A Human-Robot Interaction Dance Demo – a Short Technical Report

Pérez-Xochicale Miguel Angel
INAOE's Robotics Laboratory
perez.xochicale@gmail.com

July 2013

Abstract

This work presents a Human-Robot Interation (HRI) Dance Demo, which is based on a ZSTAR3 Radio Frequency single three-axis accelerometer and a Patrolbot mobile robot. The ZSTAR3 sensor board is located in the position of the user's left wrist. With the wereable accelerometer, we explore four movements of the arm in order to control the mobile robot. We experiment with different gesture patterns for discerning a waltz dance perfomance. This short technical report involves an explanation of the HRI connection, the HRI application, the hand gesture control algorithm, and the appendix for the class documentation.

Contents

Chapter 1

Introduction

Human-Robot Interaction (HRI) has received much attention in recent years due to its ubiquitous computing, which is based on a new family of devices bywhich their information is thoroughly integrated into everyday objects and activities. These devices are sensors for measuring different physical phenomenons. Thus, it is possible to create interfaces that merge human-wearable devices with a mobile robot, promoting human-robot interactions. However, the majority of regular tests at the Robocub@home category are aimed at home activities such as: follow me, cocktail party, clean up, and emergency fire situation in the apartment [3]; and, therefore, little attention has been paid to the entertainment activities such as dancing.

To address this concern, a human-robot interaction dance demo was developed in which a single-accelerometer is worn on the user's left wrist in order to detect four different gestures to move the robot by using statistical theory. The video `http://www.youtube.com/watch?v=Kw-lZam_qZI` shows the dance demo at the 2013 Mexican Tournament of Robotics.

This technical report describes the connection of the ZSTAR3 sensor board accelerometer with the pioneerbot robot for gesture recognition. Next, the HRI Application used is described. Methods for hand gesture recognition based on acceleration signals using statistical theory are described first; and a simple control algorithm strategy for the dance demo performance is provided. Lastly, discussion and suggestions for future work are given together with conclusions.

Chapter 2

Methodology

The current technical report involves an explanation of the HRI connection, the HRI application, and the hand gesture control algorithm.

2.1 HRI Setup

In order to use the HRI application, a scheme setup is shown in Figure 2.1, and the following steps are provided as an explanation of HRI connection:

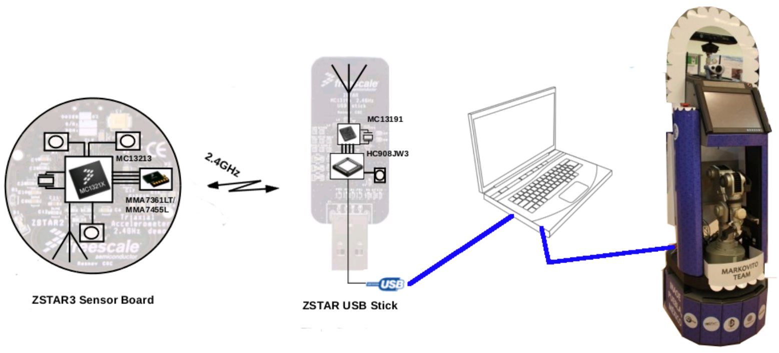


Figure 2.1: Human-Robot Interface Dance Demo Scheme Connection

1. Turn on the Pioneer Mobile Robot
2. Connect the Aria USB Connector and the ZSTAR USB Stick to the available laptop's USB plugs.

3. The user must strictly wear the USB sensor board on his/her left wrist as shown in Figure 2.2, and it is important to emphasize that similar sensor position is crucial to obtain same results.

4. Finally, turn on the sensor board by pushing any of the three white pushbuttons.

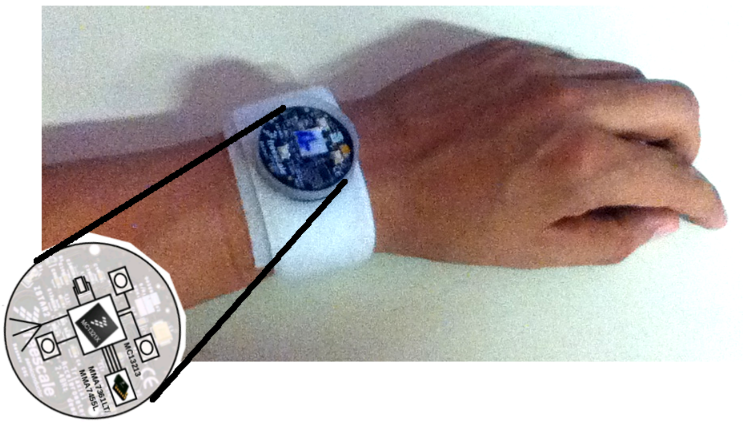


Figure 2.2: Wearable Sensor Board.

