2014 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS 2014)
September 14-18, 2014, Chicago, IL, USA

# A Real-Time Distributed Architecture for Large-Scale Tactile Sensing

Emanuele Baglini[1], Shahbaz Youssefi, Fulvio Mastrogiovanni and Giorgio Cannata

***Abstract*— This article discusses a real-time networking infrastructure for a large-scale tactile sensing system to be used with humanoid robots. In such a system, real-time networking issues are of the utmost importance. Stemming from previous work, a theoretical model is presented and experimentally validated. Tests show real-time performance in a network of distributed computational nodes, each one in charge of managing part of the tactile system.**

## I. Introduction

Recently, different solutions to the development of tactile sensors have been presented. Whilst the focus on specific sensor technologies is important to assess advantages and drawbacks of transducer modes, robots required to perform complex interaction tasks with humans or the surrounding environment must exploit tactile feedback from large areas of their body. Examples of work related to large-scale tactile sensing are not numerous and quite recent [1], [2], [3], [4], [5], [6], [7], [8].

The concept of *robot skin* entails a number of system level problems that simply do not appear when focusing on small tactile sensor arrays. On the one hand, robot skin must be modular, conformable, easy to manufacture [7] and deploy [9]. On the other hand, large-scale sensing requires to consider ancillary issues like infrastructure, networking, fault-tolerance and distributed (real-time) processing [10], to name but few.

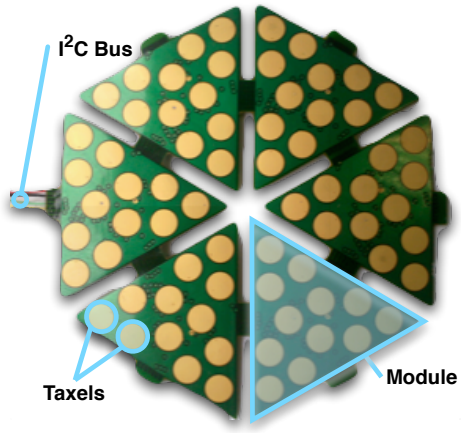


Fig. 1. The reference skin system: a skin patch.

[1]All the authors are with the Department of Informatics, Bioengineering, Robotics and Systems Engineering, University of Genova, Via Opera Pia 13, 16147, Genova, Italy. Corresponding author's e-mail emanuele.baglini@unige.it.

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n. 231500 (ROBOSKIN).

The main contribution of this article is the performance assessment of an EtherCAT-based real-time distributed architecture for large-scale tactile sensing. The reference tactile technology [6], [7] is based on a number of interconnected triangular modules, each one hosting 12 capacitive tactile elements as well as read-out electronics, which can be arranged in *skin patches* (Figure 1). Skin patches are associated with managing microcontrollers, which convey information to a centralized embedded workstation through a communication bus. Stemming from previous work [10], the article focuses on the general distributed architecture, with a special emphasis on extending the previous model and on performance assesment. This is very important in humanoid robots, where tactile-based feedback control loops are expect to guarantee a proper general behavior as well as fast reaction times to unexpected contacts.

In the literature, the analysis and performance evaluation of distributed embedded networks is focus of active research activity. In [11] an evaluation of a Linux based master for EtherCAT networks is performed. Real-time performance is guaranteed by the underlying OS kernel that exploits the RT Patch. In [12] an EtherCAT-based data acquisition system is analysed with respect to the global system performance. In [13] an interesting analysis of an EtherCAT-based system performance is carried out. The analysis is based both on mathematical models and with a set of experiments. With respect to the work described in this article, which is focused on sensing, the analyses reported in [13] address motion control tasks.

The article is organised as follows. Section II describes the global architecture, focusing on distributed sensing nodes and on the communication bus. Section III discusses relevant performance tests. Conclusion follows.

## II. An Architecture for Acquiring Large-scale Tactile Information

The robot skin described in the previous Section can be considered as a distributed system where a number of nodes acquire and communicate tactile data to one or more centralized workstations for further processing. In this Section, the envisaged real-time architecture is described (Figure 2). Relevant subsystems are as follows.

1) Skin *patches*, organized in triangular modules, each one hosting 12 capacitive taxels. Each patch is connected to a managing microcontroller board through a number of I2C bus lines.
2) A number of distributed *slave* nodes responsible for collecting tactile data from managed patches. Each slave node includes a microcontroller for early data

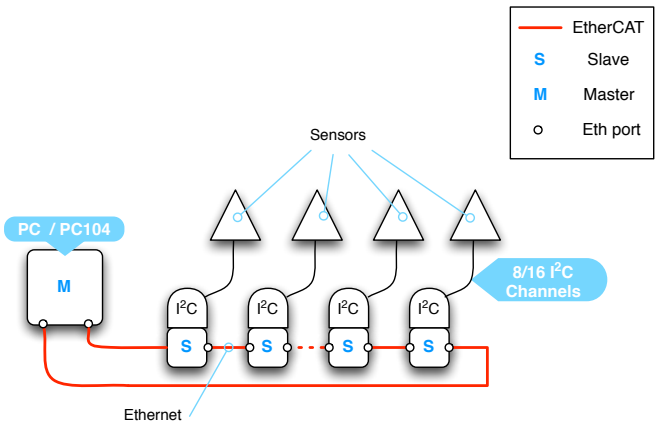


Fig. 2. General System Design.

