Smart device definition and application on embedded system: performance and optimization on a RGBD sensor

Jose-Luis Jimenez-Garcia[a], David Baselga-Masia[a], Eduardo Munera[b], Jose-Luis Poza-Lujan[b], Juan-Luis Posadas-Yagüe[b], José-Enrique Simó-Ten[b]

[a] School of Engineering in Computer Science, Universitat Politècnica de València
[b] University Institute of Control Systems and Industrial Computing, Universitat Politècnica de València

KEYWORD

Smart device
RGBD Sensor
Data acquisition
Embedded system

ABSTRACT

Embedded control systems usually are characterized by its limitations in terms of computational power and memory. Although this systems must deal with perception and actuation signal adaptation and calculate control actions ensuring its reliability and providing a certain degree of fault tolerance.
The allocation of these tasks between some different embedded nodes conforming a distributed control system allows to solve many of these issues. For that reason is proposed the application of smart devices aims to perform the data processing tasks related with the perception and actuation and offer a simple interface to be configured by other nodes in order to share processed information and raise QoS based alarms.
In this work is introduced the procedure of implementing a smart device as a sensor as an embedded node in a distributed control system. In order to analyze its benefits an application based on a RGBD sensor implemented as a smart device is proposed.

1 Introduction

The distribution of computational load between different nodes of a distributed system is a common procedure for improving the performance of almost any environment, even in a control based one [LIAN, F. et al. 2002]. The low cost of the embedded system characterized as nodes promotes this kind of implementations. Some nodes, presented as smart devices, offers the capability of working with high level information, such as pre-processed values, by offering a communication interface [BRIGNELL, JE. 1996].
Smart resources are especially suitable for these cases in which large amounts of data must be managed following a continuous flow. For that reason, in a classic control environment its task must be related with the management of sensors and actuators. In the first case, smart devices will be responsible of the acquisition and processing of the raw sensor information and offer a network interface to access to the obtained values. In the second case, must be performed the adaptation of the provided high level information to a low level actuator signal.
Nevertheless, this article will be focussed on the implementation of smart devices as sensors and its characterization as vision sensor due to the range of its possible applications [FERNANDES, J. 2013] , focusing in this concrete case to support robot navigation [CHIEN-HUI, L. 2014] [GRZEJSZCZAK, T. 2012]. It will be also remarked the importance of being adapted to the system requirements [LEE, C.S. 2013]. In order to provide this adaptation capability some

speed optimizations [DAN, A. D. 1995] and data management mechanisms [TAGAMI, Y. 2013] [KHAN, S. 2009] are evaluated and adapted to improve the smart device performance. It will be also remarked the importance of the Quality of Service (QoS) and Quality of Control (QoC) for obtaining a reliable performance and fault tolerance [POZA-LUJÁN, J.L. 2014].

This work is structured as follow: the definition and characterization of the smart devices as sensors is introduced in Section 2. In section 3 the implementation of a triple buffer as the main management mechanism is detailed. A plugin topology for raw data process is described along Section 4. An application case based on a RBGD sensor in Section 5 leads to some experimental results properly summarized on Section 6. Finally Section 7 gathers the benefits and issues of the proposal in addition to the future work.

2 Smart devices: sensors

As it has been previously defined, a smart device which performs as sensor has to deal with raw data obtained by measuring a physical magnitude and process it to offer high level data accessible by a proper interface. The smart device has three different steps: an initial data acquisition, intermediate data process and finally a communication step (Figure 1).

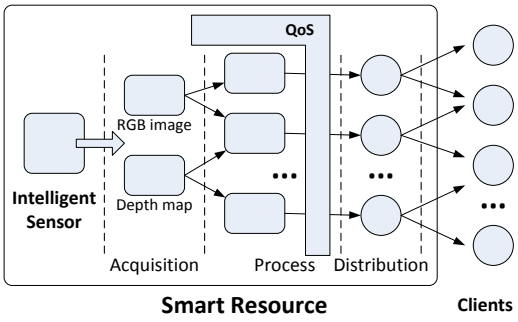

Fig. 1. Smart device on a distributed system.

