## Contents

1 BotCon Package List .......................................................... 1
   1.1 BotCon Package List .................................................... 1

2 BotCon Hierarchical Index .................................................. 3
   2.1 BotCon Class Hierarchy ................................................. 3

3 BotCon Compound Index ..................................................... 5
   3.1 BotCon Compound List ................................................ 5

4 BotCon Page Index ............................................................ 7
   4.1 BotCon Related Pages ................................................ 7

5 BotCon Package Documentation .............................................. 9
   5.1 Package net.elektronengehirn.botcon ............................... 9

6 BotCon Class Documentation ................................................ 11
   6.1 Comm Class Reference ............................................... 11
   6.2 Hemisson Class Reference .......................................... 13
   6.3 HemLinCam Class Reference ........................................ 19

7 BotCon Page Documentation ................................................ 23
   7.1 Todo List ..................................................................... 23
Chapter 1

BotCon Package List

1.1 BotCon Package List

Here are the packages with brief descriptions (if available):

\texttt{net.elektronengehirn.botcon} ........................................... 9
Chapter 2

BotCon Hierarchical Index

2.1 BotCon Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Comm .................................................. 11
Thread
  Hemisson ........................................... 13
  HemLinCam ......................................... 19
Chapter 3

BotCon Compound Index

3.1 BotCon Compound List

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

- **Comm** (Class Comm – asynchronous communication over serial line) 11
- **Hemisson** (Class Hemisson – API to control Hemisson over serial line) 13
- **HemLinCam** (Class HemLinCam – API to control HemLinCam over serial line) 19
Chapter 4

BotCon Page Index

4.1 BotCon Related Pages

Here is a list of all related documentation pages:

   Todo List ................................................................. 23
Chapter 5

BotCon Package
Documentation

5.1 Package net.elektronengehirn.botcon

Classes

- class Comm
  
  *Class* Comm – asynchronous communication over serial line.

- class Hemisson
  
  *Class* Hemisson – API to control Hemisson over serial line.

- class HemLinCam
  
  *Class* HemLinCam – API to control HemLinCam over serial line.
Chapter 6

BotCon Class Documentation

6.1 Comm Class Reference

Class Comm – asynchronous communication over serial line.

Public Methods

• **Comm** (String portName, int baudRate)
  
  *Open serial port and initialize stream-reader and -writer.*

• synchronized String **sendMessage** (String msg)
  
  *Sends an ASCII message over serial line.*

6.1.1 Detailed Description

Class Comm – asynchronous communication over serial line.

**Note:**

Please note, that you need the Java Communication API to use this interface.

You can download Sun’s Windows and Solaris implementation at: 
http://java.sun.com/products/javacomm/

If you are using Linux you should try: http://www.rxtx.org/

**Todo:**

  Timeout for receiving response.
6.1.2 Constructor & Destructor Documentation

6.1.2.1 Comm (String portName, int baudRate) [inline]
Open serial port and initialize stream-reader and -writer.

Parameters:
portName Serial port identifier.
baudRate Baudrate for serial link.

6.1.3 Member Function Documentation

6.1.3.1 synchronized String sendMessage (String msg) [inline]
Sends an ASCII message over serial line.

Parameters:
msg the ASCII text-message.

Returns:
String response message.

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

• Comm.java
6.2 Hemisson Class Reference

Class Hemisson – API to control Hemisson over serial line.

Inheritance diagram for Hemisson:

```
Thread

Hemisson
```

Collaboration diagram for Hemisson:

```
Thread Comm

/ \ / \\
Comm / \\

Hemisson
```

Public Methods

- **Hemisson ()**
  
  *Open serial port and connect to robot.*

- **Hemisson (String serialport, int baudrate)**
  
  *Open serial port and connect to robot.*

- **Hemisson (String serialport, int baudrate, long interval)**
  
  *Open serial port and connect to robot.*

- **int getBrightness (int sensor)**
  
  *Get value for brightness from infrared sensor.*

- **int getProximity (int sensor)**
  
  *Get value for proximity from infrared sensor.*

- **void setSpeed (int left, int right)**
  
  *Set values for differential steering system.*

- **void beep (int state)**
  
  *Set state for the robot’s buzzer.*

- **int getSwitchState (int switch_number)**
Get the state of one of the four DIP-switches.