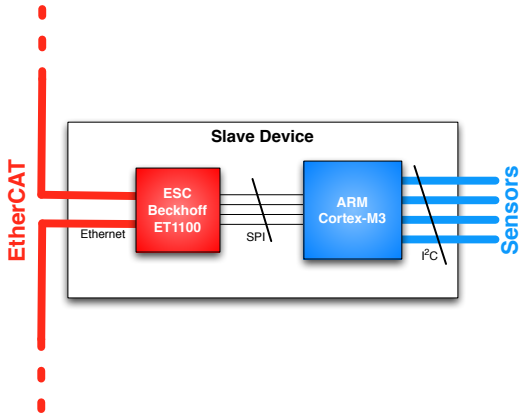


Fig. 3. Slave with SPI Bus.

processing as well as ancillary electronics for interfacing with the EtherCAT communication bus.

3) An EtherCAT based *communication bus* to connect slave nodes with a centralized master node.
4) A *master* node responsible for hosting algorithms for high-level data processing and EtherCAT network management. To enforce compatibility with respect to typical software architectures for Robotics, this node is based on a standard PC with an RTAI patched Linux kernel and the open source IgH EtherLab EtherCAT master[1].

The reference robot skin has been described in [6], [7]. The following Sections are focused on the design of the architecture of slave and master nodes as well as the EtherCAT based networking infrastructure.

### A. Slave Nodes

Slave devices have been designed both at the hardware and firmware levels (Figure 3). The architecture is based on two main subcomponents: a microcontroller acquires information from the local skin patch through an I2C interface, and an EtherCAT ESC chip is responsible for processing the EtherCAT based communication protocol. The chip enforces high-performance and real-time capabilities and frees software processes on the microcontroller from dealing with communication issues. The two components are connected through a Serial Peripheral Interface (SPI) bus line.

The ARM Cortex-M3, which has been selected as candidate microcontroller, guarantees a good trade-off between computational power and ease of embedding. Furthermore, from an engineering perspective, it is widely commercially available by different manufacturers in different packages and with different configurations of on-chip peripherals. The adopted ESC chip is ET1100 from Beckhoff[2]. Since slave nodes must be embedded in robot mechanical structures, special care has been devoted to reduce space occupation, by considering the BGA128 version with a footprint of $10 \times 10mm$. Furthermore, in the current implementation, the ARM Cortex-M3 and the ET1100 chip are connected using an SPI bus, which requires only 4 wires for communication. It is a synchronous serial data link standard operating in full duplex mode, with a transfer rate of 20 $Mbit/s$. Connected devices communicate in a master/slave mode where the master device initiates the data frame.

### B. EtherCAT as a Real-Time Communication Bus

As a result of previous work [10], the EtherCAT technology has been selected as the most promising framework to manage the lower layers of a networking architecture for large-scale tactile sensing. In this Section we provide an brief overview of the EtherCAT protocol and an updated version of its behaviour model first described in [10].

*EtherCAT Overview.* EtherCAT (ECAT) is a modified Ethernet master/slave communication bus where real-time performance is enforced. In a typical configuration, an Ethernet frame is sent each cycle by the master node through the bus reaching all the slave nodes in sequence. Each node is provided with *at least* an input and an output Ethernet interface. Slaves read the data addressed to them and provide the master node with information by encapsulating the data in the passing frame, which at the end of the sequence reaches the master node again. It is noteworthy that slave nodes operate "on the fly": differently from other protocols, where a frame is first received, stored, and eventually modified and sent back, in ECAT it only flows between the two Ethernet interfaces of each node, i.e., it is read from the first and written to the second, *byte per byte*. If the information to be sent can not fit within one Ethernet frame, the ECAT protocol exploits more frames in a transparent way.

ECAT adopts standard Ethernet frames encapsulating *telegrams* as shown in Figure 4. On the one hand, this compatibility allows a master to be implemented using any embedded workstation compatible with standard Ethernet. On the other hand, slaves elaborating data "on the fly" must be equipped with dedicated hardware. As anticipated in the previous Section, this specialized hardware is implemented in the so-called *ESC* chip. Each Ethernet frame in the

[1]For further information refer to the official RTAI and EtherLab websites at `www.rtai.org` and `www.etherlab.org`.

[2]Please refer to the official Beckoff website at `www.beckhoff.com`.

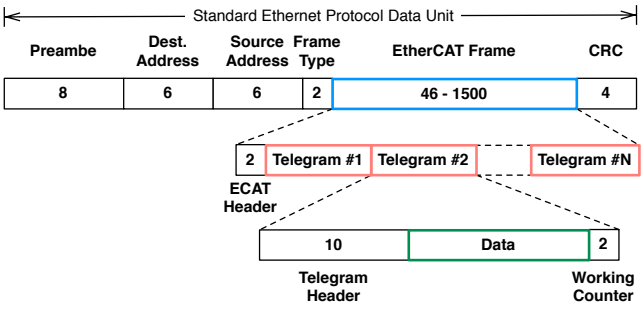


Fig. 4. EtherCAT Frame.

payload contains a 2 *B* ECAT header and *n* ECAT telegrams. Each telegram contains a 10 *B* header, which specifies the operation type (e.g., *read*, *write* or *read/write*), a data field, where master and/or slave nodes write the information that must be exchanged (i.e., from master to slaves, from slaves to the master or between slaves) and, at least, a 2 *B* working counter, which is used by the master to verify the correctness of the execution of the operation specified in the telegram header. With respect to fault tolerance, since the Ethernet frame CRC is checked by each device, bit errors are not only immediately detected (i.e., between two nodes), but can be also located exactly by checking the error counters.