For more detailed information about the Wireless Sensing Triple Axis Reference (ZSTAR3) and the patrol mobile robot, please visit the references [1, 2].

2.2 HRI Application

You should first download the source tarball from hridancedemo-0.0.1.tar.gz , then uncompress it using: `$tar xzvfp hridancedemo-0.0.1.tar.gz`.

After uncompressing, a new directory in `~/(hridancedemo-0.0.1)` will have been created. Go into `~/dancedemoapp/`directory and change the executable permissions and finally run the application by using the following commands lines:

```
$ cd ~/dancedemoapp/

$ sudo chmod +x usbresetapp dancedemoapp

$ sudo ./dancedemoapp
```

Once the application is running, the accelermoter data is displayed at the terminal, and whenever you want to stop the application you should hit any key.

2.3 Hand Gesture Recognition Algorithm

While running the application, the user is able to move his/her hand in four different gestures in order to move the robot as is illustrated below in Figure 2.3.

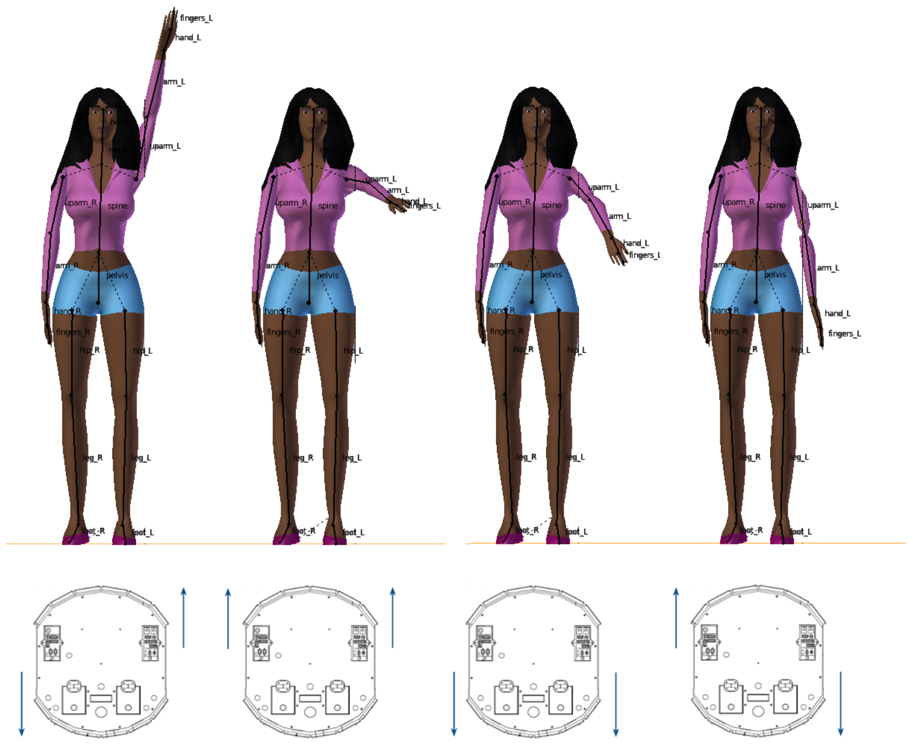


Figure 2.3: User's Left Hand Gestures with the robot's movements arrow directions.

Hence, having read the X, Y, Z accelerations from the sensor board, the arithmetic mean and the variance is computed by using the Equations 2.1.

$$\mu = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i, \quad \mathrm{Var}(X) = \frac{1}{n} \sum_{i=1}^{n} (x_i - \mu)^2. \tag{2.1}$$

During the demo, the acceleration data of the accelerometer is collected and represented as a frame of 5 samples in order to obtain the mean and standard deviation. As a result of that, a control algorithm was implemented and it is described in (19). The complete C++ source code is abailable in the Appendix A.

Algorithm 1 An example for format For & While Loop in Algorithm

```
while !kbhit() do
    Accelerometer Adquisition Data
    if (iteration mod FRAMEINTERVAL) == 0 then
        compute the mean and standard deviation
        if ($\mu_X$ < 1.1) and $\mu_X$ > 0.5 then
            Robot Spin to the left at 150 mm per second
        end if
        if $\mu_X$ < 0.5 and $\mu_X$ > $-0.3$ then
            if $\mu_Y$ > 0.5 then
                Robot move fordward at 120 mm per second
            else
                Robot move backward at 120 mm per second
            end if
        end if
        if $\mu_X$ < $-0.3$ and $\mu_X$ > $-1.1$ then
            Robot Spin to the right at 150 mm per second
        end if
    end if
end while
```

