A MEMS BASED SYSTEM TO CONTROL OBJECTS IN HOME AUTOMATION

Giovanni Di Blasi, Selene Gallo

Department of Electrical, Electronics and Information Engineering, University of Catania
Catania, Italy

gio.diblasi@gmail.com , selene.gallo@gmail.com

Abstract

This paper shows a system that allows user to point an object into a room and take the control of it, send commands that allow to do some operations. The system is composed from two parts: server and client. The server takes data from an IMU, these data are used to detect the position indicated by user. The client is an graphical interface that simulates a software to control real objects.

I. INTRODUCTION

Recently it are making great efforts to design systems that allow to automate, or make easier, execution of daily acts. These applications in addition to making more comfortable the action that a person must do, also helps those who, for pathological reason, would find it difficult to make gesture that normally would not lead to considerable strain.

This systems are widely used in home automation context. This is a multidisciplinary science that allow to make more user friendly ma-made environment, it allow an immediate interaction with objects that surround the user, through interfaces like keyboard, touch-screen and other.

This work provides a server that interact directly with the user, so it is able to choose which object to control and to send desired command (example to open a window). The server translates these information in frame to send to client in wireless mode.

In this paper it is shown a graphical interface that emulates the client that interacts with the controllable objects, inside operative environment in which user works.

Finally an analogy is made between GUI and a real client.

II. SERVER SIDE

In this work a server interacts directly with the user. An IMU is placed on the hand of the user to determine which object he wants to control.

iNemo

An important instrument for the realization of this work is the IMU platform. This device applied to a moving object is able to detect speed and direction using a variety of sensors including accelerometers and gyroscopes.

The IMU platform used is called “iNemo”, developed by ST Microelectronics.

This board finds many areas of applications such as virtual reality, enhanced reality, stabilization of platforms, human-machine interfaces and robotics. iNemo has the following characteristics:

- a LPR530AL: Two-axis gyro (pitch, roll) with a full scale at 300 °/s;
- a LPY530AL: Two-axis gyro (pitch, yaw) with a full scale at 300 °/s;
- a LIS331DLH: triax accelerometer with a full scale selectable $\pm 2g / \pm 4g / \pm 8g$;
- a HMC5843: magnetometer triax;
- a LPS001DL: pressure sensor;
- a STLM75: a temperature sensor with a range between -55° C to +125° C.

Apart from these sensors iNemo also offers a SD card slot, a USB 2.0 connection, a ZigBee module slot and can be powered via USB or external power supply.

iNemo provides the data in various ways, each sensor provides the data found:

- the values of the three-axis accelerometer in mg;
- a value for each axis of the gyroscopes in dps (degrees per second);
- a mga value in regard to the magnetometer;
- a dmbar a value for the pressure sensor;
- a value for the temperature sensor in ° C d.

In addition to the values for each sensor iNemo also incorporates a software that can return more complex values that are the result of a merger

between the different sensors (a Kalman filter is used).

Fig. 1 shows the board iNemo.

The iNemo board has a proper reference system. It is placed on the hand of the user so that his x axis point into direction reported by user. From data given by IMU is possible to determine which object is pointed by the user (Fig. 2).

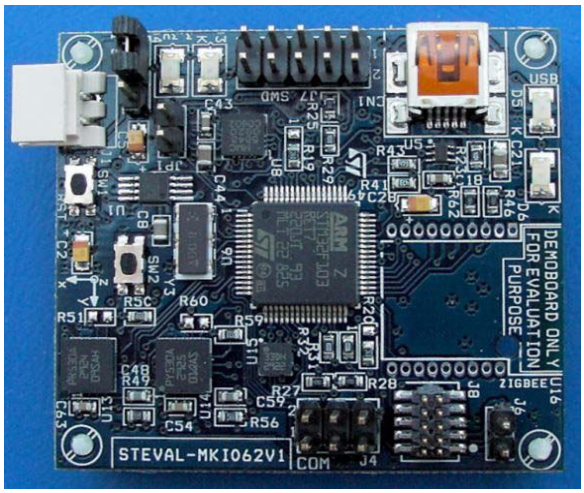

Fig. 1. The board iNemo

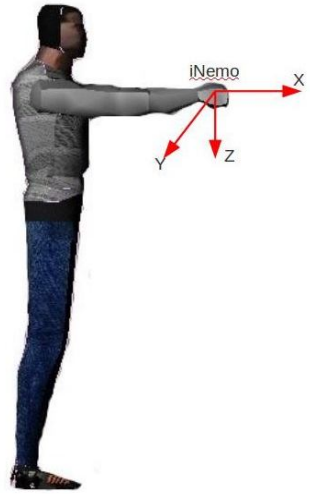

Fig. 2: iNemo reference system

Gesture recognition

Once user has engaged an object, he can send it a command making a know gesture.