From a logical perspective, an ECAT network is based on a ring topology, implemented by point-to-point links between nodes. The ECAT standard allows for variants in topology, which are based on combinations of basic ones, such as daisy chain, star or tree. As argued in [10], besides the classical and simple "open ring" configuration, in order to address issues related to fault-tolerance and redundancy, ECAT allows for a "closed ring" configuration aimed at guaranteeing the whole network to properly function even in presence of faulty nodes or damaged cables. Both the configurations do not require special hardware in the master node: a second Ethernet interface is enough for the "closed ring" solution. At the same time, all slave nodes with two or more Ethernet ports support the cable redundancy feature. According the ECAT standard benchmarks, the recovery time in case of cable failure is shorter than 15 $\mu sec$.

*EtherCAT Network Model.* In this Section we extend the mathematical analysis of the behaviour of the ECAT network presented in [10] by considering the use of Fieldbus Memory Management Units (FMMUs). It is noteworthy that the description does not assume the knowledge of the previous model.

As previously discussed, the ECAT telegram structure allows for addressing directly each slave node with a different *read*, *write* or *read/write* operation. Unfortunately, this feature adds significant overhead for communication (i.e., 12 *B* for each telegram). A better approach consists in *logical addressing*, which assumes that the master node can perform operations on a logical memory – managed by FMMUs – mapping a part of or the whole input/output process data of slave nodes. Each slave can be configured to use one o more FMMUs, whereas the master node can specify different logical addresses referring to different FMMUs (either on the same or different slave nodes) inside each ECAT telegram. When the ECAT frame flows through slave nodes, each of them checks whether the logical address matches one or more of its FMMUs: if this is the case, the slave node can perform its operations. For example, from a system designer point of view, this allows for grouping all the *skin related* nodes together, or for grouping nodes according to the robot kinematic chain (e.g., *left forearm* or *torso*).

The performed analysis focuses on *cycle time*, i.e., the time necessary for the ECAT frame to transport data between the master and all the other slave nodes, and *vice versa*. This parameter is of utmost importance in real-time aspects associated with tactile sensing. As a matter of fact, it has a great impact on the response time of a robot in contact with external objects, which must be – in general – kept at a minimum. A number of assumptions are in order:

- all slave nodes are polled each cycle to obtain data;
- only real-time process (i.e., tactile) data is exchanged;
- network topology is *daisy chain* (both non redundant and redundant cases are considered);
- the length of each Ethernet cable between two subsequent nodes is roughly 0.01 *m*;
- only the communication with the ESC chip is considered, i.e., we assume that when the EtherCAT frame reaches a specific node, data are already available;
- all slave nodes exchange the same amount ($D_{\mathrm{p}}$) of process data;
- there is only one FMMU shared by all slave nodes.

The cycle time $\Delta T_{ecat}$ for an EtherCAT based system using FMMUs can be evaluated as follows:

$$\Delta T_{ecat} = \Delta T_{net} + \Delta T_{eth\_ov} + \Delta T_{ecat\_ov} + \Delta T_{\mathrm{p}}, \tag{1}$$

where $\Delta T_{net}$ is the delay introduced by the cables and the interface hardware, $\Delta T_{eth\_ov}$ and $\Delta T_{ecat\_ov}$ are, respectively, the Ethernet and the ECAT temporal overhead, whereas $\Delta T_{\mathrm{p}}$ is the time needed to transmit process data. $\Delta T_{net}$ can be evaluated as:

$$\Delta T_{net} = n_s \times \Delta T_{ecat\_fwd} + n_c \times \Delta T_m, \tag{2}$$

where $n_s$ is the number of slave nodes, $\Delta T_{ecat\_fwd}$ is the time necessary, for each slave node, to forward the received frame from the input to the output Ethernet interface (estimated to be up to 1.35 $\mu sec$ in non-redundant configurations, and 0.675 $\mu sec$ in redundant configurations), $n_c$ is the number of cables between slave nodes in the whole network (it equals to $2 \times n_s$ for an "open ring" topology, and to $n_s + 1$ for a "closed ring" topology), and $\Delta T_m$ is the delay due to the propagation in the medium (it is approximately 0.0005 $\mu sec$ for a 0.01 *m* cable). Therefore, according to the topology, (2) becomes:

$$\Delta T_{net} = \begin{cases} n_s \times (\Delta T_{ecat\_fwd} + 2 \times \Delta T_m) \\ n_s \times (\Delta T_{ecat\_fwd} + \Delta T_m) + \Delta T_m \end{cases}, \tag{3}$$

respectively for "open ring" and "closed ring" topologies. In (1), $\Delta T_{eth\_ov}$ can be evaluated as:

$$\Delta T_{eth\_ov} = n_{fr} \times \Delta T_{eth} + \Delta T_{pad}, \tag{4}$$

where $\Delta T_{eth}$ is the time needed to send all the data due to the Ethernet frame overhead, $n_{fr}$ is the number of Ethernet frames needed to send all the data to all the slave nodes, and $\Delta T_{\mathtt{pad}}$ is the time necessary to send the possible padding bytes required to have an Ethernet frame with the minimal standard length. $\Delta T_{eth}$ can be evaluated as:

$$\Delta T_{eth} = \frac{(D_{eth} + D_{\mathtt{IFG}})}{b}, \tag{5}$$

where $D_{eth}$ is the sum of the Ethernet header and the trailer size (i.e., 26 $B$ made up of 8 $B$ preamble, 6 $B$ Destination Mac Address, 6 $B$ Source Mac Address, 2 $B$ EtherType and 4 $B$ CRC – Figure 4), $D_{\mathtt{IFG}}$ is the 12 $B$ InterFrame Gap size, and $b$ is the byte rate (in the considered scenario it amounts to 12.5 $B/\mu sec$ = 100 $Mb/sec$).

In (1), $\Delta T_{ecat\_ov}$ can be estimated as:

$$\Delta T_{ecat\_ov} = \frac{n_{fr} \times (D_{ecat\_h} + D_{ecat\_th} + D_{ecat\_wc})}{b}, \tag{6}$$

where $D_{ecat\_h}$ is the 2 $B$ ECAT header size, $D_{ecat\_th}$ is the 10 $B$ header size for each telegram and $D_{ecat\_wc}$ is the 2 $B$ working counter size for each telegram.

In (1), $\Delta T_{\mathtt{p}}$ is the time required to send all the data exchanged between the master and the slave nodes:

$$\Delta T_{\mathtt{p}} = \frac{D_{\mathtt{p}}}{b}, \tag{7}$$

where $D_{\mathtt{p}}$ is the amount of data sent by each slave.

In (4) and (6), $n_{fxr}$ can be evaluated as:

$$n_{fr} = \left\lceil \frac{n_s}{max_t} \right\rceil, \tag{8}$$

where $max_t$ is given by:

$$max_t = \left\lfloor \frac{D_{eth\_mp} - D_{ecat\_h} - D_{ecat\_th} - D_{ecat\_wc}}{D_{\mathtt{p}}} \right\rfloor. \tag{9}$$

In (9), $D_{eth\_mp}$ is the 1500 $B$ maximum size of an Ethernet frame payload.

In (4) we must still evaluate the term $\Delta T_{\mathtt{pad}}$, which represents the time necessary to send the possible padding bytes required to have a standard Ethernet frame. The amounts of data sums up to:

$$D_{ecat} = D_{ecat\_h} + (D_{ecat\_th} + D_{ecat\_wc} + D_p) \times (n \mod max_t). \tag{10}$$

If we define $D_{eth\_mpl}$ as the minimum size of the Ethernet payload (i.e., 46 $B$ from specifications), $D_{ecat} < D_{eth\_mpl}$ as *condition* $c_1$ and $n \mod max_t \neq 0$ as *condition* $c_2$, then $\Delta T_{\mathtt{pad}}$ is given as follows:

$$\Delta T_{\mathtt{pad}} = \begin{cases} \frac{D_{eth\_mpl} - D_{ecat}}{b} & \text{if } c_1 \text{ and } c_2 \text{ hold,} \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

Table I, Table II and the corresponding Figure 5 and Figure 6 present numerical results when applying the mathematical model varying the number of slave nodes in the network from 10 to 200 and the size of the exchanged data from 2 to 256 $B$. In accordance with the results reported in [10], the trend is linear in the number of slave nodes, which makes the network easily scalable.

TABLE I

CYCLE TIME - NON-REDUNDANT CONFIGURATION [$\mu sec$]

| | Bytes | | | |
|---|---|---|---|---|
| **Nodes** | **2** | **32** | **128** | **256** |
| **10** | 23,75 | 43,27 | 120,07 | 226,63 |
| **20** | 34,38 | 82,38 | 240,14 | 453,26 |
| **30** | 49,49 | 121,49 | 360,21 | 679,89 |
| **40** | 64,60 | 160,60 | 480,28 | 906,52 |
| **50** | 79,71 | 203,87 | 600,35 | 1133,15 |
| **60** | 94,82 | 242,98 | 720,42 | 1359,78 |
| **70** | 109,93 | 282,09 | 840,49 | 1586,41 |
| **80** | 125,04 | 321,20 | 960,56 | 1813,04 |
| **90** | 140,15 | 360,31 | 1080,63 | 2039,67 |
| **100** | 155,26 | 403,58 | 1200,70 | 2266,30 |
| **110** | 170,37 | 442,69 | 1316,61 | 2492,93 |
| **120** | 185,48 | 481,80 | 1436,68 | 2719,56 |
| **130** | 200,59 | 520,91 | 1556,75 | 2946,19 |
| **140** | 215,70 | 564,18 | 1676,82 | 3172,82 |
| **150** | 230,81 | 603,29 | 1796,89 | 3399,45 |
| **160** | 245,92 | 642,40 | 1916,96 | 3626,08 |
| **170** | 261,03 | 681,51 | 2037,03 | 3852,71 |
| **180** | 276,14 | 720,62 | 2157,10 | 4079,34 |
| **190** | 291,25 | 763,89 | 2277,17 | 4305,97 |
| **200** | 306,36 | 803,00 | 2397,24 | 4532,60 |

