.println("Searching in forward for position : " + curr.
      present + " orientation " + curr.orientation);
00475             }
00476             int d1 = find_distance(Straight(curr), final_pos, max_counter);
00477             if (d1 < min_distance) {
00478                 action = STRAIGHT;
00479                 min_distance = d1;
00480             }
00481         }
00482         if (grph[curr.present][curr.orientation][BACKWARD] != -1) {
00483             if (DEBUG) {
00484                 System.out.println("Searching in backward for position : " + curr
      .present + " orientation " + curr.orientation);
00485             }
00486             int d1 = find_distance(Back(curr), final_pos, max_counter);
00487             if (d1 < min_distance) {
00488                 action = BACKWARD;
00489                 min_distance = d1;
00490             }
00491         }
00492         if (min_distance == INFINTY) {
00493             if (DEBUG) {
00494                 System.out.println("NOPATH : " + curr.present + " orientation " +
       curr.orientation);
00495             }
00496             return NOPATH;
00497         } else {
00498             if (DEBUG) {
00499                 System.out.println("minimum distance is " + min_distance);
00500             }
00501             return action;
00502         }
00503     }
00504 }
```

7.23 src/SERVER_CODE/pss/server/Position.java File Reference

Classes

- class pss::server::Position

Namespaces

- namespace pss::server

7.24 src/SERVER_CODE/pss/server/Position.java

```
00001 /*
00002  * ================================================================================
       =======
00003  *
00004  *       Filename:  Position.java
00005  *
00006  *           Date:  31st March, 2010
00007  *
00008  *        Version:  2.1
00009  *       Revision:  2.1
00010  *       Compiler:  javac
00011  *
00012  *        Authors:  Pritish Kamath, pritish.kamath@iitb.ac.in
00013  *                  Rohit Saraf   , rohitsaraf@iitb.ac.in
00014  *                  Ashish Mathew , ashishmathew@iitb.ac.in
00015  *                  Vivek Madan   , vivekmadan@iitb.ac.in
00016  *
00017  *        Company:  IIT Bombay
00018  *      Copyright:  ERTS Lab, IIT Bombay
00019  *
00020  * ================================================================================
       =======
00021  */
00022
00023 package pss.server;
00024
00029 public class Position {
00030
00034     public int present;
00038     public int orientation;
00039
00046     public Position(int present, int orientation) {
00047         this.present = present;
00048         this.orientation = orientation;
00049     }
00050
00054     public Position(){
00055
00056     }
00057 }
```

7.25 src/SERVER_CODE/pss/server/RequestHandler.java File Reference

Classes

- class pss::server::RequestHandler

Namespaces

- namespace pss::server

7.26 src/SERVER_CODE/pss/server/RequestHandler.java

```
00001 /*
00002  * ================================================================================
      =======
00003  *
00004  *       Filename:  RequestHandler.java
00005  *
00006  *           Date:  31st March, 2010
00007  *
00008  *        Version:  2.1
00009  *       Revision:  2.1
00010  *       Compiler:  javac
00011  *
00012  *        Authors:  Pritish Kamath, pritish.kamath@iitb.ac.in
00013  *                  Rohit Saraf   , rohitsaraf@iitb.ac.in
00014  *                  Ashish Mathew , ashishmathew@iitb.ac.in
00015  *                  Vivek Madan   , vivekmadan@iitb.ac.in
00016  *
00017  *        Company:  IIT Bombay
00018  *      Copyright:  ERTS Lab, IIT Bombay
00019  *
00020  * ================================================================================
      =======
00021  */
00022
00080 package pss.server;
00081
00082 import java.io.IOException;
00083 import pss.server.database.DBHandler;
00084 import java.util.Iterator;
00085 import java.util.HashMap;
00086 import java.util.Map;
00087 import java.util.Map.Entry;
00088 import java.util.logging.Level;
00089 import java.util.logging.Logger;
00090 import pss.configuration.Configure;
00091 import pss.serialcomm.CommunicationAPI;
00092 import pss.server.scheduling.PollingThread;
00093
00101 public class RequestHandler {
00102
00103     public static final Boolean DEBUG = false;
00104     public static int no_bots = 0;
00108     DBHandler dbh;
00112     static Map<Integer, Bot> botList = new HashMap<Integer, Bot>();
00117     static Graph g = new Graph();
00122     public static CommunicationAPI capi;
00123
00130     public RequestHandler() {
00131         dbh = new DBHandler();
00132         dbh.createConnection();
00133         try {
00134             Configure.getInstance();
00135         } catch (IOException ex) {
00136             Logger.getLogger(RequestHandler.class.getName()).log(Level.SEVERE, nu
      ll, ex);
00137             System.out.println("Could not create connection");
00138         }
00139
00140         boolean clearDatabase = Configure.CLEAR_DB;
```

```
00141         System.out.println("Initializing database...");
00142         if (true) {
00143             try {
00144                 dbh.deleteDatabase();
00145             } catch (Exception e) {
00146                 System.out.println("No database to delete");
00147             }
00148         }
00149         try {
00150             dbh.createDatabase();
00151         } catch (Exception e) {
00152             dbh.executeUpdate("use erts");
00153             System.out.println("Reading Existing Database...");
00154         }
00155 
00156         if (DEBUG) {
00157             System.out.println("Opening the Communication API...");
00158         }
00159         capi = new CommunicationAPI(Configure.ZIGBEE_PORT);
00160         capi.open();
00161         if (DEBUG) {
00162             System.out.println("Creating bot objects...");
00163         }
00164         create_bots(this.dbh);
00165         if (DEBUG) {
00166             System.out.println("Creating patients objects...");
00167         }
00168         create_patients(this.dbh);
00169     }
00170 
00176     private void create_patients(DBHandler db) {
00177         if (DEBUG) {
00178             System.out.println("Adding patient info...");
00179         }
00180         this.dbh.addPatient(2, 5);
00181         this.dbh.addPatient(3, 9);
00182         this.dbh.addPatient(4, 13);
00183         this.dbh.addPatient(5, 10);
00184         this.dbh.addPatient(6, 6);
00185         this.dbh.addPatient(7, 1);
00186         if (DEBUG) {
00187             System.out.println("Added Patient");
00188         }
00189     }
00190 
00196     private void create_bots(DBHandler db) {
00197         int first_bot_id = Bot.bot1;
00198         Bot firstbot = new Bot(db, first_bot_id, capi);
00199         if (!botList.containsKey(first_bot_id)) {
00200             botList.put(first_bot_id, firstbot);
00201         }
00202         if (no_bots == 2) {
00203             int second_bot_id = Bot.bot2;
00204             Bot secondbot = new Bot(db, second_bot_id, capi);
00205             if (!botList.containsKey(second_bot_id)) {
00206                 botList.put(second_bot_id, secondbot);
00207             }
00208         }
00209     }
00210 
00218     public static void main(String args[]) {
00219         try {
00220             Configure.getInstance();
00221         } catch (IOException ex) {
00222             Logger.getLogger(RequestHandler.class.getName()).log(Level.SEVERE, nu
      ll, ex);
00223         }
```