Chapter 3

Conclusions

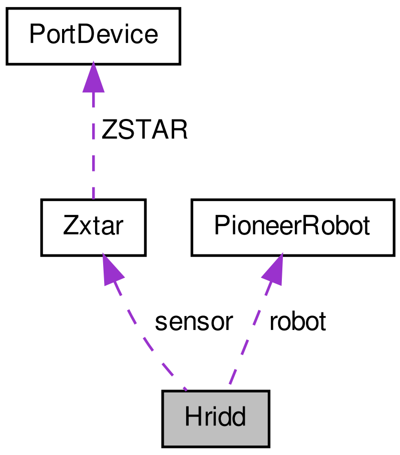
We have presented an HRI Dance Demo which is tailored to deal with the challenges of a real-time Human-Robot Interaction entertainment application. It was presented satisfactorily at the 2013 Mexican Tournament of Robotics. One future direction motivated by this work is to add four wireless sensors in order to distinguish different dance patterns, and, then, the robot will have been capable of suggesting a musical genre according the user's movements.

Appendix A

Class Documentation

A.1 Hridd Class Reference

```
#include <Hridd.h>
```



Public Member Functions

- int **Kbhit** (void)
- void **GestureControl** ()

- void **SetACCFrameWord** (unsigned int framevalue)

Public Attributes

- **PioneerRobot robot**
- **Zxtar sensor**

Detailed Description

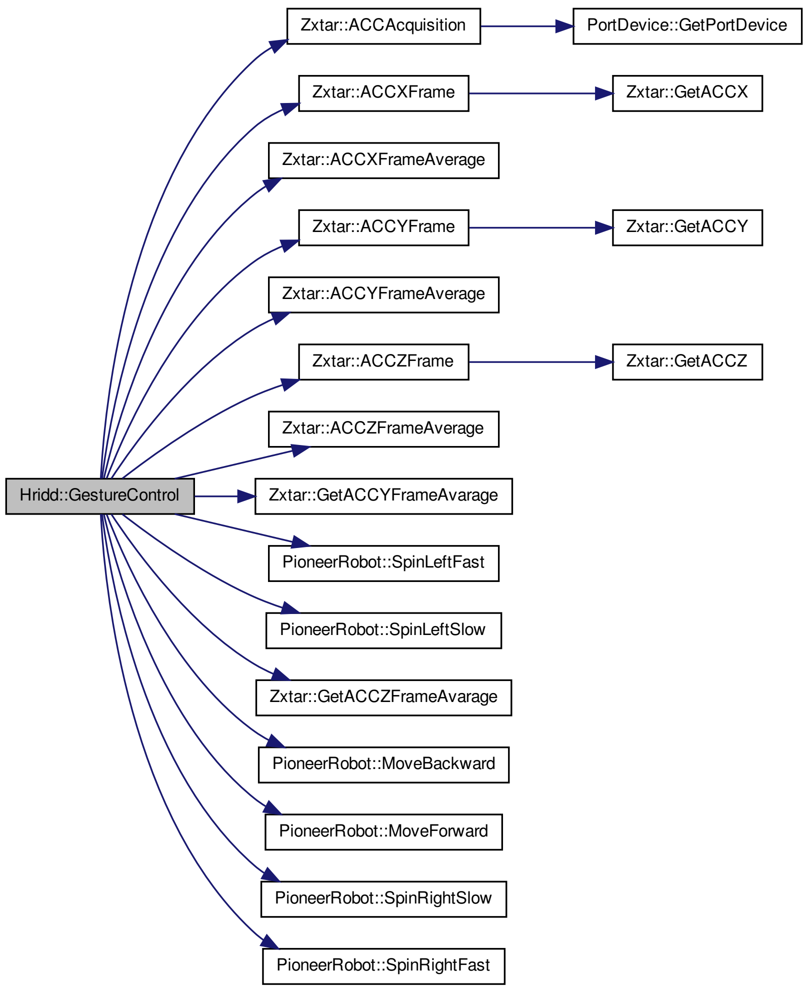
Human-Robot Interaction Dance Demo class is aimed to combine the Zxtar class accelerometer class, the PioneerRobot class so as to control the robot with the user's wearable accelerometer. Definition at line 16 of file Hridd.h.

Member Function Documentation

void Hridd::GestureControl) This method contains the main control gesture for the robot movements

Definition at line 40 of file Hridd.cpp.