During the acquisition step, smart device will have to store the values provided by the physical sensor device, ensuring a proper period selection, as defined by the Nyquist theorem [FERRARI, P. *et al*. 2011] and checking its accomplishment.

In the second step raw information is processed by the application of several plugins (plugins are isolated processes for a specific purpose), just as will be described on Section 4.

Finally processed date is supplied to the rest of the system by evaluating some QoS parameters such as time, latency or jitter between others in order to detect system malfunction.

3 Data path: Triple buffer

In order to optimize the data flow between the preciously introduced steps is presented a triple buffer mechanism. This triple buffer offers a more efficient usage of the disposed resources by providing a proper data management.

Proposed solution works as follows: a buffer is used for the current data acquisition, while another contains the processed data prepared for being sent through the communication system while the third buffer is an auxiliary one which prevents bottlenecks. The triple buffer is managed by two threads: the acquisition and process thread (APT), and the sending threads (ST). The APT changes between the acquisition buffer and the auxiliary while the ST ends sending its buffer. The Swap function does the buffer pointers exchange which corresponds. When the exchange is done, the APT sends a signal to ST to pick the last image. The APT behaviour is as follows.

```
1: I →0
2: Acquisition()→frame
3: while I < frame.MaxPixel do
4:                    frame.Process()
5: end while
6: Swap()
7: Signal()
```

When the ST has sent an image it waits, using the Wait() function, to be communicated by APT that the last image is available. As soon as the ST is unlocked by the APT from the wait function, the ST does the buffers pointer swap to send the last available image.

1: Wait()
2: Swap()
3: Send()

Because two threads are employed to decouple the acquisition from the distribution, we can find two scenarios: APT serves a frame faster than ST sends it and otherwise. In first case, the frame sequence is lost in exchange for gaining immediacy in the information. In figure 2 the sending sequence is shown where it can be appreciated how the frame i+2 is lost because the frame i+3 is acquired before the frame i+1 ends being sent. Buffer pointers are represented as A1, A2 and A3, the result of the Wait() function is the interchange between A1 and A2 (for example). In figure 3 the opposite case is shown, the ST is faster than the APT so no message in the sequence is lost.

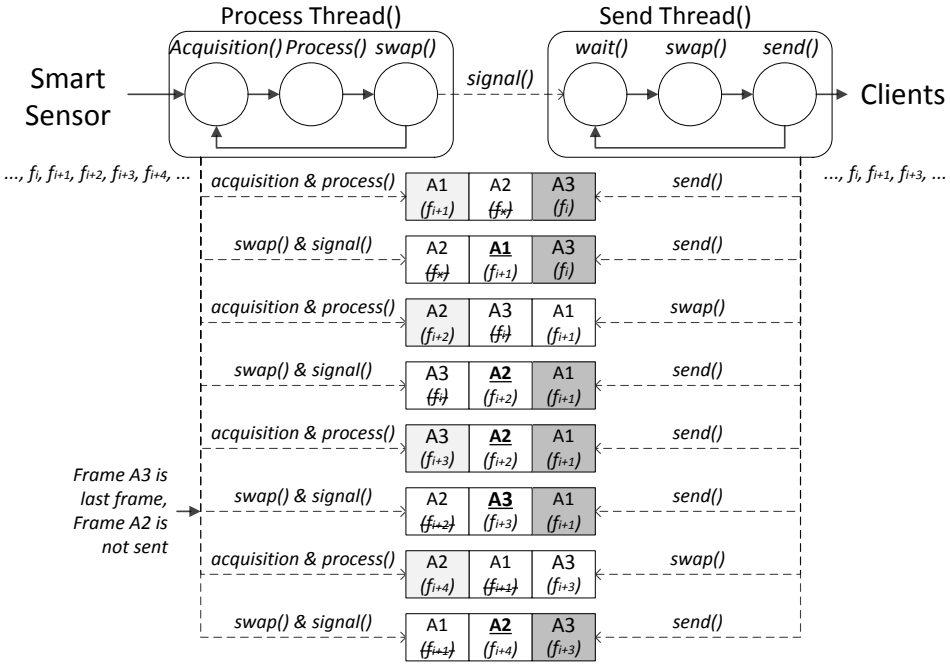

Fig. 2 Sequence of operation of triple buffer with APT faster than the ST.