```
00224         no_bots = Configure.NUM_BOTS;
00225         RequestHandler rh = new RequestHandler();
00226         PollingThread poll = new PollingThread(capi);
00227         Character patient_mess1 = 6 * 32;
00228         System.out.println("Executing patient request...");
00229         while (true) {
00230             Character comm = poll.poll_next(PollingThread.bot_or_patient);
00231             if (DEBUG) {
00232                 System.out.println("Got reply");
00233             }
00234             if (PollingThread.bot_or_patient == PollingThread.poll_bot) {
00235                 rh.execute_bot_Command(comm);
00236                 System.out.println("Got message " + Integer.toBinaryString((int)
      comm) + " from serving bot");
00237                 if (poll.poll_id == poll.number_bots - 1) {
00238                     PollingThread.bot_or_patient = PollingThread.poll_patient;
00239                 }
00240             } else if (PollingThread.bot_or_patient == PollingThread.poll_patient
      ) {
00241                 rh.execute_patient_request(comm);
00242                 System.out.println("Got message " + Integer.toBinaryString((int)
      comm) + " from patient bot");
00243                 PollingThread.bot_or_patient = PollingThread.poll_bot;
00244             }
00245             for (int i = 0; i < 2000; i++) {
00246                 try {
00247                     Thread.sleep(1);
00248                 } catch (InterruptedException ex) {
00249                 }
00250             }
00251         }
00252     }
00256     public static final int BLOCKED = 0;
00260     public static final int ACK = 1;
00264     public static final int IN_PROGRESS = 2;
00265 
00283     public void execute_bot_Command(char command) {
00284         int comm_int = (int) command;
00285         int bot_id = comm_int / 32;
00286         int request = comm_int % 32;
00287         switch (request) {
00288             case BLOCKED:
00289                 System.out.println("Patient is blocked ... SENDING SMS");
00290                 try {
00291                     Runtime.getRuntime().exec("python send-sms.py " + Configure.S
      MS_NUM + Configure.SMS_MSG);
00292                 } catch (IOException ex) {
00293                     ex.printStackTrace();
00294                 }
00295                 bot_blocked(bot_id);
00296                 break;
00297             case ACK:
00298                 System.out.println("Bot has completed its previous request");
00299                 Bot x = botList.get(bot_id);
00300                 Bot other_bot = botList.get((bot_id + 1) % no_bots);
00301                 Position pos = x.currpos;
00302                 if (x.isAtHome() && x.isOriginalOrientation()) {
00303                     System.out.println("Bot is idle");
00304                     assign_if_request_available(bot_id);
00305                 } else {
00306                     System.out.println("Patient being served is " + x.pid);
00307                     int action = g.search(x.pid, pos);
00308                     System.out.println("Action is " + action);
00309                     boolean finish = true;
00310                     while (finish) {
00311                         switch (action) {
00312                             case Graph.NOPATH:
```

```
00313                                   System.out.println("There is no path from this ve
      rtex");
00314                                   break;
00315                               case Graph.FINISH:
00316                                   System.out.println("Bot is in patient room");
00317                                   request_completed(bot_id);
00318                                   action = g.search(x.pid, pos);
00319                                   finish = false;
00320                                   break;
00321                               case Graph.LEFT:
00322                                   System.out.println("Bot should turn left");
00323                                   if (no_bots == 2) {
00324                                       if (!(other_bot.isSafePos(g.Left_turn(pos))))
       {
00325                                           break;
00326                                       }
00327                                   }
00328                                   x.turnLeft();
00329                                   finish = false;
00330                                   break;
00331                               case Graph.RIGHT:
00332                                   if (no_bots == 2) {
00333                                       if (!(other_bot.isSafePos(g.Right_turn(pos)))
      ) {
00334                                           break;
00335                                       }
00336                                   }
00337                                   x.turnRight();
00338                                   finish = false;
00339                                   break;
00340                               case Graph.STRAIGHT:
00341                                   if (no_bots == 2) {
00342                                       if (!(other_bot.isSafePos(g.Straight(pos))))
      {
00343                                           break;
00344                                       }
00345                                   }
00346
00347                                   x.gotoNextCross();
00348                                   finish = false;
00349                                   break;
00350                               case Graph.BACKWARD:
00351                                   if (no_bots == 2) {
00352                                       if (!(other_bot.isSafePos(g.Left_turn(pos))))
       {
00353                                           break;
00354                                       }
00355                                   }
00356
00357                                   x.turnBack();
00358                                   finish = false;
00359                                   break;
00360                           }
00361                       }
00362                   }
00363                   break;
00364               case IN_PROGRESS:
00365                   break;
00366           }
00367       }
00368
00373       public void bot_blocked(int bot_id) {
00374           try {
00375               Configure.getInstance();
00376           } catch (IOException ex) {
00377               Logger.getLogger(RequestHandler.class.getName()).log(Level.SEVERE, nu
      ll, ex);
```

```
00378         }
00379     }
00380 
00385     public void buzzer_off(int bot_id) {
00386     }
00387 
00393     public void assign_if_request_available(int bot_id) {
00394         if (DEBUG) {
00395             System.out.println("Checking for pending requests...");
00396         }
00397         Bot x = botList.get(bot_id);
00398         int request_id = dbh.getNextRequestFIFO();
00399         if (request_id == -1) {
00400             if (DEBUG) {
00401                 System.out.println("Setting bot to \"not is use\" mode...");
00402             }
00403             x.isInUse = false;
00404             return;
00405         } else {
00406             assign_bot_req(bot_id, request_id);
00407             if (DEBUG) {
00408                 System.out.println("Setting bot to \"in use\" mode...");
00409             }
00410             x.isInUse = true;
00411         }
00412     }
00413 
00424     public void assign_bot_req(int bot_id, int req_id) {
00425         Bot x = botList.get(bot_id);
00426         if (DEBUG) {
00427             System.out.println("Setting bot to \"in use\" mode");
00428         }
00429         x.isInUse = true;
00430         x.turnBack();
00431         x.pid = dbh.getPatientOfRequest(req_id);
00432         try {
00433             Configure.getInstance();
00434         } catch (IOException ex) {
00435             Logger.getLogger(RequestHandler.class.getName()).log(Level.SEVERE, nu
      ll, ex);
00436         }
00437         try {
00438             Runtime.getRuntime().exec("python send-sms.py " + Configure.SMS_NUM +
       " " + "Bot #" + bot_id + " is going to server the patient #" + (x.pid - 1));
00439         } catch (IOException ex) {
00440             Logger.getLogger(RequestHandler.class.getName()).log(Level.SEVERE, nu
      ll, ex);
00441         }
00442 
00443         dbh.addAssignment(req_id, bot_id);
00444         dbh.updateRequestStatus(req_id, "serving");
00445     }
00446 
00454     public void request_completed(int bot_id) {
00455         int request_id = dbh.getAssignedRequest(bot_id);
00456         int patient_id = dbh.getPatientOfRequest(request_id);
00457         try {
00458             Configure.getInstance();
00459         } catch (IOException ex) {
00460             Logger.getLogger(RequestHandler.class.getName()).log(Level.SEVERE, nu
      ll, ex);
00461         }
00462         dbh.deleteAssignment(request_id, true);
00463         dbh.updateRequestStatus(request_id, "done");
00464         Bot x = botList.get(bot_id);
00465         if (bot_id == Bot.bot1) {
00466             x.pid = Graph.SERVER_ID1;
```

```
00467         } else if (bot_id == Bot.bot2) {
00468 
00469             x.pid = Graph.SERVER_ID2;
00470         }
00471     }
00472 
00476     public static final int WATER = 0;
00480     public static final int MEDICINE = 1;
00484     public static int no_request = 0;
00485 
00490     public int get_new_request_id() {
00491         return no_request++;
00492     }
00493 
00498     public int get_a_free_bot() {
00499         if (DEBUG) {
00500             System.out.println("Searching for a free bot...");
00501         }
00502         Iterator it = botList.entrySet().iterator();
00503         while (it.hasNext()) {
00504             Entry t = (Entry) it.next();
00505             if (((Bot) t.getValue()).isBotIdle()) {
00506                 System.out.println("Bot " + ((Bot) t.getValue()).id + "  is free"
      );
00507                 return ((Bot) t.getValue()).id;
00508             } else {
00509                 if (((Bot) t.getValue()).isInUse) {
00510                     System.out.println(((Bot) t.getValue()).id + "is in use");
00511                 } else {
00512                     System.out.println(((Bot) t.getValue()).id + " is neither in
      use nor idle");
00513                 }
00514                 continue;
00515             }
00516         }
00517         System.out.println("No bot is free");
00518         return -1;
00519     }
00520 
00525     public void print_patient_command(char comm) {
00526         int command = (int) comm;
00527         if (DEBUG) {
00528             System.out.println("Patient is " + comm / 32);
00529         }
00530     }
00531     public static int request_absent = 127;
00532 
00542     public void execute_patient_request(char comm) {
00543         System.out.println("Patient request code is " + (int) comm);
00544         print_patient_command(comm);
00545         int comm_int = (int) comm;
00546         if (comm_int == request_absent) {
00547             return;
00548         }
00549         int patient_id = comm_int / 32;
00550 
00551 
00552         System.out.println("Patiend id is " + patient_id);
00553         int request = comm_int % 32;
00554         int request_id = get_new_request_id();
00555         dbh.addRequest(request_id, patient_id, Integer.toString(request), "pendin
      g");
00556         int free_bot = get_a_free_bot();
00557         if (free_bot != -1) {
00558             System.out.println("request for patient" + patient_id + "assigned to
      bot" + free_bot);
00559             assign_bot_req(free_bot, request_id);
```