TABLE II

CYCLE TIME - REDUNDANT CONFIGURATION [$\mu sec$]

| | Bytes | | | |
|---|---|---|---|---|
| **Nodes** | **2** | **32** | **128** | **256** |
| **10** | 17,00 | 36,52 | 113,32 | 219,88 |
| **20** | 20,87 | 68,87 | 226,63 | 439,75 |
| **30** | 29,23 | 101,23 | 339,95 | 659,63 |
| **40** | 37,58 | 133,58 | 453,26 | 879,50 |
| **50** | 45,94 | 170,10 | 566,58 | 1099,38 |
| **60** | 54,29 | 202,45 | 679,89 | 1319,25 |
| **70** | 62,65 | 234,81 | 793,21 | 1539,13 |
| **80** | 71,00 | 267,16 | 906,52 | 1759,00 |
| **90** | 79,36 | 299,52 | 1019,84 | 1978,88 |
| **100** | 87,71 | 336,03 | 1133,15 | 2198,75 |
| **110** | 96,07 | 368,39 | 1242,31 | 2418,63 |
| **120** | 104,42 | 400,74 | 1355,62 | 2638,50 |
| **130** | 112,78 | 433,10 | 1468,94 | 2858,38 |
| **140** | 121,13 | 469,61 | 1582,25 | 3078,25 |
| **150** | 129,49 | 501,97 | 1695,57 | 3298,13 |
| **160** | 137,84 | 534,32 | 1808,88 | 3518,00 |
| **170** | 146,20 | 566,68 | 1922,20 | 3737,88 |
| **180** | 154,55 | 599,03 | 2035,51 | 3957,75 |
| **190** | 162,91 | 635,55 | 2148,83 | 4177,63 |
| **200** | 171,26 | 667,90 | 2262,14 | 4397,50 |

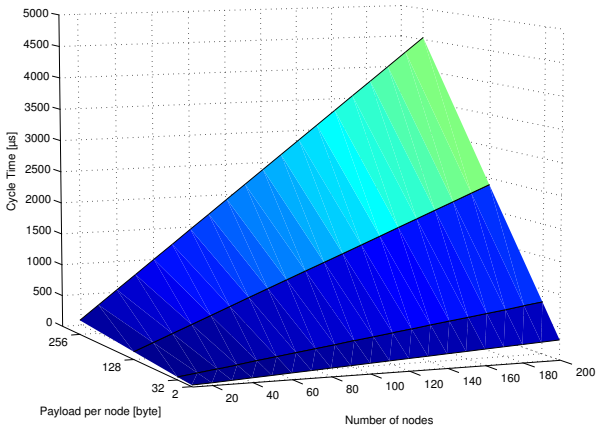


Fig. 5. EtherCAT Cycle Time - Non-redundant configuration.

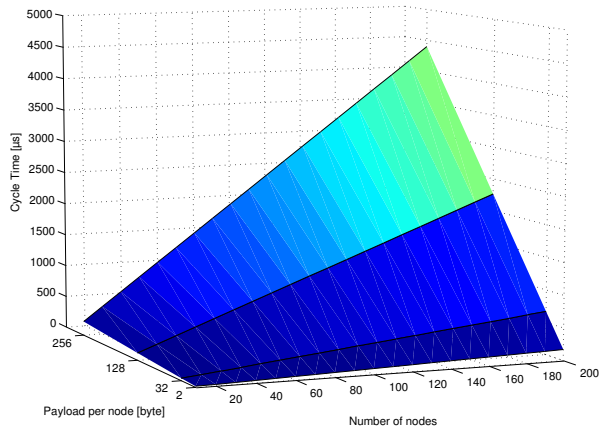


Fig. 6. EtherCAT Cycle Time - Redundant configuration.

Figure 7 shows a comparison between cycle times data obtained in a redundant network: solid lines refer to a network using FMMUs, whereas dashed lines represents a network without FMMUs. For both the cases, red lines corresponds to a data exchange of 128 *B* per slave node and blue lines to 2 *B* per node. The data for the case without FMMUs are obtained from our previous model [10]. As it can be seen in Figure 7, using the FMMUs the cycle time is always reduced, due to the reduced overhead. The two models give the upper and lower boundaries for the cycle time: when the slave nodes are configured to be associated with only one FMMU, this is the lower boundary; when there are no FMMUs, and each slave node is queried with a different ECAT telegram, this is the the upper boundary. All the other possible solutions, where slaves are mapped to different FMMUs, lie between the two boundaries.

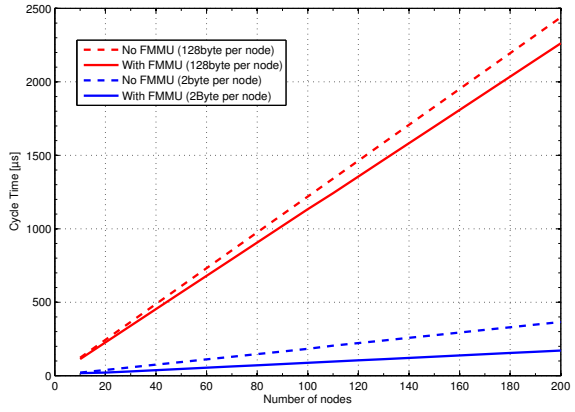


Fig. 7. EtherCAT Cycle Time - With and without FMMU comparison (redundant configuration).

