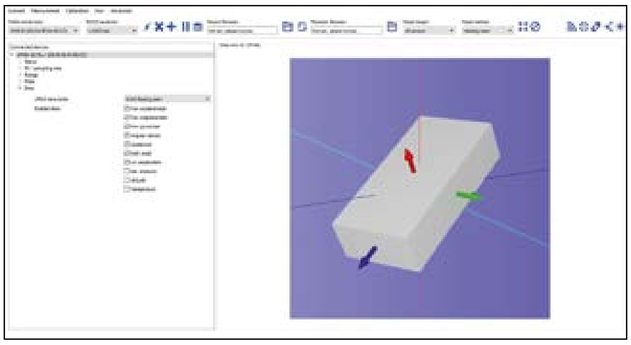
Software LPMS-Control

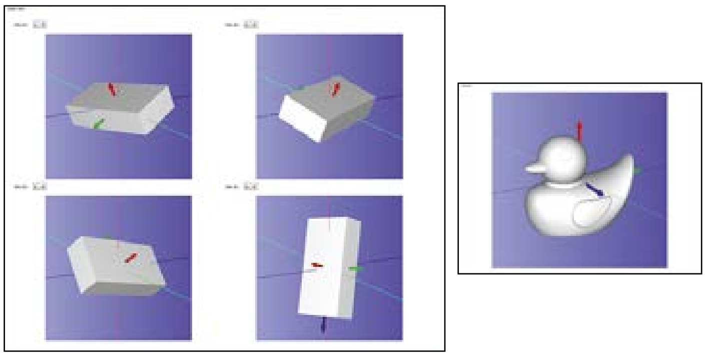
Overview

All LP-RESEARCH sensor devices are delivered with a software package. The software package contains the LPMS-Control application for general control of the sensors and a programming library for users to implement their own applications communicating with the sensor.

The LPMS-Control application allows users to control various aspects of an LPMS device. In particular the application has the following core functionality:

- Lists all LPMS devices connected to the system
- Connects to up to 256 sensors simultaneously. Multiple connection interfaces (Bluetooth, CAN bus etc.) can be used at the same time
- Adjusts all sensor parameters (sensor range etc.)
- Sets orientation offsets
- Initiates accelerometer, gyroscope and magnetometer calibration
- Displays the acquired data in real-time either as line graphs or a 3D image
- Records data from the sensors to a CSV data file
- Plays back data from a previously recorded CSV file
- Uploads new firmware and in-application-programming software to the sensor





Figures 2-4 - Orientation data can be 3D visualized in real-time, either as a simple cube or a more complex custom shape. The motion of several sensors can be displayed simultaneously, if multiple sensors are connected to LPMS-Control

Data Visualization

The screenshots below illustrate some of the functionality of the LPMS-Control software.

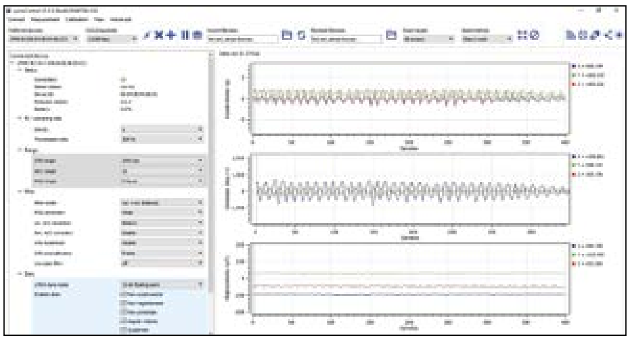


Figure 1 - Raw measurement data (gyroscope, accelerometer, magnetometer, temperature sensor, barometric pressure sensor) and processed data (orientation, linear acceleration, altitude) can be visualized as line graphs. Data can be logged and saved to a CSV file.

Sensor Calibration

Besides data acquisition, visualization and logging, LPMS-Control allows calibration of the LP motion sensor's MEMS components. The application offers offset and misalignment calibration for the unit's gyroscope and acclerometer, as well as magnetometer calibration functionality.