```
00560         }
00561     }
00562 }
```

7.27 src/SERVER_CODE/pss/server/scheduling/PollingThread.java File Reference

Classes

- class pss::server::scheduling::PollingThread

Namespaces

- namespace pss::server::scheduling

7.28 src/SERVER_CODE/pss/server/scheduling/PollingThread.java

```
00001 /*
00002  * ================================================================================
       =======
00003  *
00004  *       Filename:  PollingThread.java
00005  *
00006  *           Date:  31st March, 2010
00007  *
00008  *        Version:  2.1
00009  *       Revision:  2.1
00010  *       Compiler:  javac
00011  *
00012  *        Authors:  Pritish Kamath, pritish.kamath@iitb.ac.in
00013  *                  Rohit Saraf   , rohitsaraf@iitb.ac.in
00014  *                  Ashish Mathew , ashishmathew@iitb.ac.in
00015  *                  Vivek Madan   , vivekmadan@iitb.ac.in
00016  *
00017  *        Company:  IIT Bombay
00018  *      Copyright:  ERTS Lab, IIT Bombay
00019  *
00020  * ================================================================================
       =======
00021  */
00022
00023 package pss.server.scheduling;
00024
00025 import java.io.IOException;
00026 import java.util.logging.Level;
00027 import java.util.logging.Logger;
00028 import pss.configuration.Configure;
00029 import pss.serialcomm.CommunicationAPI;
00030 import pss.server.Bot;
00031 import pss.server.RequestHandler;
00032
00037 public class PollingThread {
00038
00043     CommunicationAPI capi;
00047     public int poll_id = Bot.bot1;
00051     public int number_bots = 0;
00055     public static final int poll_patient = 0;
00059     public static final int poll_bot = 1;
00060
00064     public static int bot_or_patient = poll_bot;
00068     public static final Boolean DEBUG = false;
```

```
00072     public static int BOT_POLLING = 5;
00076     public static int PATIENT_POLLING = 128;
00081     private RequestHandler rh;
00082
00088     public PollingThread(CommunicationAPI capi) {
00089         this.capi = capi;
00090         bot_or_patient = poll_bot;
00091         try {
00092             Configure.getInstance();
00093         } catch (IOException ex) {
00094             ex.printStackTrace();
00095         }
00096         number_bots = Configure.NUM_BOTS;
00097     }
00098
00104     public Character poll_next(int poll_bot_or_patient) {
00105         while (true) {
00106             if (poll_bot_or_patient == poll_bot) {
00107                 if (DEBUG) {
00108                     System.out.println("polling bot " + poll_id);
00109                 }
00110                 int poll_message = poll_id * 32 + BOT_POLLING;
00111                 capi.send(Character.toString((char) poll_message));
00112                 Character c = capi.next_char_in_buffer();
00113                 if (c == 0xff) {
00114                     continue;
00115                 }
00116                 poll_id = (poll_id + 1) % number_bots;
00117                 return c;
00118             } else {
00119                 int poll_message = PATIENT_POLLING;
00120                 capi.send(Character.toString((char) poll_message));
00121                 Character c = capi.next_char_in_buffer();
00122                 if (c == 0xff) {
00123                     continue;
00124                 }
00125                 return c;
00126             }
00127         }
00128     }
00129 }
```

7.29 src/SERVER_CODE/pss/server/test/BotMotionTester1.java File Reference

Classes

- class pss::server::test::BotMotionTester1

Namespaces

- namespace pss::server::test

7.30 src/SERVER_CODE/pss/server/test/BotMotionTester1.java

```
00001 /*
00002  * ================================================================================
      =======
00003  *
00004  *       Filename:  BotMotionTester1.java
00005  *
00006  *           Date:  31st March, 2010
```

```
00007  *
00008  *        Version:  2.1
00009  *       Revision:  2.1
00010  *       Compiler:  javac
00011  *
00012  *        Authors:  Pritish Kamath, pritish.kamath@iitb.ac.in
00013  *                  Rohit Saraf   , rohitsaraf@iitb.ac.in
00014  *                  Ashish Mathew , ashishmathew@iitb.ac.in
00015  *                  Vivek Madan   , vivekmadan@iitb.ac.in
00016  *
00017  *        Company:  IIT Bombay
00018  *      Copyright:  ERTS Lab, IIT Bombay
00019  *
00020  * ================================================================================
       =======
00021  */
00022
00023 package pss.server.test;
00024
00025 import java.io.IOException;
00026 import java.util.logging.Level;
00027 import java.util.logging.Logger;
00028 import pss.configuration.Configure;
00029 import pss.serialcomm.CommunicationAPI;
00030 import pss.server.Bot;
00031 import pss.server.Graph;
00032 import pss.server.RequestHandler;
00033
00037 public class BotMotionTester1 {
00038
00042     public static final Boolean DEBUG = false;
00046     public static final Boolean mode1 = false;
00050     public static final Boolean mess_debug = true;
00051
00056     public static void printMess(Character message) {
00057         if (DEBUG) {
00058             System.out.println("Message is " + (int) message);
00059         }
00060         int mess_int = (int) message;
00061         int bot_id = mess_int / 32;
00062         int status = mess_int % 32;
00063         if (status == Graph.RIGHT) {
00064             if (mess_debug) {
00065                 System.out.println("Server says: Bot #" + bot_id + " should move
       right " + status);
00066             }
00067         }
00068         if (status == Graph.LEFT) {
00069             if (mess_debug) {
00070                 System.out.println("Server says: Bot #" + bot_id + " should move
       left " + status);
00071             }
00072         }
00073         if (status == Graph.STRAIGHT) {
00074             if (mess_debug) {
00075                 System.out.println("Server says: Bot #" + bot_id + " should move
       straight " + status);
00076             }
00077         }
00078         if (status == Graph.BACKWARD) {
00079             if (mess_debug) {
00080                 System.out.println("Server says: Bot #" + bot_id + " should move
       backward " + status);
00081             }
00082         }
00083
00084     }
```

```
00085
00097     public static void main(String[] args) {
00098         try {
00099             Configure.getInstance();
00100         } catch (IOException ex) {
00101             Logger.getLogger(BotMotionTester1.class.getName()).log(Level.SEVERE,
      null, ex);
00102         }
00103         CommunicationAPI capi = new CommunicationAPI(Configure.TEST_PORT1);
00104         capi.open();
00105         Boolean if_send = true;
00106         Character patient_mess = ((char) 3 * 32 + RequestHandler.WATER);
00107         //capi.send(Character.toString(patient_mess));
00108         int ack = Bot.bot1 * 32 + RequestHandler.ACK;
00109         char ack_message = (char) ack;
00110         //capi.send(Character.toString(ack_message));
00111         int inprogress = Bot.bot1 * 32 + RequestHandler.IN_PROGRESS;
00112         char inprogress_message = (char) inprogress;
00113         while (true) {
00114             if (DEBUG) {
00115                 System.out.println("Bot motion tester1:");
00116             }
00117             Character inp_mess = null;
00118             if (DEBUG) {
00119                 System.out.println(" getting next_char_in_buffer");
00120             }
00121             inp_mess = capi.next_char_in_buffer();
00122             if (DEBUG) {
00123                 System.out.println("Got next_char_");
00124             }
00125             if (Bot.getIdMess(inp_mess) == Bot.bot1) {
00126                 printMess(inp_mess);
00127                 if (((int) inp_mess) % 32 == Graph.BOT_POLLING) {
00128                     if (if_send) {
00129                         capi.send(Character.toString(ack_message));
00130                         if_send = false;
00131                         if (DEBUG) {
00132                             System.out.println("sending an ack ");
00133                         }
00134
00135                     } else {
00136                         capi.send(Character.toString(inprogress_message));
00137                         if_send = true;
00138                         if (DEBUG) {
00139                             System.out.println("sending in progress");
00140                         }
00141
00142                     }
00143                 }
00144             }
00145
00146         }
00147     }
00148 }
```

7.31 src/SERVER_CODE/pss/server/test/BotMotionTester2.java File Reference

Classes

- class pss::server::test::BotMotionTester2

Namespaces

- namespace pss::server::test

7.32 src/SERVER_CODE/pss/server/test/BotMotionTester2.java