The system is being configured to recognize know gestures and to record a gesture. Example of gestures are:

– "open" gesture: user moves his hand from left to right holding the hand parallel to floor;

– "close" gesture: user moves his hand from right to left;

– "up" gesture: user moves his hand from bottom to top with the hand parallel to floor;

– "down" gesture: hand is moved from top to bottom.

To recognize the gestures, accelerations are used. These accelerations are detected by accelerometer mounted on iNemo. This data are processed by two filters:

One of them filter vibrations and the other filter accelerations that are similar. The filtered data are quantized and processed by an Hidden Markov Model. After that, the data are classified using probabilistic method.

III. CLIENT SIDE

The client receives frame from server and it controls the objects in operative environment. In this work the client is implemented by a graphical interface using Ogre3D.

Ogre3D

Ogre is the acronym for Object-Oriented Rendering Engine, it's a rendering 3D engine, flexible and scene oriented. The engine is a free software, under LGPL license, and has an active community. The main role of Ogre is to give solutions for graphic rendering.

It also includes structures like matrix, vectors and tools for memory management. Choose Ogre let developers to use any library for audio, physics, etc.

Ogre is object-oriented with a structure that allow to use plug-in to add features. Because it is scene oriented, it is supported by many scene manager like octree and BSP.

Ogre is multi-platform and supports directX and OpenGL. This engine also has a compositing manager with a scripting language and a full-window post-processing to manage effects like HDR, blooming, saturation, lighting and noise.

Graphical interfaces

The GUI developed is a representation of operative environment. It shows a room with the objects recorded in the server.

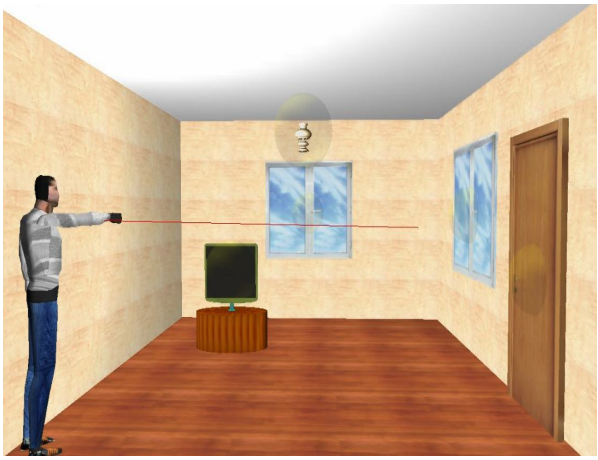

Fig. 3: GUI screenshot

The construction of the room is driven by information that this client receives with the frames exchanged during the connection stage.

Objects are placed in according to data received with “Object” frames.

Each object are surrounded by a sphere, this sphere indicates the objects state, in particular:

– a transparent sphere means that object is in idle mode;

– a red sphere indicates that the object is detected;

– a green sphere indicates that the object is engaged.

Each time the client receive a “state” frame, if the state is “ENGAGE” or “DETECTED” the sphere of the object, indicated by “object identifier” field, is changed (respectively green or red). If the state is “IDLE” all spheres are put in transparent mode.

When client receive a “command” frame, it checks his “object identifier” field to control if this object is engaged (it possible to give command only to an engaged object). The controlled object change his aspect in function to command received.

Fig. 4: Example of detected object

Fig. 5: Example of engaged object

Real Client

In this section a real client is described. It consider the following scenario: a computer communicates, in wireless mode, with object controlled by user. Each of these objects integrates a microcontroller, actuators (Engines, switches, ...) and a wireless module (for example a ZigBee module).

As well as, in the graphical interface, it associates a sphere to each object, in the real environment each device is equipped by a led to give a feedback to user.