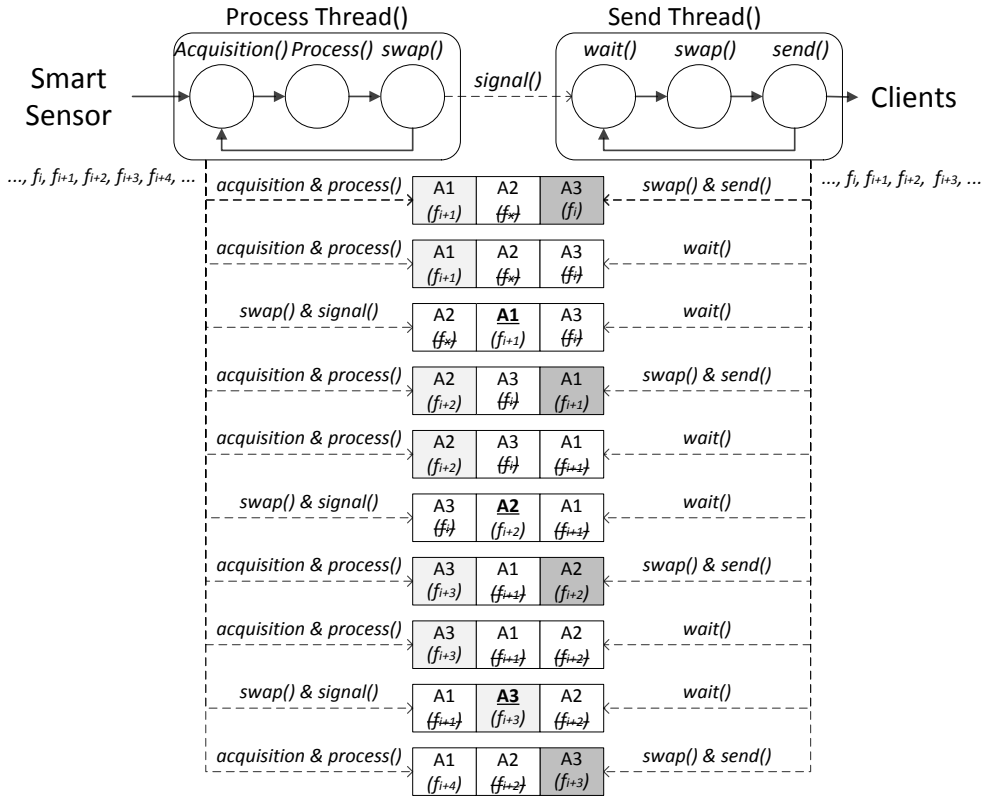

Fig. 3 Sequence of operation of triple buffer with ST faster than the AP

.

4 Process architecture: Plugin

In order to perform the processing step, it is necessary to execute various specific processes, called as plugins. Furthermore, plugins must have capacity of organizing among them, due to the data of one of them being (possibly) relevant to others. The structure within the plugins are organized, has been called Smart Plugin Topology (SPT). The main objective of the SPT is to manage efficiently the processing stage in order to avoid producing duplicities or an inappropriate use of the system resources (Figure 4).

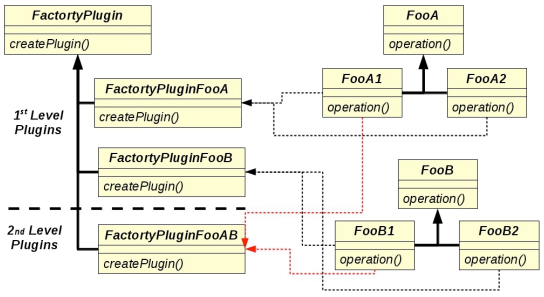

Fig. 4. Smart Plugin Topology (SPT).