```
00001 /*
00002  * ================================================================================
       =======
00003  *
00004  *       Filename:  BotMotionTester2.java
00005  *
00006  *           Date:  31st March, 2010
00007  *
00008  *        Version:  2.1
00009  *       Revision:  2.1
00010  *       Compiler:  javac
00011  *
00012  *        Authors:  Pritish Kamath, pritish.kamath@iitb.ac.in
00013  *                  Rohit Saraf   , rohitsaraf@iitb.ac.in
00014  *                  Ashish Mathew , ashishmathew@iitb.ac.in
00015  *                  Vivek Madan   , vivekmadan@iitb.ac.in
00016  *
00017  *        Company:  IIT Bombay
00018  *      Copyright:  ERTS Lab, IIT Bombay
00019  *
00020  * ================================================================================
       =======
00021  */
00022
00023 package pss.server.test;
00024
00025 import pss.serialcomm.CommunicationAPI;
00026 import pss.server.Bot;
00027 import pss.server.Graph;
00028 import pss.server.RequestHandler;
00029
00034 public class BotMotionTester2 {
00035
00036     public static final Boolean DEBUG = false;
00037     public static final Boolean mode1 = false;
00038     public static final Boolean mess_debug = true;
00039
00040     public static void print_mess(Character message) {
00041         if (DEBUG) {
00042             System.out.println("Message is " + (int) message);
00043         }
00044         int mess_int = (int) message;
00045         int bot_id = mess_int / 32;
00046         int instruction = mess_int % 32;
00047         if (instruction == Graph.RIGHT) {
00048             if (mess_debug) {
00049                 System.out.println("Server says: Bot #" + bot_id + " should move
       right " + instruction);
00050             }
00051         }
00052
00053         if (instruction == Graph.LEFT) {
00054             if (mess_debug) {
00055                 System.out.println("Server says: Bot #" + bot_id + " should move
       left " + instruction);
00056             }
00057         }
00058         if (instruction == Graph.STRAIGHT) {
00059             if (mess_debug) {
```

```
00060                 System.out.println("Server says: Bot #" + bot_id + " should move
      straight " + instruction);
00061             }
00062         }
00063         if (instruction == Graph.BACKWARD) {
00064             if (mess_debug) {
00065                 System.out.println("Server says: Bot #" + bot_id + " should move
      backward " + instruction);
00066             }
00067         }
00068     }
00069
00070
00071     public static void main(String[] args) {
00072         CommunicationAPI capi = new CommunicationAPI("/dev/ttyUSB0");
00073         capi.open();
00074         //initially patient sends message that I want water
00075         //Keep sending acks
00076         Boolean if_send = true;
00077         Character patient_mess = (char) 5 * 32 + RequestHandler.WATER;
00078         capi.send(Character.toString(patient_mess));
00079         int ack = Bot.bot2 * 32 + RequestHandler.ACK;
00080         char ack_message = (char) ack;
00081         //capi.send(Character.toString(ack_message));
00082         int inprogress = Bot.bot2 * 32 + RequestHandler.IN_PROGRESS;
00083         char inprogress_message = (char) inprogress;
00084        /* while (true) {
00085             Character inp_mess = null;
00086             if (DEBUG) {
00087                 System.out.println(" getting next_char_in_buffer");
00088             }
00089             inp_mess = capi.next_char_in_buffer();
00090
00091             if (Bot.get_id_mess(inp_mess) == Bot.bot2) {
00092                 print_mess(inp_mess);
00093
00094                 if (if_send) {
00095                     capi.send(Character.toString(ack_message));
00096                     if_send = false;
00097                     if (DEBUG) {
00098                         System.out.println("sending an ack ");
00099                     }
00100
00101                 } else {
00102                     capi.send(Character.toString(inprogress_message));
00103                     if_send = true;
00104                     if (DEBUG) {
00105                         System.out.println("sending in progress");
00106                     }
00107
00108                 }
00109             }
00110         }*/
00111     }
00112 }
```

7.33 src/SERVER_CODE/pss/server/test/GraphTester.java File Reference

Classes

- class pss::server::test::GraphTester

Namespaces

- namespace pss::server::test

7.34 src/SERVER_CODE/pss/server/test/GraphTester.java

```
00001 /*
00002  * ================================================================================
      =======
00003  *
00004  *       Filename:  GraphTester.java
00005  *
00006  *           Date:  31st March, 2010
00007  *
00008  *        Version:  2.1
00009  *       Revision:  2.1
00010  *       Compiler:  javac
00011  *
00012  *        Authors:  Pritish Kamath, pritish.kamath@iitb.ac.in
00013  *                  Rohit Saraf   , rohitsaraf@iitb.ac.in
00014  *                  Ashish Mathew , ashishmathew@iitb.ac.in
00015  *                  Vivek Madan   , vivekmadan@iitb.ac.in
00016  *
00017  *        Company:  IIT Bombay
00018  *      Copyright:  ERTS Lab, IIT Bombay
00019  *
00020  * ================================================================================
      =======
00021  */
00022
00023 package pss.server.test;
00024
00025 import pss.server.Graph;
00026 import pss.server.Position;
00027
00033 public class GraphTester {
00034
00035     public static void main(String[] args) {
00036         Graph g =new Graph();
00037         g.init_graph();
00038         Position curr = new Position();
00039         curr.present =0;
00040         curr.orientation=Graph.WEST;
00041         int patient_id=-2;
00042         g.move_the_bot(curr,patient_id);
00043     }
00044 }
```

7.35 src/SERVER_CODE/pss/server/test/Patient_simulator.java File Reference

Classes

- class pss::server::test::Patient_simulator

Namespaces

- namespace pss::server::test

7.36 src/SERVER_CODE/pss/server/test/Patient_simulator.java

```
00001 /*
00002  * ================================================================================
       =======
00003  *
00004  *       Filename:  Patient_simulator.java
00005  *
00006  *           Date:  31st March, 2010
00007  *
00008  *        Version:  2.1
00009  *       Revision:  2.1
00010  *       Compiler:  javac
00011  *
00012  *        Authors:  Pritish Kamath, pritish.kamath@iitb.ac.in
00013  *                  Rohit Saraf   , rohitsaraf@iitb.ac.in
00014  *                  Ashish Mathew , ashishmathew@iitb.ac.in
00015  *                  Vivek Madan   , vivekmadan@iitb.ac.in
00016  *
00017  *        Company:  IIT Bombay
00018  *      Copyright:  ERTS Lab, IIT Bombay
00019  *
00020  * ================================================================================
       =======
00021  */
00022
00023 package pss.server.test;
00024
00025 import pss.serialcomm.CommunicationAPI;
00026 import pss.server.RequestHandler;
00027
00032 public class Patient_simulator {
00033
00034     public static void main(String[] args) {
00035         CommunicationAPI capi = new CommunicationAPI("/dev/ttyUSB1");
00036         capi.open();
00037         Character patient_mess = (char) 6 * 32 + RequestHandler.WATER;
00038         capi.send(Character.toString(patient_mess));
00039     }
00040 }
```

7.37 src/SERVER_CODE/pss/server/test/SimpleRead.java File Reference

Classes

- class pss::server::test::SimpleRead

Namespaces

- namespace pss::server::test

7.38 src/SERVER_CODE/pss/server/test/SimpleRead.java

```
00001 /*
00002  * ================================================================================
       =======
00003  *
00004  *       Filename:  SimpleRead.java
00005  *
00006  *           Date:  31st March, 2010
00007  *
```