### C. Master Node

The master node is responsible for two important tasks: (i) to manage the overall EtherCAT network, with respect to timing properties as well as off-line and on-line configuration; (ii) to provide higher architectural layers with tactile data to process [14].

On the one hand, in the current implementation, to guarantee proper timing performance, a standard embedded real-time operating system has been adopted. On the other hand, a special attention has been devoted to design specific tools to automatically configure and manage the network. These tools are able, at start-up, to scan the whole network, to identify the skin slave devices in the network, and to properly configure the master as well as its data structures to gather tactile information. Their description is out of the scope of this article.

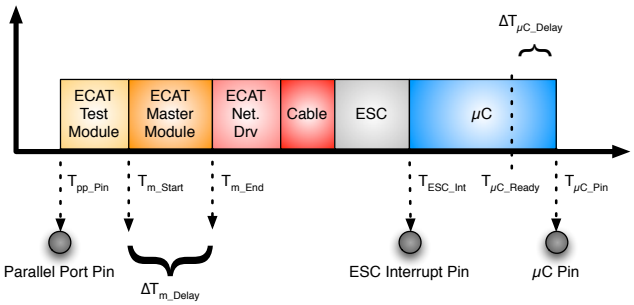


Fig. 8. Experimental test: timing.

## III. Model Validation and Performance Tests

### A. Performance Tests Methodology

The goal of this series of tests is to evaluate the overall *delay* and *jitter* introduced by the tactile sensing system. To this aim, we separately tested the delay introduced by different system components. As anticipated, this is of utmost importance for robots involved in physical human-robot or environment-robot interaction tasks, since it impacts over the global robot response time to contacts.

The experimental test-bed is made-up of three main components: (i) a master node based on a standard embedded PC with an Intel Pentium 4@2.40 *GHz* and 2 *GB* dual-channel DDR2 RAM running Linux, RTAI 3.8.1 and the IgH EtherCAT master module 1.5; (ii) a slave composed by a Beckhoff FB1111-0141 board and a Texas Instruments ARM Cortex-M3 microcontroller interconnected using an SPI bus; (iii) an oscilloscope able to perform statistical analysis.

Figure 8 depicts the sequence of all the delays that need to be evaluated. It is worth noticing that the actual sequence of delays we measure is the opposite on what append when acquiring tactile data. However this is not an issue since al the delays related to the EtherCAT bus are symmetric. Using the oscilloscope we measured the delay $\Delta T_{pins}$ between the rise of a microcontroller pin on the slave node (at time $T_{\mu C\_Pin}$) and the rise of a parallel port pin on the master node (at time $T_{pp\_Pin}$). This delay corresponds to:

$$\Delta T_{pins} = T_{\mu C\_Pin} - T_{pp\_Pin}, \tag{12}$$

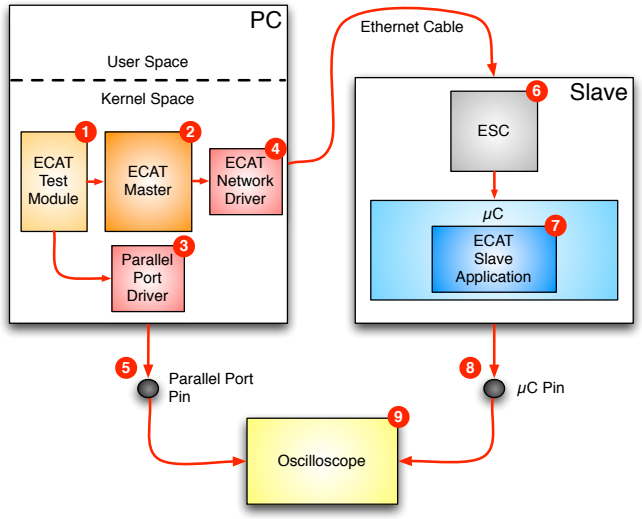


Fig. 9. Experimental test: architecture.

but the real delay introduced by the system is $\Delta T_{delay}$, which corresponds to:

$$\begin{aligned} \Delta T_{delay} &= T_{\mu C\_Ready} - T_{pp\_Pin} \\ &= \Delta T_{pins} - \Delta T_{\mu C\_Delay} - T_{pp\_Pin}, \end{aligned} \quad (13)$$

where $\Delta T_{\mu C\_Delay}$ is the delay due to the microcontroller for rising or lowering a pin:

$$\Delta T_{\mu C\_Delay} = T_{\mu C\_Pin} - T_{\mu C\_Ready}. \quad (14)$$

To evaluate (13), we generate on the master node a square wave at different frequencies that is sent to the slave node. Both the nodes raise a pin as soon as – respectively – generate and receive the square wave. In the master node, a pin of the parallel port is raised. Both the signals on the pins are then analysed using an oscilloscope.

In Figure 9 a detailed view of this test is shown. An RTAI kernel module (1) generates a square wave at 1 $kHz$, setting a variable to 0 and 1 alternatively. The same module (1) outputs this value on a pin of the PC parallel port (5), and transmit the value to the EtherCAT master module (2). Instead of using a parallel port driver (3), which could introduce further latencies, we access directly the memory address of the parallel port. The EtherCAT master module (2) transmits this data to the dedicated EtherCAT device driver (4). The data sent in the bus is received by the ESC chip (6) of the slave device and presented in the memory of the microcontroller (7) by a task invoked by an interrupt. Then, the microcontroller drives a pin (8) corresponding to the value sent by the master node. Using the oscilloscope (9), the delay between the rising edges of the two signals is measured.