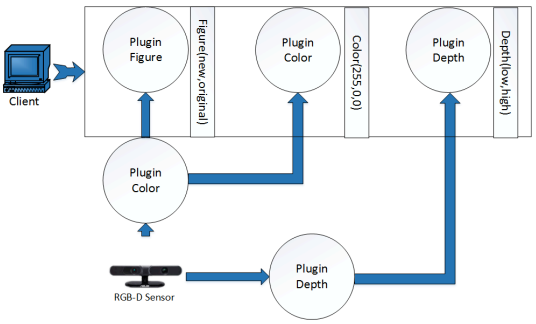

Fig. 5. Plugin composition example.

The plugins may be combined between them to generate dynamically new plugins and, then, being able to design specific plugins with the clients requirements. In Figure 5, it is shown an example of the design pattern based in the Composite pattern [GAMMA, E. et al. 2006] used to join plugins. The example shows an plugin based method to detect certain objects type, using a depth camera. This task is very useful to do, for example to count people [VAN OOSTERHOUT, T. Et al., 2012].

The SPT needs a structure that supports the wide plugins variety of the smart device. Besides, the SPT must provide operations that allow using various plugins to join data from the smart sensor and generate enriched information.

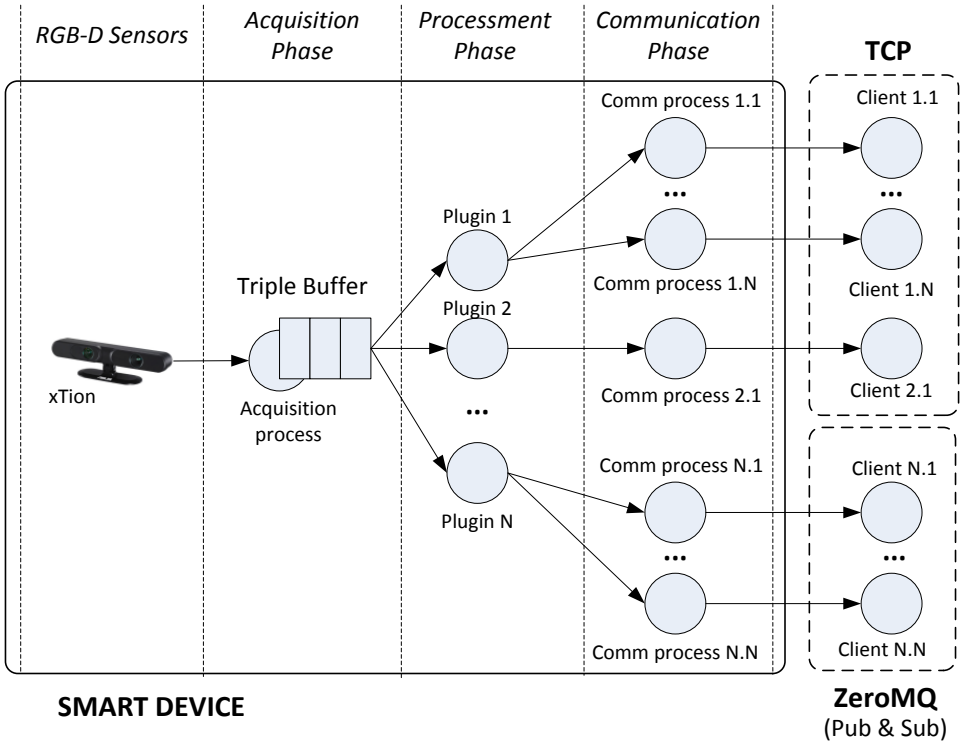

Fig. 6. RGBD Smart device components and phases.

5 RGBD as a smart device

The smart device implementation is made by means of the inclusion of an RGB-D sensor, model xTION [GONZALEZ-JORGE, H. 2013], for the capture of the pictures, and also a Raspberry PI (RPI) [UPTON, E., et al. 2013] in order to integrate the acquisition, processing and communication stages.