```
00008  *        Version:  2.1
00009  *       Revision:  2.1
00010  *       Compiler:  javac
00011  *
00012  *        Authors:  Pritish Kamath, pritish.kamath@iitb.ac.in
00013  *                  Rohit Saraf   , rohitsaraf@iitb.ac.in
00014  *                  Ashish Mathew , ashishmathew@iitb.ac.in
00015  *                  Vivek Madan   , vivekmadan@iitb.ac.in
00016  *
00017  *        Company:  Oracle
00018  *      Copyright:  Oracle
00019  *
00020  * ================================================================================
       =======
00021  */
00022
00023 package pss.server.test;
00024 import java.io.*;
00025 import java.util.*;
00026 import gnu.io.*;
00027
00028
00029 public class SimpleRead implements Runnable, SerialPortEventListener {
00030
00031    static CommPortIdentifier portId;
00032    static Enumeration portList;
00033    InputStream inputStream;
00034    SerialPort serialPort;
00035    Thread readThread;
00036
00037
00038    public static void main(String[] args) {
00039       boolean portFound = false;
00040       String defaultPort = "/dev/ttyUSB1";
00041
00042       if (args.length > 0) {
00043          defaultPort = args[0];
00044       }
00045
00046       portList = CommPortIdentifier.getPortIdentifiers();
00047
00048       while (portList.hasMoreElements()) {
00049          portId = (CommPortIdentifier) portList.nextElement();
00050          if (portId.getPortType() == CommPortIdentifier.PORT_SERIAL) {
00051             if (portId.getName().equals(defaultPort)) {
00052                System.out.println("Found port: " + defaultPort);
00053                portFound = true;
00054                SimpleRead reader = new SimpleRead();
00055                System.out.println("Creating new SimpleRead");
00056             }
00057          }
00058       }
00059       if (!portFound) {
00060          System.out.println("port " + defaultPort + " not found.");
00061       }
00062
00063    }
00064
00065    public SimpleRead() {
00066       try {
00067          serialPort = (SerialPort) portId.open("SimpleReadApp", 20000);
00068       } catch (PortInUseException e) {
00069       }
00070       try {
00071          inputStream = serialPort.getInputStream();
00072       } catch (IOException e) {
00073       }
```

```
00074
00075       try {
00076          serialPort.addEventListener(this);
00077       } catch (TooManyListenersException e) {
00078       }
00079
00080       serialPort.notifyOnDataAvailable(true);
00081
00082       try {
00083          serialPort.setSerialPortParams(9600, SerialPort.DATABITS_8,
00084                  SerialPort.STOPBITS_1,
00085                  SerialPort.PARITY_NONE);
00086       } catch (UnsupportedCommOperationException e) {
00087       }
00088
00089       readThread = new Thread(this);
00090       readThread.start();
00091    }
00092
00093    public void run() {
00094       try {
00095          while (true) {
00096             Thread.sleep(2000);
00097          }
00098       } catch (InterruptedException e) {
00099       }
00100    }
00101
00102    public void serialEvent(SerialPortEvent event) {
00103       switch (event.getEventType()) {
00104
00105          case SerialPortEvent.BI:
00106
00107          case SerialPortEvent.OE:
00108
00109          case SerialPortEvent.FE:
00110
00111          case SerialPortEvent.PE:
00112
00113          case SerialPortEvent.CD:
00114
00115          case SerialPortEvent.CTS:
00116
00117          case SerialPortEvent.DSR:
00118
00119          case SerialPortEvent.RI:
00120
00121          case SerialPortEvent.OUTPUT_BUFFER_EMPTY:
00122             break;
00123
00124          case SerialPortEvent.DATA_AVAILABLE:
00125             byte[] readBuffer = new byte[1];
00126             List<Character> lc = new ArrayList<Character>();
00127             try {
00128                while (inputStream.available() > 0) {
00129                   int numBytes = inputStream.read();
00130                   if (((char) numBytes) != '\n') {
00131                      lc.add((char) numBytes);
00132                   }
00133                }
00134                for (int i = 0; i < lc.size(); i++) {
00135                   System.out.print(lc.get(i));
00136                }
00137                //System.out.println("");
00138                System.out.flush();
00139                //System.out.print(new String(readBuffer));
00140             } catch (IOException e) {
```

```
00141             }
00142
00143             break;
00144         }
00145     }
00146 }
```

7.39 src/SERVER_CODE/pss/server/test/SimpleWrite.java File Reference

Classes

- class pss::server::test::SimpleWrite

Namespaces

- namespace pss::server::test

7.40 src/SERVER_CODE/pss/server/test/SimpleWrite.java

```
00001 /*
00002  * ================================================================================
      =======
00003  *
00004  *       Filename:  SimpleWrite.java
00005  *
00006  *           Date:  31st March, 2010
00007  *
00008  *        Version:  2.1
00009  *       Revision:  2.1
00010  *       Compiler:  javac
00011  *
00012  *        Authors:  Pritish Kamath, pritish.kamath@iitb.ac.in
00013  *                  Rohit Saraf   , rohitsaraf@iitb.ac.in
00014  *                  Ashish Mathew , ashishmathew@iitb.ac.in
00015  *                  Vivek Madan   , vivekmadan@iitb.ac.in
00016  *
00017  *        Company:  Oracle
00018  *      Copyright:  Oracle
00019  *
00020  * ================================================================================
      =======
00021  */
00022
00023 package pss.server.test;
00024
00025 import java.io.*;
00026 import java.util.*;
00027 import gnu.io.*;
00028
00029 public class SimpleWrite {
00030
00031    static Enumeration portList;
00032    static CommPortIdentifier portId;
00033    static String messageString = "rohit kumar saraf";
00034    static SerialPort serialPort;
00035    static OutputStream outputStream;
00036    static boolean outputBufferEmptyFlag = false;
00037
00038    public static void main(String[] args) {
00039       boolean portFound = false;
00040       String defaultPort = "/dev/ttyUSB1";
```

```
00041
00042       if (args.length > 0) {
00043          defaultPort = args[0];
00044       }
00045
00046       portList = CommPortIdentifier.getPortIdentifiers();
00047
00048       while (portList.hasMoreElements()) {
00049          portId = (CommPortIdentifier) portList.nextElement();
00050
00051          if (portId.getPortType() == CommPortIdentifier.PORT_SERIAL) {
00052
00053             if (portId.getName().equals(defaultPort)) {
00054                System.out.println("Found port " + defaultPort);
00055
00056                portFound = true;
00057
00058                try {
00059                   serialPort = (SerialPort) portId.open("SimpleWrite", 2000);
00060                } catch (PortInUseException e) {
00061                   System.out.println("Port in use.");
00062
00063                   continue;
00064                }
00065
00066                try {
00067                   outputStream = serialPort.getOutputStream();
00068                } catch (IOException e) {
00069                }
00070
00071                try {
00072                   serialPort.setSerialPortParams(9600,
00073                           SerialPort.DATABITS_8,
00074                           SerialPort.STOPBITS_1,
00075                           SerialPort.PARITY_NONE);
00076                } catch (UnsupportedCommOperationException e) {
00077                }
00078
00079
00080                try {
00081                   serialPort.notifyOnOutputEmpty(true);
00082                } catch (Exception e) {
00083
00084                   System.out.println("Error setting event notification");
00085                   System.out.println(e.toString());
00086                   System.exit(-1);
00087                }
00088                System.out.println(
00089                        "Writing \"" + messageString + "\" to "
00090                        + serialPort.getName());
00091                try {
00092                   outputStream.write((String.valueOf(messageString)).getBytes());

00093                   outputStream.flush();
00094                } catch (IOException e) {
00095                }
00096                try {
00097                   Thread.sleep(2000);  // Be sure data is xferred before closing
00098                } catch (Exception e) {
00099                }
00100                serialPort.close();
00101             }
00102          }
00103       }
00104       if (!portFound) {
00105          System.out.println("port " + defaultPort + " not found.");
00106       }
```

```
00107     }
00108 }
```

7.41 src/SERVER_CODE/pss/server/Utils.java File Reference

Classes

- class pss::server::Utils

Namespaces

- namespace pss::server

7.42 src/SERVER_CODE/pss/server/Utils.java