With reference to Figure 8, we performed experiments also to evaluate $T_{ESC\_Int}$. This is the time instant corresponding to the availability of a new data for the microcontroller to access. When the data arrives at a slave node an SPI interrupt is generated, which triggers a falling edge on a pin of the Beckhoff ESC chip. Therefore, using the technique described above, we can measure the delay $\Delta T_{pp\_Pin \div ESC\_Int}$ between the triggers on the PC parallel port and the SPI interrupt, as follows:

$$\Delta t_{pp\_Pin \div ESC\_Int} = T_{ESC\_Int} - T_{pp\_Pin}, \quad (15)$$

which corresponds to the latency between the the data send instruction and the data arrival time on the ESC chip. Analogously, we can measure the delay $\Delta T_{ESC\_Int \div \mu C\_Pin}$ between the SPI interrupt and the trigger on the microcontroller pin, as follows:

$$\Delta T_{ESC\_Int \div \mu C\_Pin} = T_{\mu C\_Pin} - T_{ESC\_Int}, \quad (16)$$

which is the latency between the availability of the data on the ESC chip and on the microcontroller, respectively.

Finally, it is also possible to measure the delay $\Delta T_{md}$ introduced by the EtherCAT master function calls. It can be evaluated as follows:

$$\Delta T_{md} = \Delta T_{mr} + \Delta T_{mp} + \Delta T_{mq} + \Delta T_{ms}, \quad (17)$$

where $\Delta T_{mr}$, $\Delta T_{mp}$, $\Delta T_{mq}$ and $\Delta T_{ms}$ can be obtained statistically computing the time needed to perform a number of corresponding blocking function calls of the EtherCAT master module.

### B. Performance Tests Results

The results of the measurement of the delay between the two pins (i.e., parallel port and microcontroller pin), namely $\Delta T_{pins}$, and the measurement of the delay due to the microcontroller, namely $\Delta T_{\mu C\_Delay}$, are shown in Table III. Data are obtained using $15 \times 10^6$ samples with a 1 $msec$ cycle time. Using these results and (13), we obtain $\Delta T_{Delay} = 26.18$ $\mu sec$.

TABLE III
MEASURED DELAYS.

| | Latency [$\mu sec$] | | | |
|---|---|---|---|---|
| | mean | min. | max. | std. dev. |
| $\Delta T_{Pins}$ | 26.580 | 24.700 | 32.700 | 1.410 |
| $\Delta T_{\mu C\_Delay}$ | 0.398 | 0.397 | 0.399 | 0.0003 |
| $\Delta T_{pp\_Pin \div ESC\_Int}$ | 9.661 | 9.303 | 14.276 | 0.199 |
| $\Delta T_{ESC\_Int \div \mu C\_Pin}$ | 16.664 | 15.082 | 19.579 | 1.405 |

In Table III, the measurements of, respectively, $\Delta T_{pp\_Pin \div ESC\_Int}$ and $\Delta T_{ESC\_Int \div \mu C\_Pin}$ are reported. Analogously to the previous tests, data are obtained using $15 \times 10^6$ samples with a 1 $ms$ cycle time. It is important to remark that this result shows that the bigger part of the delay is due to the microcontroller.

Network latencies, when multiple slaves are added in between the master node and the test slave device, have been assessed. The raw latencies $\Delta T_{Pins}$ are presented in Table IV. In the table, the last column shows real delays $\Delta T_{delay}$. Again, data are obtained using $15 \times 10^6$ samples with a 1 $ms$ cycle time. In Figure 10, $\Delta T_{delay}$ values for all the configurations are shown.

TABLE IV

MEASUREMENTS OF DELAY WITH MULTIPLE SLAVES.

| Slaves | $\Delta T_{Pins}$ [$\mu$s] | | | | $\Delta T_{delay}$ [$\mu$s] |
|---|---|---|---|---|---|
| | mean | min. | max. | std. dev. | mean |
| 1 + 1 | 26.97 | 25.50 | 33.99 | 1.43 | 26.57 |
| 1 + 2 | 27.31 | 25.43 | 33.42 | 1.42 | 26.91 |
| 1 + 3 | 27.51 | 25.59 | 34.00 | 1.43 | 27.11 |
| 1 + 4 | 27.82 | 25.90 | 34.81 | 1.42 | 27.42 |
| 1 + 8 | 28.32 | 26.40 | 34.00 | 1.43 | 27.92 |

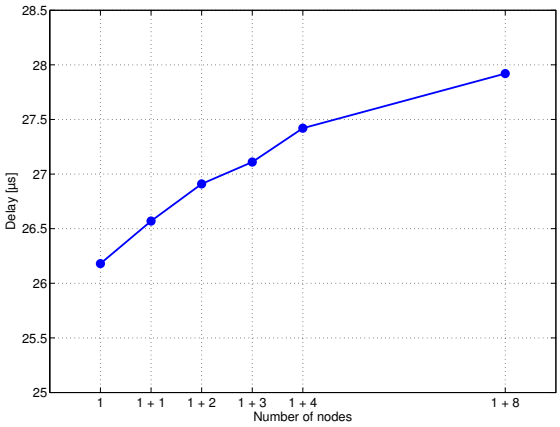


Fig. 10. $\Delta T_{delay}$ versus number of nodes.