The computer has the responsible of distributing fames which contain state of objects and the command to send.

The other modules of the network receive these frame and turn on (or turn off) the led and, eventually, operate the actuators .

In details, the system should function as follows:

The computer has a list of controllable object, and for each category of these a list of command permitted.

When computer receives frame by server, it makes some control to avoid inconsistencies before send frame to device.

The first type of control provides to check if the command received for a specific object is in the list of category of that object.

For “device” category is less easy: this category contains several kind of object, so the client doesn't know what commands are permitted. A solution consist in transfer, only for this category, of the control to device (each device defines in his firmware what command can manage).

Another control avoids to send duplicated commands: if a command is sent twice consecutively, second time this command is ignored. In this way network overload is avoided.

Devices receive frame sent by client, if type of this frame is “object's state” then the led is turned on/off, if it is “command” the actuator is activated.

A Led red is turned on if object is “detected”, led green if it is “engaged”. Each led is turned off if the state is “idle”. This is the same behavior seen for the spheres in the graphical interface.

The kind of actuator depends by object category: for a window, for example, it will be an engine (to open or close), for a lamp it will be a switch.

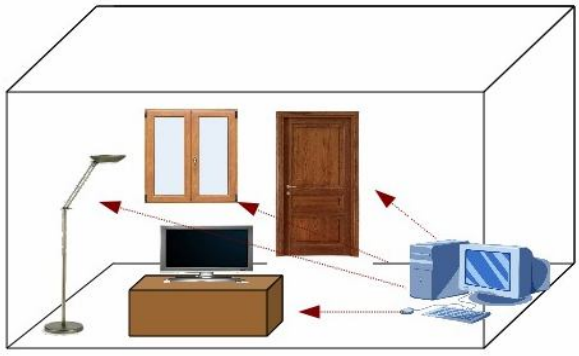

Fig. 6: Real client application

IV. COMMUNICATION

Server and client communicate through sockets. The server receives a connection request by a client and sends a “connectionACK” that includes the recorded objects number. In this way the client knows

how many objects are in the room and, for each objects, sends a frame “ReqObject” to have information about them. The server that receive this frame, answers with a “Object” frame that includes following information:

- object identifier;
- object category;
- a flag to determine if the object is under user control.

In a second moment client send a “ReqParam” frame in order to obtain information about the operative environment. When the server receive the “ReqParam” frame, it responds with a “Parameter” frame with all information needed.

At this point the server send to client two kind of frame: State and Command.

The first kind of frame contains information about the objects state. It specifies if there is an object under the user control, includes the object identifier and a field that assumes following values:

- DETECT: object, specified by identifier, is detected;
- ENGAGE: object, specified by identifier, is engaged;
- IDLE: all objects are in idle state.

The second kind of frame, “Command”, specifies the command given by user to a specific object (for example “open”, “close”, etc.). In this frame there are following fields:

- Object identifier;
- Command number.

The command number is an identifier of a specific command.

All of this frames are confirmed by an ACK transmission.

V. CONCLUSIONS

The system designed aid user to make routine action, it can work in home environments and other man-made environments. Empiric proofs have enlightening a good precision of the system. There are some imprecision on recognizing gestures because it used a probabilistic method.

REFERENCES

[1] Oliver J. Woodman. An introduction to inertial navigation. Technical Report 696, University of Cambridge, August 2007.

[2] N.Abbate, A. Basile,C. Brigante, A. Faulisi, Development of a MEMS based wearable motion capture system. HSI'09 Proceedings of the 2nd conference on Human System Interactions.

[3] STEVAL-MKI062V1, iNEMO. User Manual. Available: http://www.st.com/internet/com/TECHNICAL_RESOURCES/TECHNICAL_LITERATURE/USER_MANUAL/CD00241278.pdf

[4] Gary Bishop and Greg Welch, “An Introduction to the Kalman Filter”. University of North Carolina SIGGRAPH 2001 course notes. ACM Inc., North Carolina,2001 .

[5] Giuseppe Mastroeni, Riconoscimento di gesti tramite accelerometro e applicazioni orientate alla domotica per S.O. Android. Master Thesis 2010.

[6] Ogre: Open Source 3D Graphics Engine, sito: http://www.ogre3d.org.

[7] Ogre 3D Italia, sito: http://www.ogre3d.it.