To acquire the pictures, it is used the OpenNI v2 [FALAHATI, S. 2013] API to do the different settings of the device. The acquisition stage is optimized by using a variation of the triple buffer algorithm [JIMENEZ-GARCIA, JL. 2014] that improves the obtaining of an only RGB-D sensor when various consumers need the data. In order to make easy the plugins programming, OpenCV v2.4.8 [BRADSKI, G. 2008] API functions are used. Each plugin sends its result to the consumers, limiting the number of connections that guarantees the QoS of the clients. These consumers might access to plugins establishing a TCP [PETERSON, L. 2007] connection or being subscribers by the use of a Publisher-Subscriber (PUB-SUB) ZeroMQ [HINTJENS, P. 2013] pattern. Details of the RGBD Smart Device can be viewed on Figure 6.

The plugins used for data processing are gathered within Table 1, offer information of the smart sensor with a variable processing level. For example, Colour plugin supplies the raw picture or changing the channels, Grey plugin transform from three channels to just one, what entails more processing. Some plugins, like Contour and Motion, need an advanced processing.

Plugin	Description
Colour	Provides RGB image or transformation to BGR.
Grey	Provides the image, but in grey scale.
Contour	Recognizes outlines in the image given by the smart sensor.
Resize	Resizes the image to a specific size.
Soften image	Erases imperfection like, for example, excessive bright or reflections.
Motion Detection	Detects moving objects.
Colour detection	Allows client to detect pixels of a specific colour.
Grey Depth	Represents a depth matrix in grey scale.
Depth	Distances matrix of the image.
Binary Depth	Provides a binary matrix where 1, represents the existence of an object in the sight range.
Closest Farthest Point	Provides the nearest and furthest object in the sight range.

Table. 1. Description example.

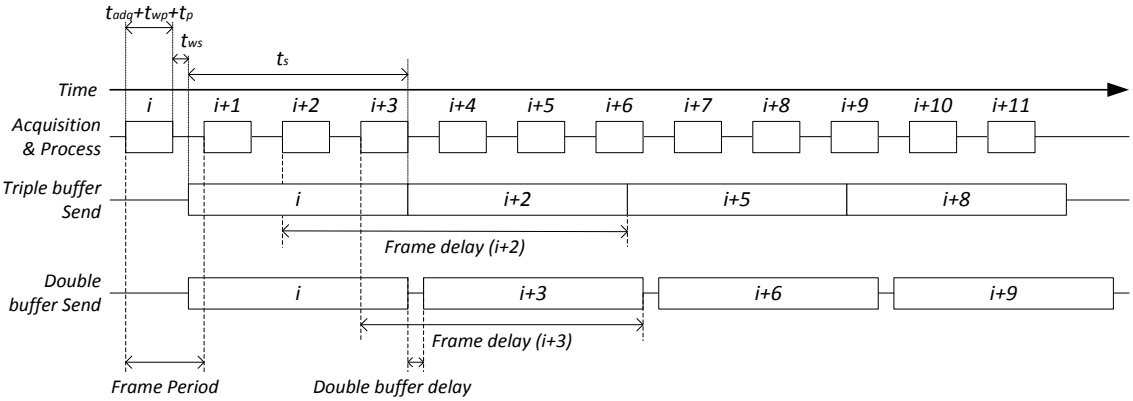

Fig. 7 Variables measured at the experiments.

6 Experiments and results

To check the correct operation of triple buffer implemented, it has been tested using a computer as smart resource to check highest level of efficiency, and a raspberry PI on the final system. Because the system must operate in a distributed closed network, servers and clients work with fixed IP addresses, thereby, the effect of messages noise on network is reduced. All tests were performed with loads of 1000 frames and 30 frames per second (fps) as sampling rate. Previously, it has been verified that the number of frames not affect the operation. Has been measured times with one, two and four clients, comparing the performance of a simple, double and triple buffering based server. The measured variables are the average of the all the frame delays. Frame delay is defined in the equation 1.

$$t_{frame_{delay}} = t_{adq} + t_{wp} + t_p + t_{ws} + t_s \qquad (1)$$

In equation 1, t_{adq} represents the time inverted to acquire new measures from the sensor, t_{wp} represents the time that the APT is waiting to obtain the measures and the t_p is the time used by the APT to process the measures, t_{ws} is the time

that the ST waits for new sample, if it is necessary, and t_s is the time inverted to send the processed measures (data) to all clients.
The results for the frame delay, expressed in microseconds, obtained are shown at table 1.