- void `delay` (long ms)

  Interrupt task execution for a few milliseconds.

- void `run` ()

  Refresh sensor data and motor control in the background every 100 ms by default.

**Static Public Attributes**

- final int `FRONT` = 2
- final int `FRONTLEFT` = 1
- final int `FRONTRIGHT` = 0
- final int `LEFT` = 4
- final int `RIGHT` = 5
- final int `REAR` = 3
- final int `GROUNDLEFT` = 6
- final int `GROUNDRIGHT` = 7

**Protected Methods**

- void `finalize` ()

  Stop the robot on destruction of the instance of Hemisson.

**Protected Attributes**

- `Comm comm`

### 6.2.1 Detailed Description

Class Hemisson – API to control Hemisson over serial line.

This class implements the commands for communication with a Hemisson robot over serial line.

It uses the same ASCII text-protocol as described in the Hemisson manual. For programming java controllers, there are member functions similar to the HemiOs C functions like `get_proximity(FRONT)`, `set_speed(left, right)` etc.

The class `Comm` (p.11) is used for serial transmission. The `sendMessage` function in `Comm` (p.11) is called within the synchronized private functions to get sensor data and send motor commands, e.g. `get_brightness()`. The same or a comparable API can be used to communicate over other hardware or even with a simulated Hemisson robot.
Communication starts immediately after initializing an instance of Hemisson by switching the thread in its run-state. Therefore your Hemisson must be activated in default-mode (serial communication) before you start your java controller! The sensor data is refreshed and motor commands are sent resp. every 100 ms (10 Hz) by default. This is the same refresh-rate as used by the robot’s low level sensor/motor control system. So it doesn’t make any sense to read sensory information faster than 10 Hz—you should clock your robot controller with the same frequency or even slower.

You can use another constructor to change your serial port, baudrate or the refresh-rate. The default port is "/dev/ttyS0", the default baudrate is 115200 (should not be changed, due to fixed baudrate in the robot firmware).

To get an approximation of the proximity (in mm) from the proximity-value \( x \) use this logarithm function:

\[
f(x) = \log_{0.906475} \frac{x}{255}
\]

The inverse function would be useful for robot simulation (\( x \) is the calculated proximity in mm):

\[
f(x) = 255 \cdot 0.906475^x
\]

The following linear functions could be used to calculate the speed for a wheel in mm/s, i.e. for pseudo-odometric data (\( x \) is the int-value for the left or right motor):

\[
f(x) = 21.3939 \cdot x - 16.6727
\]

And in case of negative int-values for motor control:

\[
f(x) = 21.3939 \cdot x + 16.6727
\]

These formulas were acquired by measurements on a real Hemisson robot moving within a maze of white styrofoam.

**Note:**
Sun’s Windows implementation of the Java **Comm** (p. 11) API is very slow and therefore the real refresh-rate can be slower than 10 Hz.

**Version:**
0.1

**Author:**
Ingo Frank <if@elektronengehirn.net>
6.2.2 Constructor & Destructor Documentation

6.2.2.1 Hemisson () [inline]
Open serial port and connect to robot.
The default constructor opens port "COM1" with 115200 baud.

6.2.2.2 Hemisson (String serialport, int baudrate) [inline]
Open serial port and connect to robot.
Set your port identifier and baudrate.

Parameters:
serialport Serial port identifier.
baudrate Baudrate for serial link.

6.2.2.3 Hemisson (String serialport, int baudrate, long interval)
[inline]
Open serial port and connect to robot.
Set your port identifier, baudrate and refresh-rate.

Parameters:
serialport Serial port identifier.
baudrate Baudrate for serial link.
interval Refresh-rate (in ms).

6.2.3 Member Function Documentation

6.2.3.1 void beep (int state) [inline]
Set state for the robot’s buzzer.
State can be 0 (off) or 1 (on).

Note:
The buzzer generates a continuous sound—or silence.