Here is the call graph for this function:


Zxtar::ACCAcquisition
PortDevice::GetPortDevice
Zxtar::ACCXFrame
Zxtar::GetACCX
Zxtar::ACCXFrameAverage
Zxtar::ACCYFrame
Zxtar::GetACCY
Zxtar::ACCYFrameAverage
Zxtar::ACCZFrame
Zxtar::GetACCZ
Zxtar::ACCZFrameAverage
Hridd::GestureControl
Zxtar::GetACCYFrameAvarage
PioneerRobot::SpinLeftFast
PioneerRobot::SpinLeftSlow
Zxtar::GetACCZFrameAvarage
PioneerRobot::MoveBackward
PioneerRobot::MoveForward
PioneerRobot::SpinRightSlow
PioneerRobot::SpinRightFast

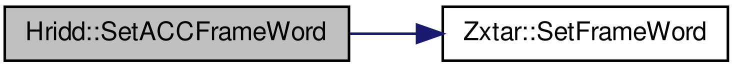
int Hridd::Kbhit ( ) Method for the Keyboard hit

Definition at line 8 of file Hridd.cpp.

void Hridd::SetACCFrameWord ( ) Function to set the Acceleration Frame Word

Definition at line 34 of file Hridd.cpp.

Here is the call graph for this function:



Member Data Documentation

PioneerRobot Hridd::robot Data Member for the PionerRobot Class which defines and initializes the robot

Definition at line 23 of file Hridd.h.

Zxtar Hridd::sensor Data Member for the PionerRobot Class which defines and initializes the accelerometer sensor

Definition at line 28 of file Hridd.h.

A.2 PioneerRobot Class Reference

`#include <PioneerRobot.h>`

Public Member Functions

- **PioneerRobot** ()
- ~**PioneerRobot** ()
- virtual bool **init** ()
- virtual void **getCurrentConfiguration** (std::vector< double > &q)
- virtual bool **moveToConfiguration** (std::vector< double > configuration)
- void **MoveForward** ()
- void **MoveBackward** ()
- void **SpinLeftSlow** ()

- void **SpinRightSlow** ()
- void **SpinLeftFast** ()
- void **SpinRightFast** ()

Public Attributes

- ArRobot **pioneer**

Protected Member Functions

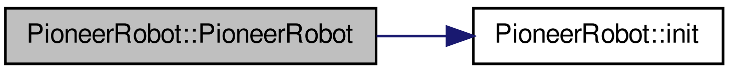
- double **validateAngle** (double angle)

Detailed Description

Definition at line 13 of file PioneerRobot.h.

Constructor & Destructor Documentation

PioneerRobot::PioneerRobot ( ) Definition at line 3 of file PioneerRobot.cpp.
Here is the call graph for this function:



PioneerRobot::~PioneerRobot ( ) Definition at line 9 of file PioneerRobot.-cpp.

Member Function Documentation

void PioneerRobot::getCurrentConfiguration (

std::vector< double > & q

) **`[virtual]`** Definition at line 76 of file PioneerRobot.cpp.

bool PioneerRobot::init ( ) `[virtual]` Definition at line 15 of file PioneerRobot.cpp.

void PioneerRobot::MoveBackward ( ) Definition at line 144 of file PioneerRobot.cpp.

void PioneerRobot::MoveForward ( ) Definition at line 149 of file PioneerRobot.cpp.

bool PioneerRobot::moveToConfiguration (

std::vector< double > configuration

) `[virtual]` Definition at line 91 of file PioneerRobot.cpp.

void PioneerRobot::SpinLeftFast ( ) Definition at line 164 of file PioneerRobot.cpp.

void PioneerRobot::SpinLeftSlow ( ) Definition at line 154 of file PioneerRobot.cpp.

void PioneerRobot::SpinRightFast ( ) Definition at line 169 of file PioneerRobot.cpp.

void PioneerRobot::SpinRightSlow ( ) Definition at line 159 of file PioneerRobot.cpp.

double PioneerRobot::validateAngle (

double angle

) `[protected]` Definition at line 132 of file PioneerRobot.cpp.

Member Data Documentation

ArRobot PioneerRobot::pioneer Definition at line 31 of file PioneerRobot.h.

A.3 PortDevice Class Reference

`#include <PortDevice.h>`

Public Member Functions

- **PortDevice** ()
- int **AllocatePortDevice** ()
- int **ConfigPortDevice** ()
- int **GetPortDevice** ()
- void **CleanPortDevice** ()
- void **ClosePortDevice** ()

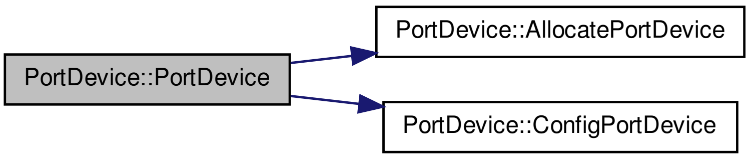
Detailed Description

Definition at line 27 of file PortDevice.h.