```
00001 /*
00002  * ================================================================================
       =======
00003  *
00004  *       Filename:  Utils.java
00005  *
00006  *           Date:  31st March, 2010
00007  *
00008  *        Version:  2.1
00009  *       Revision:  2.1
00010  *       Compiler:  javac
00011  *
00012  *        Authors:  Pritish Kamath, pritish.kamath@iitb.ac.in
00013  *                  Rohit Saraf   , rohitsaraf@iitb.ac.in
00014  *                  Ashish Mathew , ashishmathew@iitb.ac.in
00015  *                  Vivek Madan   , vivekmadan@iitb.ac.in
00016  *
00017  *        Company:  IIT Bombay
00018  *      Copyright:  ERTS Lab, IIT Bombay
00019  *
00020  * ================================================================================
       =======
00021  */
00022
00023 package pss.server;
00024
00025 import javax.swing.text.GapContent;
00026
00032 public class Utils {
00033
00040     public static char left_message(int bot_id) {
00041         int message = bot_id * 32 + Graph.LEFT;
00042         return ((char) message);
00043     }
00044
00051     public static char right_message(int bot_id) {
00052         int message = bot_id * 32 + Graph.RIGHT;
00053         return ((char) message);
00054     }
00055
00062     public static char straight_message(int bot_id) {
00063         int message = bot_id * 32 + Graph.STRAIGHT;
00064         return ((char) message);
00065     }
00066
00073     public static char back_message(int bot_id) {
```