Variables	1 PC client	1 RPI client
Double buffer	67350 μs.	203086 μs.
Triple buffer	66940 μs.	199647 μs.

Table. 2. Description example

Variables	2 PC clients	2 RPI clients
Double buffer	66936 μs.	334824 μs.
Triple buffer	67386 μs.	332900 μs.

Table. 3. Description example

Variables	4 PC clients	4 RPI clients
Double buffer	66945 μs.	640506 μs.
Triple buffer	66930 μs.	611752 μs.

Table. 4. Description example

The main difference between double and triple buffering is that the triple buffer always has a frame ready to send. However, the triple buffer adds more delay time. These results are predictable but it's necessary that the smart resource can provide to the clients in order to improve the system with quality of service policies.
As can be seen in the table 1, when the image server is a powerful computer (PC) differences between the types of buffer are not significant. However, an embedded system provides less efficient results for a simple buffer, for a double buffering and triple buffering delay times is less in the case of double buffering when we increase the client number. So that, triple buffer offers the same updated information with an added immediacy and more efficiently than double.
In triple buffering there is always a frame available to be sent. So that it is not necessary to wait for the image acquisition to send a new frame. With this method, an intelligent sensor can provide the same performance as the double buffering method, but performing others tasks, as image processing, image segmentation or image resizing.

7 Conclusions

In this article the definition and implementation of a sensor as a smart device has been detailed. Thus, sensor tasks have been divided into three main steps: acquisition, processing and distribution.
The flow rate of the processed data distribution has been improved by the application of a triple buffer, which always guarantees to provide this kind of information. Next the design of a SPT eases the implementation of different data processing methods by the application and combination of the available plugins in each sensor.
For validating this proposal a Asus Xtion in combination with a RaspBerry Pi has been implemented as a smart device. Furthermore several plugins, like colour or depth image acquisition, have been developed.
In the result section the Xtion-based smart device is configured to serve processed data to a variable number of clients. During this experiment the performance of implementing a triple buffer has been compared to the application of a double buffer data path. That way, the efficiency of the triple buffer has been verified.
As conclusion the application of a smart device in a distributed control system allows to enhance its performance by the distribution of the load, just as has been proved.

8 Acknowledgment

This work has been supported by the coordinated project COBAMI: Mission-based Hierarchical Control. Education and Science Department, Spanish Government. CICYT: MICINN: DPI2011-28507-C02-01/02 and project "Real time distributed control systems" of the Support Program for Research and Development 2012 UPV (PAID-06-12)

9 References

[BRADSKI, G. *et al*. 2008] BRADSKI, Gary; KAEHLER, Adrian. *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.", 2008.

[BRIGNELL, JE. et al. 1996] BRIGNELL, J. E. The future of intelligent sensors: a problem of technology or ethics?. Sensors and Actuators A: Physical, 1996, vol. 56, no 1, p. 11-15.

[CHIEN-HUI, L. et al. 2014] LIAO, Chien-Hui Christina, et al. Fall Detection by a SVM-Based Cloud System with Motion Sensors. In Advanced Technologies, Embedded and Multimedia for Human-centric Computing. Springer Netherlands, 2014. p. 37-45.5

[DAN, A. D. et al. 1995] DAN, Asit, et al. *Buffering and caching in large-scale video servers*. IEEE, 1995.

[EDWARDS, C. 2013] EDWARDS, Chris. Not-so-humble raspberry pi gets big ideas. *Engineering & Technology*, 2013, vol. 8, no 3, p. 30-33.

[FALAHATI, S. 2013] FALAHATI, Soroush. *OpenNI Cookbook*. Packt Publishing Ltd, 2013.

[FERNANDES, J. et al. 2013] FERNANDES, João, et al. A context aware architecture to support people with partial visual impairments. In Distributed Computing and Artificial Intelligence. Springer International Publishing, 2013. p. 333-340.

[FERRARI, P. *et al*. 2011] FERRARI, Paolo; FLAMMINI, Alessandra; SISINNI, Emiliano. New architecture for a wireless smart sensor based on a software-defined radio. *Instrumentation and Measurement, IEEE Transactions on*, 2011, vol. 60, no 6, p. 2133-2141.