Constructor & Destructor Documentation

PortDevice::PortDevice ( ) Constructor to reset, to allocate and configurate the ZSTAR USB Stick

Definition at line 4 of file PortDevice.cpp.
Here is the call graph for this function:



Member Function Documentation

int PortDevice::AllocatePortDevice ( ) SetPortDeviceFD
Definition at line 13 of file PortDevice.cpp.

void PortDevice::CleanPortDevice ( ) function to clean the Port Device
Definition at line 89 of file PortDevice.cpp.

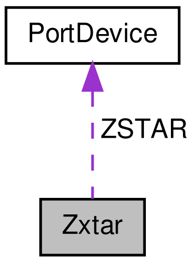
void PortDevice::ClosePortDevice ( ) ClosePortDevice
Definition at line 83 of file PortDevice.cpp.

int PortDevice::ConfigPortDevice ( ) ConfigPortDeviceFD
Definition at line 28 of file PortDevice.cpp.

int PortDevice::GetPortDevice ( ) function to get the port's FILE DESCRIPT-OR
Definition at line 78 of file PortDevice.cpp.

A.4 Zxtar Class Reference

`#include <Zxtar.h>`
Collaboration diagram for Zxtar:



Public Member Functions

- **Zxtar** ()
- **Zxtar** (int gSensitivity, int setRate)
- void **AvailableACC** ()
- int **Kbhit** (void)
- void **SetSensitivity** (int gSensitivity)

- void **DataRate** (int setRate)
- void **ACCAcquisition** ()
- double **GetACCX** ()
- double **GetACCY** ()
- double **GetACCZ** ()
- void **ACCPrint** ()
- void **startTimer** ()
- double **stopTimer** ()
- void **SetFrameWord** (unsigned int framevalue)
- void **ACCXFrame** ()
- void **ACCXFramePrint** ()
- double **ACCXFrameAverage** ()
- double **GetACCXFrameAvarage** ()
- double **ACCXFrameStdDeviation** ()
- double **GetACCXFrameStdDeviation** ()
- void **ACCYFrame** ()
- void **ACCYFramePrint** ()
- double **ACCYFrameAverage** ()
- double **GetACCYFrameAvarage** ()
- double **ACCYFrameStdDeviation** ()
- double **GetACCYFrameStdDeviation** ()
- void **ACCZFrame** ()
- void **ACCZFramePrint** ()
- double **ACCZFrameAverage** ()
- double **GetACCZFrameAvarage** ()

- double **ACCZFrameStdDeviation** ()
- double **GetACCZFrameStdDeviation** ()

Static Public Member Functions

- static void **printTime** (double duration)

Public Attributes

- **PortDevice ZSTAR**

Detailed Description

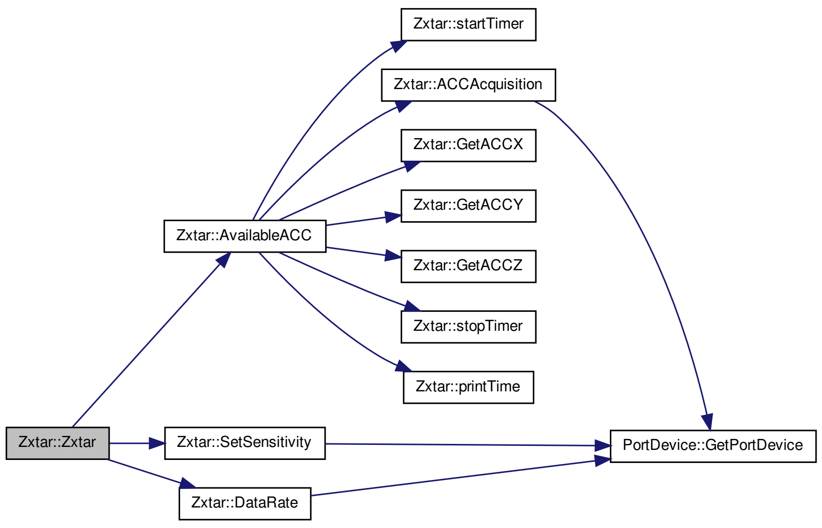
Zxtar class permitis to extract a frame, and compute the avarage and Standar - Deviation, as well to get these values for the X, Y and Z axis. The following code is an example for its use; uncomment the code for further test.

Zxtar sensor; // Declaration of the sensor Zstar class type
sensor.ACCXFrame(); // Getting the ACC of the X-axis frame word
sensor.ACCXFrameAverage(); // X-axis ACC frame mean
sensor.ACCXFrameStdDeviation(); // X-axis ACC frame Standard Deviation
cout << sensor.GetACCXFrameAvarage() << endl; //Printing the ACCX value of the frame avarage
cout << sensor.GetACCXFrameStdDeviation() << endl; //Printing the ACCX value of the frame standard deviation
Definition at line 47 of file Zxtar.h.