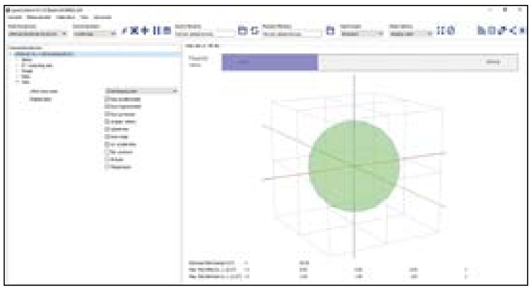


Figure 5 - The shape of the magnetic field surrounding the LPMS is visualized as a 3D sphere. A bar graph indicates the current magnetic noise level in real-time.

While accelerometer and gyroscope calibration is usually done at our factory before delivery to the customer, the calibration parameters of the magnetometer can vary strongly depending on the usage of the sensor.

Therefore the so-called hard-and soft iron calibration is an essential tool to adjust the sensor to work well in a specific environment. LPMS-Control offers a comfortable method to calibrate the magnetometer. Additionally to the pure calibration, the application also visualizes the shape of the environment magnetic field and shows the impact of magnetic noise on the sensor performance.

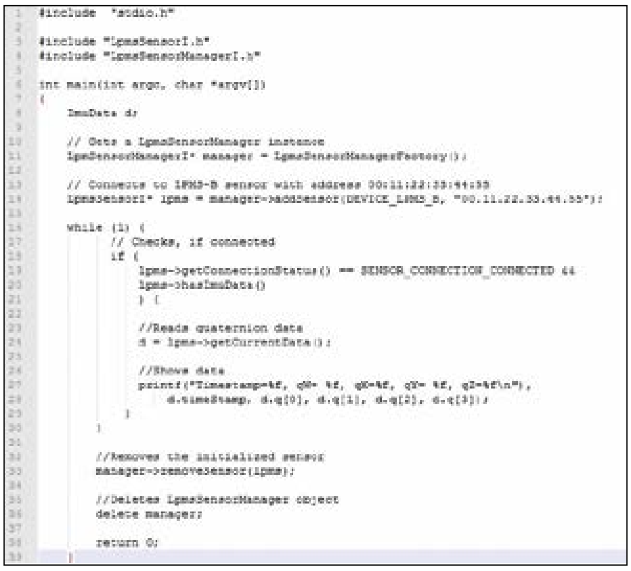
LpSensor Library

The LpSensor library contains classes that allow a user to integrate LPMS devices into their own applications. We provide the library pre-compiled for different operating systems. The standard library is written in C++, but we also offer wrappers for Python, C# and C. MATLAB and LabView ports are under development. Should your application require a specific software integration please contact us.

The code snippet below shows the most simple method of connecting to the sensor and reading data. For deeper insight in the programming library and the full source code of LPMS-Control please have a look at our code repository.

Mobile Application

As part of the open motion analysis toolkit (OpenMAT) we offer support to monitor and log data from our wireless sensors using mobile devices. See below a screenshot of a sample program for Android. The application is well suitable for data logging tasks as is. Additionally we provide source code for users to modify the program to fit their specific requirements. An iOS version with similar functionality to the Android application is currently under development.



```
#include  "stdio.h"

#include "LpmsSensorI.h"
#include "LpmsSensorManagerI.h"

int main(int argc, char *argv[])
{
    ImuData d;

    // Gets a LpmsSensorManager instance
    LpmsSensorManagerI* manager = LpmsSensorManagerFactory();

    // Connects to LPMS-B sensor with address 00:11:22:33:44:55
    LpmsSensorI* lpms = manager->addSensor(DEVICE_LPMS_B, "00:11:22:33:44:55");

    while (1) {
            // Checks, if connected
            if (
                lpms->getConnectionStatus() == SENSOR_CONNECTION_CONNECTED &&
                lpms->hasImuData()
                ) {

                //Reads quaternion data
                d = lpms->getCurrentData();

                //Shows data
                printf("Timestamp=%f, qW= %f, qX=%f, qY= %f, qZ=%f\n",
                    d.timeStamp, d.q[0], d.q[1], d.q[2], d.q[3]);
            }
        }

        //Removes the initialized sensor
        manager->removeSensor(lpms);

        //Deletes LpmsSensorManager object
        delete manager;

        return 0;
}
```

Figure 6 - The shape of the magnetic field surrounding the LPMS is visualized as a 3D sphere. A bar graph indicates the current magnetic noise level in real-time.