```
00074         int message = bot_id * 32 + Graph.BACKWARD;
00075         System.out.println((char) message);
00076         return ((char) message);
00077     }
00078 }
```

Index

ACK
pss::server::RequestHandler, 43
ADC_Conversion
bot_motion.h, 64
adc_init
bot_motion.h, 64
adc_pin_config
bot_motion.h, 64
ADC_Value
bot_motion.h, 67
addAssignment
pss::server::database::DBHandler, 21
addPatient
pss::server::database::DBHandler, 21
addRequest
pss::server::database::DBHandler, 21
assign_bot_req
pss::server::RequestHandler, 40
assign_if_request_available
pss::server::RequestHandler, 40

Back
pss::server::Graph, 27
back_message
pss::server::Utils, 49
BACKWARD
pss::server::Graph, 30
Bit
IR.c, 75
BIT_MACROS
bot.c, 51
BLOCKED
bot_motion.h, 62
pss::server::RequestHandler, 43
Bot
pss::server::Bot, 6
bot.c
BIT_MACROS, 51
GetBit, 51
GO_UPTO_CROSS, 51
ID_MASK, 51
init_devices_1, 53
INST_MASK, 51
main, 53
MY_ID, 51
NOTHING, 52
POLLING, 52
ResetBit, 52
SetBit, 52
SIGNAL, 53
TURN_AROUND, 52
TURN_LEFT, 52
TURN_RIGHT, 52
USART_Init, 53
bot1
pss::server::Bot, 8
bot2
pss::server::Bot, 8
bot_2.c
GetBit, 56
GO_UPTO_CROSS, 56
ID_MASK, 57
init_devices_1, 58
INST_MASK, 57
main, 58
MY_ID, 57
NOTHING, 57
POLLING, 57
ResetBit, 57
SetBit, 57
SIGNAL, 58
TURN_AROUND, 57
TURN_LEFT, 57
TURN_RIGHT, 58
USART_Init, 58
bot_blocked
pss::server::RequestHandler, 40
bot_motion.h
ADC_Conversion, 64
adc_init, 64
adc_pin_config, 64
ADC_Value, 67
BLOCKED, 62
buzzer_off, 64
buzzer_on, 64
CENTER_SENSOR, 62
Center_white_line, 67
CONT_BLACK, 62
FCPU, 62
flag, 67
forward, 64
FRONT_IR_SENSOR, 63
Front_IR_Sensor, 67
GetBit, 63
go_distance, 65
go_upto_next_cross, 65
IDLE, 63
init_devices, 65
ISR, 65
lcd_port_config, 65
left_position_encoder_interrupt_init, 65

LEFT_SENSOR, 63
Left_white_line, 68
motion_pin_config, 65
motion_set, 66
motors_delay, 66
port_init, 66
print_sensor, 66
print_sensor_data, 66
PROCESSING, 63
read_sensors, 66
reset_shaft_counters, 66
ResetBit, 63
right_position_encoder_interrupt_init, 66
RIGHT_SENSOR, 63
Right_white_line, 68
ROTATE_THRESHOLD, 63
SetBit, 63
stop, 66
timer5_init, 67
turn_left, 67
turn_right, 67
velocity, 67
W_THRESHOLD, 64
W_THRESHOLD_STOP, 64
bot_or_patient
pss::server::scheduling::PollingThread, 35
BOT_POLLING
pss::server::Graph, 30
pss::server::scheduling::PollingThread, 35
botList
pss::server::RequestHandler, 43
buzzer_off
bot_motion.h, 64
pss::server::RequestHandler, 40
buzzer_on
bot_motion.h, 64

capi
pss::server::Bot, 9
pss::server::RequestHandler, 43
pss::server::scheduling::PollingThread, 35
cbit
lcd.h, 81
CENTER_SENSOR
bot_motion.h, 62
Center_white_line
bot_motion.h, 67
CLEAR_DB
pss::configuration::Configure, 18
close
pss::serialcomm::CommunicationAPI, 14
closeConnection
pss::server::database::DBHandler, 22
code
IR.c, 77
CODESIZE
IR.c, 75
CommunicationAPI
pss::serialcomm::CommunicationAPI, 14
CONT_BLACK
bot_motion.h, 62
createConnection
pss::server::database::DBHandler, 22
createDatabase
pss::server::database::DBHandler, 22
createInstance
pss::configuration::Configure, 17
currpos
pss::server::Bot, 9

DB_DRIVER
pss::configuration::Configure, 18
DB_PASS
pss::configuration::Configure, 19
DB_UN
pss::configuration::Configure, 19
DB_URL
pss::configuration::Configure, 19
dbh
pss::server::Bot, 9
pss::server::RequestHandler, 43
DEBUG
pss::serialcomm::CommunicationAPI, 15
pss::server::Bot, 9
pss::server::database::DBHandler, 25
pss::server::Graph, 30
pss::server::RequestHandler, 43
pss::server::scheduling::PollingThread, 35
pss::server::test::BotMotionTester1, 11
pss::server::test::BotMotionTester2, 12
defaultPort
pss::serialcomm::CommunicationAPI, 15
deleteAssignment
pss::server::database::DBHandler, 22
deleteDatabase
pss::server::database::DBHandler, 23
deletePatient
pss::server::database::DBHandler, 23
deleteRequest
pss::server::database::DBHandler, 23
distance_pos_ori_pos
pss::server::Graph, 30

EAST
pss::server::Graph, 30
EN
lcd.h, 81
execute_bot_Command

pss::server::RequestHandler, 40
execute_patient_request
pss::server::RequestHandler, 41
executeStatement
pss::server::database::DBHandler, 23
executeUpdate
pss::server::database::DBHandler, 23

FCPU
bot_motion.h, 62
lcd.h, 81
find_distance
pss::server::Graph, 27
FINISH
pss::server::Graph, 30
flag
bot_motion.h, 67
forward
bot_motion.h, 64
FRONT_IR_SENSOR
bot_motion.h, 63
Front_IR_Sensor
bot_motion.h, 67

g
pss::server::Bot, 9
pss::server::RequestHandler, 43
get_a_free_bot
pss::server::RequestHandler, 41
get_new_request_id
pss::server::RequestHandler, 42
getAssignedRequest
pss::server::database::DBHandler, 24
GetBit
bot.c, 51
bot_2.c, 56
bot_motion.h, 63
IR.c, 75
getIdMess
pss::server::Bot, 6
getInstance
pss::configuration::Configure, 17
getNextCharFromBufferIfPresent
pss::serialcomm::CommunicationAPI, 14
getNextRequestFIFO
pss::server::database::DBHandler, 24
getPatientOfRequest
pss::server::database::DBHandler, 24
go_distance
bot_motion.h, 65
GO_UPTO_CROSS
bot.c, 51
bot_2.c, 56
go_upto_next_cross
bot_motion.h, 65
gotoNextCross
pss::server::Bot, 6
Graph
pss::server::Graph, 26
grph
pss::server::Graph, 30

hundred
lcd.h, 83

i
lcd.h, 83
id
pss::server::Bot, 9
ID_MASK
bot.c, 51
bot_2.c, 57
IR.c, 75
IDLE
bot_motion.h, 63
in_buffer
pss::serialcomm::CommunicationAPI, 15
IN_PROGRESS
pss::server::RequestHandler, 44
INFINTY
pss::server::Graph, 31
init_devices
bot_motion.h, 65
init_devices_1
bot.c, 53
bot_2.c, 58
init_graph
pss::server::Graph, 27
init_ports
IR.c, 76
lcd.h, 82
inputStream
pss::serialcomm::CommunicationAPI, 15
pss::server::test::SimpleRead, 46
INST_MASK
bot.c, 51
bot_2.c, 57
IR.c, 75
instance
pss::configuration::Configure, 19
IR.c
Bit, 75
code, 77
CODESIZE, 75
GetBit, 75
ID_MASK, 75
init_ports, 76
INST_MASK, 75

IR_Get_Input_Vector, 76
IR_init_devices, 76
IR_read, 76
ISR, 76
learn, 76
main, 76
MY_ID, 75
num_bits_matched, 77
Pat, 77
patient_id, 77
ResetBit, 75
SetBit, 76
USART_Init, 77
IR_Get_Input_Vector
IR.c, 76
IR_init_devices
IR.c, 76
IR_read
IR.c, 76
isAtHome
pss::server::Bot, 6
isBotIdle
pss::server::Bot, 7
isInUse
pss::server::Bot, 9
isOriginalOrientation
pss::server::Bot, 7
ISR
bot_motion.h, 65
IR.c, 76
isSafePos
pss::server::Bot, 7

lcd.h
cbit, 81
EN, 81
FCPU, 81
hundred, 83
i, 83
init_ports, 82
lcd_cursor, 82
lcd_home, 82
lcd_init, 82
lcd_port, 81
lcd_print, 82
lcd_reset_4bit, 83
lcd_set_4bit, 83
lcd_string, 83
lcd_wr_char, 83
lcd_wr_command, 83
million, 83
RS, 82
RW, 82
sbit, 82
temp, 84
tens, 84
thousand, 84
unit, 84
lcd_cursor
lcd.h, 82
lcd_home
lcd.h, 82
lcd_init
lcd.h, 82
lcd_port
lcd.h, 81
lcd_port_config
bot_motion.h, 65
lcd_print
lcd.h, 82
lcd_reset_4bit
lcd.h, 83
lcd_set_4bit
lcd.h, 83
lcd_string
lcd.h, 83
lcd_wr_char
lcd.h, 83
lcd_wr_command
lcd.h, 83
learn
IR.c, 76
LEFT
pss::server::Graph, 31
left_message
pss::server::Utils, 49
left_position_encoder_interrupt_init
bot_motion.h, 65
LEFT_SENSOR
bot_motion.h, 63
Left_turn
pss::server::Graph, 27
Left_white_line
bot_motion.h, 68
loadProperties
pss::configuration::Configure, 18

main
bot.c, 53
bot_2.c, 58
IR.c, 76
pss::serialcomm::CommunicationAPI, 14
pss::server::database::DBHandler, 24
pss::server::RequestHandler, 42
pss::server::test::BotMotionTester1, 11
pss::server::test::BotMotionTester2, 12
pss::server::test::GraphTester, 33
pss::server::test::Patient_simulator, 34

pss::server::test::SimpleRead, 46
pss::server::test::SimpleWrite, 48
MEDICINE
pss::server::RequestHandler, 44
mess_debug
pss::server::test::BotMotionTester1, 11
pss::server::test::BotMotionTester2, 12
messageString
pss::serialcomm::CommunicationAPI::Sender, 45
pss::server::test::SimpleWrite, 48
million
lcd.h, 83
mode1
pss::server::test::BotMotionTester1, 11
pss::server::test::BotMotionTester2, 13
motion_pin_config
bot_motion.h, 65
motion_set
bot_motion.h, 66
motors_delay
bot_motion.h, 66
move_the_bot
pss::server::Graph, 28
MY_ID
bot.c, 51
bot_2.c, 57
IR.c, 75

next_char_in_buffer
pss::serialcomm::CommunicationAPI, 14
no_bots
pss::server::RequestHandler, 44
no_calls
pss::server::Graph, 31
no_request
pss::server::RequestHandler, 44
NOPATH
pss::server::Graph, 31
NORTH
pss::server::Graph, 31
NOTHING
bot.c, 52
bot_2.c, 57
num_bits_matched
IR.c, 77
NUM_BOTS
pss::configuration::Configure, 19
number_bots
pss::server::scheduling::PollingThread, 35

obstruction
pss::server::Bot, 9
open
pss::serialcomm::CommunicationAPI, 15
orientation
pss::server::Position, 37
outputBufferEmptyFlag
pss::serialcomm::CommunicationAPI, 15
pss::server::test::SimpleWrite, 48
outputStream
pss::serialcomm::CommunicationAPI, 16