Constructor & Destructor Documentation

Zxtar::Zxtar ( ) Default Constructor with the default values for sensitibility and data rate: 2g and 60Hz
Definition at line 4 of file Zxtar.cpp.

Here is the call graph for this function:



Zxtar::Zxtar (

int gSensitivity,

int setRate

) Constructor with parameter options, so as the user is able to configure the sensor by using the following values

Zxtar name(g-range, DataRate);

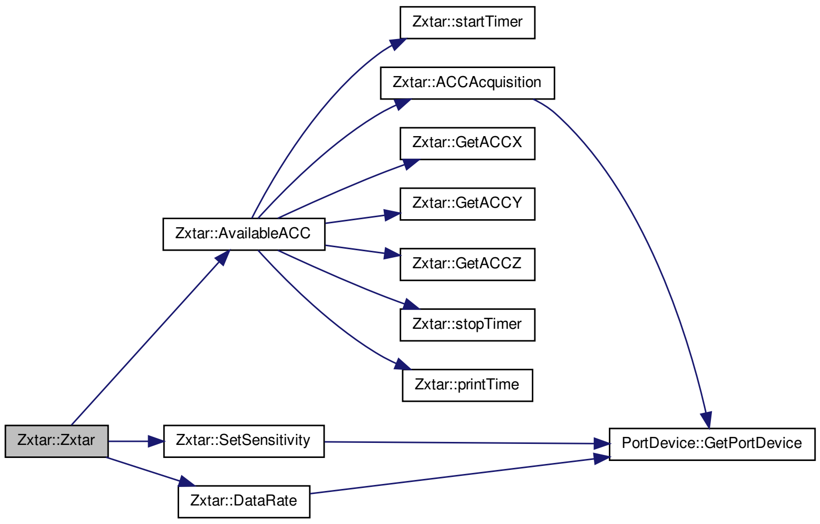
.............|........|

.............|........|--------- Available Values for DataRate are 30, 60, and 120

.............|------------------- Available Values for g-range parameter are 2, 4, and 8.

Definition at line 21 of file Zxtar.cpp.

Here is the call graph for this function:

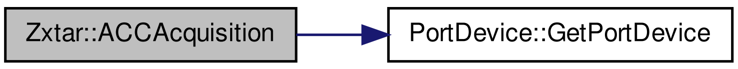


Member Function Documentation

void Zxtar::ACCAcquisition ( ) Method for the Acceleration Acquisition

Definition at line 149 of file Zxtar.cpp.

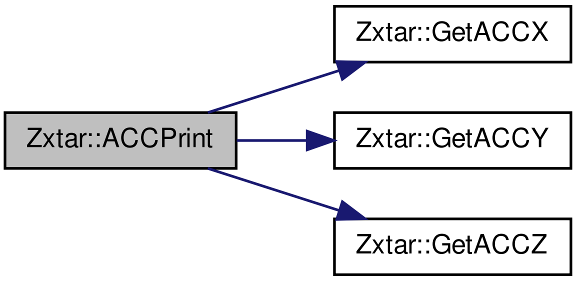
Here is the call graph for this function:



void Zxtar::ACCPrint ( ) Method to print the Acceleartion Values

Definition at line 186 of file Zxtar.cpp.

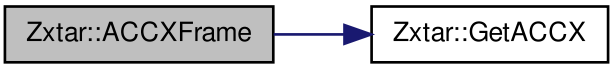
Here is the call graph for this function:



void Zxtar::ACCXFrame ( ) Method for the adquisition of X-axis data frame

Definition at line 234 of file Zxtar.cpp.

Here is the call graph for this function:



double Zxtar::ACCXFrameAverage ( ) Method to compute the X-axis data frame avarage

Definition at line 261 of file Zxtar.cpp.

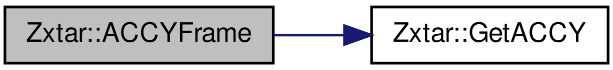
void Zxtar::ACCXFramePrint ( ) Method for printing the X-axis data frame

Definition at line 248 of file Zxtar.cpp.

double Zxtar::ACCXFrameStdDeviation ( ) Method to compute the X-axis data frame standard deviation

Definition at line 282 of file Zxtar.cpp.

void Zxtar::ACCYFrame ( ) Method for the adquisition of Y-axis data frame
Definition at line 304 of file Zxtar.cpp.
Here is the call graph for this function:

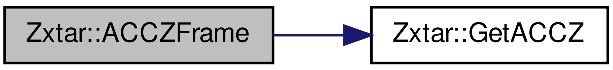


double Zxtar::ACCYFrameAverage ( ) Method to compute the Y-axis data frame avarage
Definition at line 331 of file Zxtar.cpp.

void Zxtar::ACCYFramePrint ( ) Method for printing the Y-axis data frame
Definition at line 318 of file Zxtar.cpp.

double Zxtar::ACCYFrameStdDeviation ( ) Method to compute the Y-axis data frame standard deviation
Definition at line 351 of file Zxtar.cpp.

void Zxtar::ACCZFrame ( ) Method for the adquisition of Z-axis data frame
Definition at line 376 of file Zxtar.cpp.
Here is the call graph for this function:



double Zxtar::ACCZFrameAverage ( ) Method to compute the Z-axis data frame avarage
Definition at line 403 of file Zxtar.cpp.

void Zxtar::ACCZFramePrint ( ) Method for printing the Z-axis data frame

Definition at line 390 of file Zxtar.cpp.