Parameters:
state state for buzzer (0 or 1).
6.2.3.2 void delay (long \textit{ms})  [inline]

Interrupt task execution for a few milliseconds.
Wait is done by calling the static method Thread.sleep(). This will delay the
calling thread for the given time.

\textbf{Parameters:}

\begin{itemize}
  \item \textit{ms} time to wait in milliseconds.
\end{itemize}

6.2.3.3 int getBrightness (int \textit{sensor})  [inline]

Get value for brightness from infrared sensor.

\textbf{Parameters:}

\begin{itemize}
  \item \textit{sensor} the number of infrared sensor (FRONT, FRONTLEFT etc.).
\end{itemize}

\textbf{Returns:}

\begin{itemize}
  \item value 0–255 (very bright – very dark).
\end{itemize}

6.2.3.4 int getProximity (int \textit{sensor})  [inline]

Get value for proximity from infrared sensor.

\textbf{Parameters:}

\begin{itemize}
  \item \textit{sensor} the number of infrared sensor (FRONT, FRONTLEFT etc.).
\end{itemize}

\textbf{Returns:}

\begin{itemize}
  \item value 0–255 (very far – very close).
\end{itemize}

6.2.3.5 int getSwitchState (int \textit{switch\_number})  [inline]

Get the state of one of the four DIP-switches.

\textbf{Parameters:}

\begin{itemize}
  \item \textit{switch\_number} the number of switch.
\end{itemize}

\textbf{Returns:}

\begin{itemize}
  \item value 0 or 1 (right – left).
\end{itemize}

6.2.3.6 void setSpeed (int \textit{left}, int \textit{right})  [inline]

Set values for differential steering system.
Speed can be set from -9 to 9 for both motors: 0,0 will stop the robot, -1,1 will
start a slow left-rotation and 9,9 will move it fast forward.
Speed settings between -3 and 3 are good choices (not too fast).
Note:
The last received motor command is continuously executed.

Parameters:
- \texttt{left}  speed for left motor.
- \texttt{right} speed for right motor.

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

- Hemisson.java
6.3 HemLinCam Class Reference

Class HemLinCam – API to control HemLinCam over serial line.

Inheritance diagram for HemLinCam:

 Collaboration diagram for HemLinCam:

Public Methods

- **HemLinCam** (Hemisson hemisson)
  
  *Initialize HemLinCam.*

- **HemLinCam** (Hemisson hemisson, long interval)
  
  *Initialize HemLinCam.*

- void **setThreshold** (int value)
  
  *Set threshold.*

- void **setExposure** (int time)
  
  *Set exposure time.*

- int [] **getPixels** ()
  
  *Get 255 grey-values from linear camera.*

- void **run** ()
  
  *Transmit camera settings and get pixels/image in the background every 100 ms by default.*
6.3.1 Detailed Description

Class HemLinCam – API to control HemLinCam over serial line.

This class implements the commands for communication with the HemLinCam module for the Hemisson (p. 13) robot over serial line.

Version:

0.1

Author:

Ingo Frank <if@elektronengehirn.net>

6.3.2 Constructor & Destructor Documentation

6.3.2.1 HemLinCam (Hemisson hemisson) [inline]

Initialize HemLinCam.

Plug it to existing instance of Hemisson (p. 13).

Parameters:

hemisson Existing instance of Hemisson (p. 13).

6.3.2.2 HemLinCam (Hemisson hemisson, long interval) [inline]

Initialize HemLinCam.

Plug it to existing instance of Hemisson (p. 13) and set refresh-rate.

Parameters:

hemisson Existing instance of Hemisson (p. 13).
interval Refresh-rate (in ms).

6.3.3 Member Function Documentation

6.3.3.1 int[] getPixels () [inline]

Get 255 grey-values from linear camera.

Returns:

Array with 255 values 0–255 (black – white).

6.3.3.2 void setExposure (int time) [inline]

Set exposure time.

Parameters:

time Exposure time 1–10.
6.3.3.3 void setThreshold (int value) [inline]

Set threshold.

Parameters:

value Threshold 0–255.

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

- HemLinCam.java
Chapter 7

BotCon Page
Documentation

7.1 Todo List

Class Comm  Timeout for receiving response.