[FREESE, M. *et al*. 2010] FREESE, Marc, et al. Virtual robot experimentation platform v-rep: a versatile 3d robot simulator. In *Simulation, Modeling, and Programming for Autonomous Robots*. Springer Berlin Heidelberg, 2010. p. 51-62.

[GAMMA, E. *et al*. 2006] GAMMA, Erich, et al. *Design patterns: elements of reusable object-oriented software*. Pearson Education, 1994.

[GONZALEZ-JORGE, et al. H. 2013] GONZALEZ-JORGE, H., et al. Metrological evaluation of microsoft kinect and asus xtion sensors. *Measurement*, 2013, vol. 46, no 6, p. 1800-1806.

[GRZEJSZCZAK, T. et al. 2012] GRZEJSZCZAK, Tomasz, et al. Gesture based robot control. In *Computer Vision and Graphics*. Springer Berlin Heidelberg, 2012. p. 407-413.

[HINTJENS, P. *et al*. 2013] HINTJENS, Pieter. *ZeroMQ: Messaging for Many Applications*. " O'Reilly Media, Inc.", 2013.

[JIMENEZ-GARCIA, JL. *et al*. 2014] JIMENEZ-GARCIA, Jose-Luis, et al. Performance and Results of the Triple Buffering Built-In in a Raspberry PI to Optimize the Distribution of Information from a Smart Sensor. In *Distributed Computing and Artificial Intelligence, 11th International Conference*. Springer International Publishing, 2014. p. 279-286.

[KHAN, S. et al. 2009] KHAN, Shujjat; BAILEY, Donald; GUPTA, G. Simulation of Triple Buffer Scheme. In *Second International Conference on Computer and Electrical Engineering*. 2009.

[LEE, C.S. et al. 2013] LEE, Chae Sub; LEE, Gyu Myoung; RHEE, Woo-Seop. Standardization and challenges of smart ubiquitous networks in ITU-T. *IEEE Communications Magazine*, 2013, vol. 51, no 10, p. 102-110.

[LIAN, F. *et al*. 2002] LIAN, Feng-Li; MOYNE, William; TILBURY, Dawn. Network design consideration for distributed control systems. Control Systems Technology, IEEE Transactions on, 2002, vol. 10, no 2, p. 297-307.

[OLLERO, A. 2005] CUESTA, Federico; OLLERO, Aníbal. *Intelligent mobile robot navigation*. Springer, 2005.

[PETERSON, L. *et al*. 2007] PETERSON, Larry L.; DAVIE, Bruce S. *Computer networks: a systems approach*. Elsevier, 2007.

[POZA-LUJÁN, J.L. et al. 2014] POZA-LUJÁN, José-Luis; POSADAS-YAGÜE, Juan-Luis; SIMÓ-TEN, José-Enrique. Quality of Control and Quality of Service in Mobile Robot Navigation.*International Journal of Imaging and Robotics*, 2014, vol. 8, no 1.

[TAGAMI, Y. et al. 2013] TAGAMI, Yuya; WATANABE, Makoto; YAMAGUCHI, Yuko. Development Environment of 3D Graphics Systems. *Fujitsu Scientific & Technical Journal*, 2013, vol. 49, no 1, p. 64-70.

[UPTON, E., *et al*. 2013] UPTON, Eben; HALFACREE, Gareth. *Raspberry Pi user guide*. John Wiley & Sons, 2013.

[VAN OOSTERHOUT, T. *Et al*., 2012] VAN OOSTERHOUT, Tim; KRÖSE, B.; ENGLEBIENNE, Gwenn. People counting with stereo cameras: two template-based solutions. In International Conference on Computer Vision Theory and Applications (2) 2012, pp. 404-408.

[VILLAROMAN, N. *et al*. 2011] VILLAROMAN, Norman; ROWE, Dale; SWAN, Bret. Teaching natural user interaction using OpenNI and the Microsoft Kinect sensor. In *Proceedings of the 2011 conference on Information technology education*. ACM, 2011. p. 227-232.