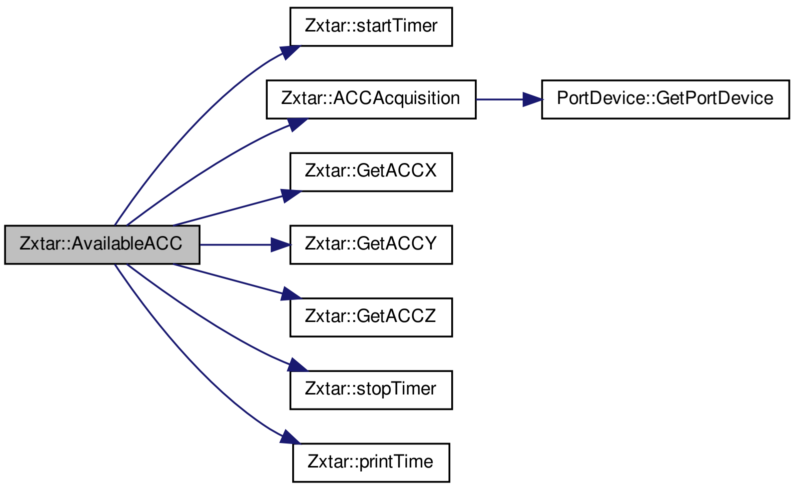
double Zxtar::ACCZFrameStdDeviation ( ) Method to compute the Z-axis data frame standard deviation

Definition at line 425 of file Zxtar.cpp.

void Zxtar::AvailableACC ( ) To wait until the data is available by using a while loop

Definition at line 51 of file Zxtar.cpp.

Here is the call graph for this function:



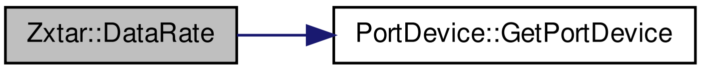
void Zxtar::DataRate (

int setRate

) Method to set the Data Rate

Definition at line 124 of file Zxtar.cpp.

Here is the call graph for this function:



double Zxtar::GetACCX ( ) Method to get X-axis Acceleration
Definition at line 171 of file Zxtar.cpp.

double Zxtar::GetACCXFrameAvarage ( ) Method to get the X-axis data frame avarage
Definition at line 276 of file Zxtar.cpp.

double Zxtar::GetACCXFrameStdDeviation ( ) Method to get the X-axis data frame standard deviation
Definition at line 298 of file Zxtar.cpp.

double Zxtar::GetACCY ( ) Method to get Y-axis Acceleration
Definition at line 176 of file Zxtar.cpp.

double Zxtar::GetACCYFrameAvarage ( ) Method to get the Y-axis data frame avarage
Definition at line 346 of file Zxtar.cpp.

double Zxtar::GetACCYFrameStdDeviation ( ) Method to get the Y-axis data frame standard deviation
Definition at line 367 of file Zxtar.cpp.

double Zxtar::GetACCZ ( ) Method to get Z-axis Acceleration
Definition at line 181 of file Zxtar.cpp.

double Zxtar::GetACCZFrameAvarage ( ) Method to get the Z-axis data frame avarage
Definition at line 418 of file Zxtar.cpp.

double Zxtar::GetACCZFrameStdDeviation ( ) Method to get the Z-axis data frame standard deviation

Definition at line 445 of file Zxtar.cpp.

int Zxtar::Kbhit (

void

) Method for the keyboard hit

Definition at line 70 of file Zxtar.cpp.

void Zxtar::printTime (

double duration

) `[static]` Method to print the timer

Definition at line 218 of file Zxtar.cpp.

void Zxtar::SetFrameWord (

unsigned int framevalue

) Method to set the frame word of the acceleration readings

Definition at line 226 of file Zxtar.cpp.

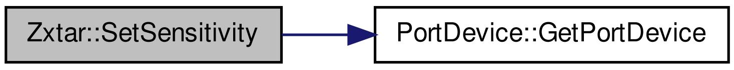
void Zxtar::SetSensitivity (

int gSensitivity

) Method to select the g-range for the accelerometer Sensor Board BY SENDI-NG A g0 g1 or g2 you are able to change the SENSITIVITY g0 = 8g ; g1 = 2g ; g2 = 4g

Definition at line 96 of file Zxtar.cpp.