pss::server::test::SimpleWrite, 48

Pat
IR.c, 77
patient_id
IR.c, 77
PATIENT_POLLING
pss::server::Graph, 31
pss::server::scheduling::PollingThread, 36
patientPos
pss::server::Graph, 31
pid
pss::server::Bot, 10
poll_bot
pss::server::scheduling::PollingThread, 36
poll_id
pss::server::scheduling::PollingThread, 36
poll_next
pss::server::scheduling::PollingThread, 35
poll_patient
pss::server::scheduling::PollingThread, 36
POLLING
bot.c, 52
bot_2.c, 57
PollingThread
pss::server::scheduling::PollingThread, 34
port_init
bot_motion.h, 66
portId
pss::serialcomm::CommunicationAPI, 16
pss::server::test::SimpleRead, 46
pss::server::test::SimpleWrite, 48
portList
pss::serialcomm::CommunicationAPI, 16
pss::server::test::SimpleRead, 47
pss::server::test::SimpleWrite, 48
pos_after_action
pss::server::Graph, 28
Position
pss::server::Position, 37
present
pss::server::Position, 37
print_mess
pss::server::test::BotMotionTester2, 12
print_patient_command
pss::server::RequestHandler, 42

print_sensor
bot_motion.h, 66
print_sensor_data
bot_motion.h, 66
printBotPos
pss::server::Bot, 7
printMess
pss::server::test::BotMotionTester1, 11
PROCESSING
bot_motion.h, 63
pss, 3
pss::configuration, 4
pss::configuration::Configure, 16
CLEAR_DB, 18
createInstance, 17
DB_DRIVER, 18
DB_PASS, 19
DB_UN, 19
DB_URL, 19
getInstance, 17
instance, 19
loadProperties, 18
NUM_BOTS, 19
setValues, 18
SMS_MSG, 19
SMS_NUM, 19
SMS_PASS, 19
SMS_UN, 19
TEST_PORT1, 20
TEST_PORT2, 20
ZIGBEE_PORT, 20
pss::serialcomm, 4
pss::serialcomm::CommunicationAPI, 13
close, 14
CommunicationAPI, 14
DEBUG, 15
defaultPort, 15
getNextCharFromBufferIfPresent, 14
in_buffer, 15
inputStream, 15
main, 14
next_char_in_buffer, 14
open, 15
outputBufferEmptyFlag, 15
outputStream, 16
portId, 16
portList, 16
readThread, 16
receive, 15
send, 15
serialPort, 16
pss::serialcomm::CommunicationAPI::Receiver, 37
Receiver, 38
run, 38
serialEvent, 38
pss::serialcomm::CommunicationAPI::Sender, 44
messageString, 45
run, 45
Sender, 45
pss::server, 4
pss::server::Bot, 5
Bot, 6
bot1, 8
bot2, 8
capi, 9
currpos, 9
dbh, 9
DEBUG, 9
g, 9
getIdMess, 6
gotoNextCross, 6
id, 9
isAtHome, 6
isBotIdle, 7
isInUse, 9
isOriginalOrientation, 7
isSafePos, 7
obstruction, 9
pid, 10
printBotPos, 7
running, 10
setPosition, 8
stationary, 10
status, 10
turnBack, 8
turnLeft, 8
turnRight, 8
pss::server::database, 4
pss::server::database::DBHandler, 20
addAssignment, 21
addPatient, 21
addRequest, 21
closeConnection, 22
createConnection, 22
createDatabase, 22
DEBUG, 25
deleteAssignment, 22
deleteDatabase, 23
deletePatient, 23
deleteRequest, 23
executeStatement, 23
executeUpdate, 23
getAssignedRequest, 24
getNextRequestFIFO, 24
getPatientOfRequest, 24
main, 24
updateRequestStatus, 25
pss::server::Graph, 25

Back, 27
BACKWARD, 30
BOT_POLLING, 30
DEBUG, 30
distance_pos_ori_pos, 30
EAST, 30
find_distance, 27
FINISH, 30
Graph, 26
grph, 30
INFINTY, 31
init_graph, 27
LEFT, 31
Left_turn, 27
move_the_bot, 28
no_calls, 31
NOPATH, 31
NORTH, 31
PATIENT_POLLING, 31
patientPos, 31
pos_after_action, 28
RIGHT, 31
Right_turn, 28
search, 29
SERVER_ID1, 32
SERVER_ID1_orientation, 32
SERVER_ID1_position, 32
SERVER_ID2, 32
SERVER_ID2_orientation, 32
SERVER_ID2_position, 32
SOUTH, 32
STRAIGHT, 32
Straight, 29
WEST, 32
pss::server::Position, 36
orientation, 37
Position, 37
present, 37
pss::server::RequestHandler, 38
ACK, 43
assign_bot_req, 40
assign_if_request_available, 40
BLOCKED, 43
bot_blocked, 40
botList, 43
buzzer_off, 40
capi, 43
dbh, 43
DEBUG, 43
execute_bot_Command, 40
execute_patient_request, 41
g, 43
get_a_free_bot, 41
get_new_request_id, 42
IN_PROGRESS, 44
main, 42
MEDICINE, 44
no_bots, 44
no_request, 44
print_patient_command, 42
request_absent, 44
request_completed, 42
RequestHandler, 39
WATER, 44
pss::server::scheduling, 4
pss::server::scheduling::PollingThread, 34
bot_or_patient, 35
BOT_POLLING, 35
capi, 35
DEBUG, 35
number_bots, 35
PATIENT_POLLING, 36
poll_bot, 36
poll_id, 36
poll_next, 35
poll_patient, 36
PollingThread, 34
pss::server::test, 5
pss::server::test::BotMotionTester1, 10
DEBUG, 11
main, 11
mess_debug, 11
mode1, 11
printMess, 11
pss::server::test::BotMotionTester2, 12
DEBUG, 12
main, 12
mess_debug, 12
mode1, 13
print_mess, 12
pss::server::test::GraphTester, 33
main, 33
pss::server::test::Patient_simulator, 33
main, 34
pss::server::test::SimpleRead, 45
inputStream, 46
main, 46
portId, 46
portList, 47
readThread, 47
run, 46
serialEvent, 46
serialPort, 47
SimpleRead, 46
pss::server::test::SimpleWrite, 47
main, 48
messageString, 48
outputBufferEmptyFlag, 48

outputStream, 48
portId, 48
portList, 48
serialPort, 48
pss::server::Utils, 49
back_message, 49
left_message, 49
right_message, 49
straight_message, 50

read_sensors
bot_motion.h, 66
readThread
pss::serialcomm::CommunicationAPI, 16
pss::server::test::SimpleRead, 47
receive
pss::serialcomm::CommunicationAPI, 15
Receiver
pss::serialcomm::CommunicationAPI::Receiver, 38
request_absent
pss::server::RequestHandler, 44
request_completed
pss::server::RequestHandler, 42
RequestHandler
pss::server::RequestHandler, 39
reset_shaft_counters
bot_motion.h, 66
ResetBit
bot.c, 52
bot_2.c, 57
bot_motion.h, 63
IR.c, 75
RIGHT
pss::server::Graph, 31
right_message
pss::server::Utils, 49
right_position_encoder_interrupt_init
bot_motion.h, 66
RIGHT_SENSOR
bot_motion.h, 63
Right_turn
pss::server::Graph, 28
Right_white_line
bot_motion.h, 68
ROTATE_THRESHOLD
bot_motion.h, 63
RS
lcd.h, 82
run
pss::serialcomm::CommunicationAPI::Receiver, 38
pss::serialcomm::CommunicationAPI::Sender, 45
pss::server::test::SimpleRead, 46
running
pss::server::Bot, 10
RW
lcd.h, 82

sbit
lcd.h, 82
search
pss::server::Graph, 29
send
pss::serialcomm::CommunicationAPI, 15
Sender
pss::serialcomm::CommunicationAPI::Sender, 45
serialEvent
pss::serialcomm::CommunicationAPI::Receiver, 38
pss::server::test::SimpleRead, 46
serialPort
pss::serialcomm::CommunicationAPI, 16
pss::server::test::SimpleRead, 47
pss::server::test::SimpleWrite, 48
SERVER_ID1
pss::server::Graph, 32
SERVER_ID1_orientation
pss::server::Graph, 32
SERVER_ID1_position
pss::server::Graph, 32
SERVER_ID2
pss::server::Graph, 32
SERVER_ID2_orientation
pss::server::Graph, 32
SERVER_ID2_position
pss::server::Graph, 32
SetBit
bot.c, 52
bot_2.c, 57
bot_motion.h, 63
IR.c, 76
setPosition
pss::server::Bot, 8
setValues
pss::configuration::Configure, 18
SIGNAL
bot.c, 53
bot_2.c, 58
SimpleRead
pss::server::test::SimpleRead, 46
SMS_MSG
pss::configuration::Configure, 19
SMS_NUM
pss::configuration::Configure, 19
SMS_PASS

pss::configuration::Configure, 19
SMS_UN
pss::configuration::Configure, 19
SOUTH
pss::server::Graph, 32
src/BOT_CODE/bot.c, 50
src/BOT_CODE/bot.d, 56
src/BOT_CODE/bot_2.c, 56
src/BOT_CODE/bot_motion.h, 61
src/BOT_CODE/IR.c, 74
src/BOT_CODE/lcd.h, 80
src/SERVER_CODE/pss/configuration/Configure.java, 88
src/SERVER_CODE/pss/serialcomm/CommunicationAPI.java, 89
src/SERVER_CODE/pss/server/Bot.java, 93
src/SERVER_CODE/pss/server/database/DBHandler.java, 96
src/SERVER_CODE/pss/server/Graph.java, 102
src/SERVER_CODE/pss/server/Position.java, 109
src/SERVER_CODE/pss/server/RequestHandler.java, 109
src/SERVER_CODE/pss/server/scheduling/PollingThread.java, 116
src/SERVER_CODE/pss/server/test/BotMotionTester1.java, 117
src/SERVER_CODE/pss/server/test/BotMotionTester2.java, 119
src/SERVER_CODE/pss/server/test/GraphTester.java, 121
src/SERVER_CODE/pss/server/test/Patient_-simulator.java, 122
src/SERVER_CODE/pss/server/test/SimpleRead.java, 123
src/SERVER_CODE/pss/server/test/SimpleWrite.java, 126
src/SERVER_CODE/pss/server/Utils.java, 128
stationary
pss::server::Bot, 10
status
pss::server::Bot, 10
stop
bot_motion.h, 66
STRAIGHT
pss::server::Graph, 32
Straight
pss::server::Graph, 29
straight_message
pss::server::Utils, 50

temp
lcd.h, 84
tens
lcd.h, 84
TEST_PORT1
pss::configuration::Configure, 20
TEST_PORT2
pss::configuration::Configure, 20
thousand
lcd.h, 84
timer5_init
bot_motion.h, 67
TURN_AROUND
bot.c, 52
bot_2.c, 57
TURN_LEFT
bot.c, 52
bot_2.c, 57
turn_left
bot_motion.h, 67
TURN_RIGHT
bot.c, 52
bot_2.c, 58
turn_right
bot_motion.h, 67
turnBack
pss::server::Bot, 8
turnLeft
pss::server::Bot, 8
turnRight
pss::server::Bot, 8

unit
lcd.h, 84
updateRequestStatus
pss::server::database::DBHandler, 25
USART_Init
bot.c, 53
bot_2.c, 58
IR.c, 77

velocity
bot_motion.h, 67

W_THRESHOLD
bot_motion.h, 64
W_THRESHOLD_STOP
bot_motion.h, 64
WATER
pss::server::RequestHandler, 44
WEST
pss::server::Graph, 32

ZIGBEE_PORT
pss::configuration::Configure, 20