Here is the call graph for this function:



void Zxtar::startTimer ( ) Method to start the timer
Definition at line 197 of file Zxtar.cpp.

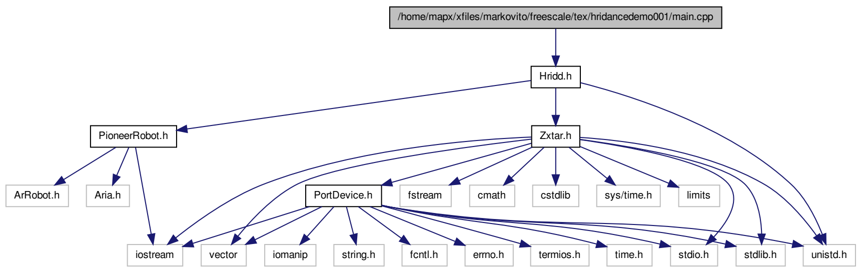
double Zxtar::stopTimer ( ) Method to stop the timer
Definition at line 202 of file Zxtar.cpp.

Member Data Documentation

PortDevice Zxtar::ZSTAR Data Member for the use fo the PortDevice class with the sensor board
Definition at line 88 of file Zxtar.h.

A.5 main.cpp File Reference

`#include "Hridd.h"` Include dependency graph for main.cpp:



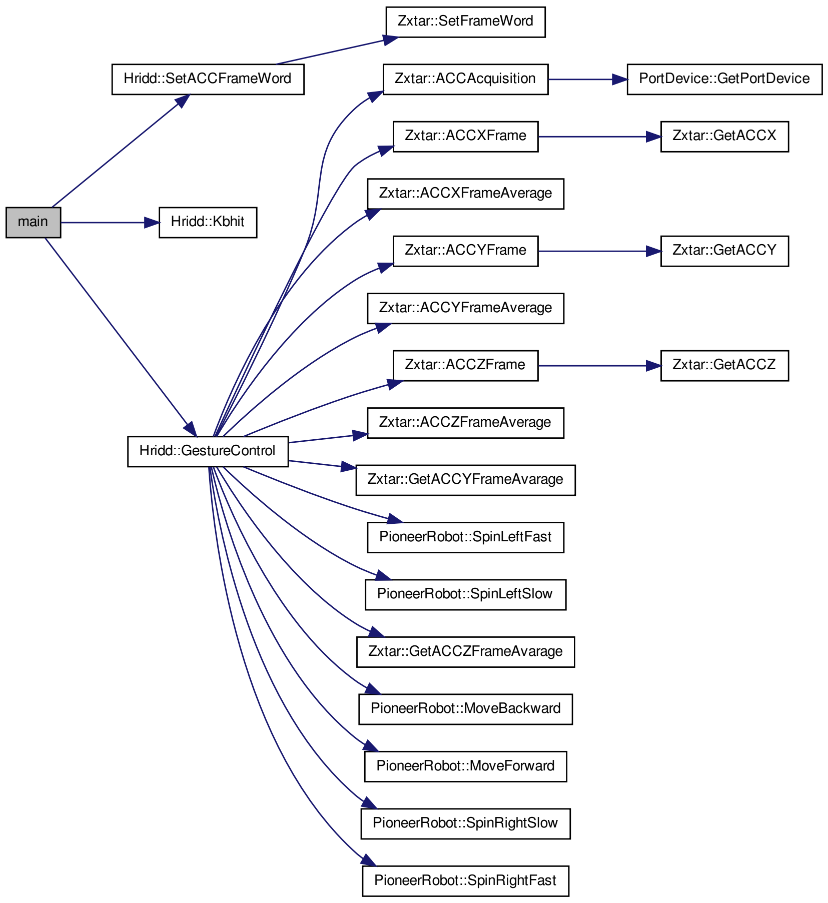
Functions

- int **main** ()

Function Documentation

int main ( ) Definition at line 3 of file main.cpp.

Here is the call graph for this function:



Bibliography

[1] Drm103 designer reference manual. http://www.freescale.com/files/microcontrollers/doc/ref_manual/ZSTAR3RM.pdf. Accessed: 2013-06-24.

[2] Research patrolbot. http://www.mobilerobots.com/ResearchRobots/ResearchPatrolBot.aspx. Accessed: 2013-06-15.

[3] Robocup@home rulebook (revision 411, 2013-02-17). http://ccc.inaoep.mx/~tmr2013/pdf/robocupathome2013.pdf, 2013. Accessed: 2013-04-24.