The RTAI time log feature has been used to measure the execution time of the EtherCAT master send and receive functions. The test has been performed to monitor the execution time of the four master functions necessary for data transfer (Table V). Using these results and (17), we obtain $\Delta T_{md} = 10.216\ \mu sec$. The data are obtained using $3.5 \times 10^4$ samples with a 1 *ms* cycle time. It is important to remark that the software used to perform the pin-to-pin delay measurement does not use two of these functions, namely `ecrt_master_receive()` and `ecrt_domain_process()`. As a result, the pin-to-pin measurements produce lower delays.

TABLE V

ETHERCAT MASTER FUNCTIONS EXECUTION TIMES.

| | Exec. time [$\mu$s] | | | |
|---|---|---|---|---|
| | mean | min. | max. | std. dev. |
| $\Delta T_{mr}$ | 7.448 | 6.897 | 13.324 | 0.642 |
| $\Delta T_{mp}$ | 0.125 | 0.098 | 2.913 | 0.091 |
| $\Delta T_{mq}$ | 0.157 | 0.092 | 0.812 | 0.099 |
| $\Delta T_{ms}$ | 2.486 | 1.973 | 7.175 | 0.634 |

## IV. CONCLUSION

This article discusses a real-time, EtherCAT-based and distributed architecture for large-scale tactile sensing. In particular, the different components of the overall system have been introduced, with a particular emphasis on performance assessment and evaluation. Although a deeper characterization of the architecture is subject of on-going work, preliminary results show that EtherCAT is a promising solution for such *deeply* embedded systems as humanoid robots. This confirms results from previous work [10]: from a real-time perspective, timing results match with typical prerequisites for implementing robot control loops based on tactile feedback; from an architectural point of view, this solution is sufficiently flexible to adapt to different robot configurations.

## REFERENCES


[1] Y. Ohmura, Y. Kunyoshi, and A. Nagakubo, "Conformable and scalable tactile sensor skin for curved surfaces," in *Proceedings of the 2006 IEEE International Conference on Robotics and Automation (ICRA 2006)*, Orlando, FL, USA, May 2006.

[2] T. Tajika, T. Miyashita, and H. Ishiguro, "Automatic categorization of haptic interactions - what are the typical haptic interactions between a human and a robot?" in *Proceedings of the 2006 IEEE-RAS International Conference on Humanoid Robots (HUMANOIDS 2006)*, Genova, Italy, December 2006.

[3] I. Mizuuchi, T. Yoshikai, T. Nishino, and M. Inaba, "Development of the musculoskeletal humanoid kotaro," in *Proceedings of the 2007 IEEE International Conference on Robotics and Automation (ICRA 2006)*, Orlando, FL, USA, May 2006.

[4] T. Minato, Y. Yoshikawa, H. Ishiguro, and M. Asada, "CB2: A child robot with biomimetic body for cognitive developmental robotics," in *Proceedings of the 2007 IEEE-RAS International Conference on Humanoid Robots (HUMANOIDS 2007)*, Pittsburg, PN, USA, December 2007.

[5] T. Mukai, M. Onishi, S. Hirano, and Z. Luo, "Development of the tactile sensor system of a human-interactive Robot RI-MAN," *IEEE Transactions on Robotics*, vol. 24, no. 2, pp. 505–512, April 2008.

[6] G. Cannata, M. Maggiali, G. Metta, and G. Sandini, "An embedded artificial skin for humanoid robots," in *Proceedings of the 2008 IEEE International Conference on Multi-sensor Fusion and Integration (MFI 2008)*, Seoul, Korea, August 2008.

[7] A. Schmitz, P. Maiolino, M. Maggiali, L. Natale, G. Cannata, and G. Metta, "Methods and technologies for the implementation of large-scale robot tactile sensors," *IEEE Transactions on Robotics*, vol. 27, no. 3, pp. 389–400, June 2011.

[8] P. Mittendorfer and G. Cheng, "Humanoid multi-modal tactile sensing modules," *IEEE Transactions on Robotics*, vol. 27, no. 3, pp. 401–410, 2011.

[9] D. Anghinolfi, G. Cannata, F. Mastrogiovanni, C. Nattero, and M. Paolucci, "On the problem of the automated design of large-scale robot skin," *IEEE Transactions on Automation Science and Engineering*, vol. 10, no. 4, pp. 1087–1100, 2013.

[10] E. Baglini, G. Cannata, and F. Mastrogiovanni, "Design of an embedded networking infrastructure for whole-body tactile sensing in humanoid robots," in *Proceedings of the 2010 IEEE-RAS International Conference on Humanoid Robots (HUMANOIDS 2010)*, Nashville, TN, USA, December 2010.

[11] M. Cereia, I. Bertolotti, and S. Scanzio, "Performance evaluation of an EtherCAT master using Linux and the RT Patch," in *Proceedings of the 2010 International Symposium on Industrial Electronics (ISIE 2010)*, Bari, Italy, July 2010.

[12] L. Wang, M. Li, J. Wang, and Q. Zhang, "The design and performance analysis for real-time EtherCAT network data acquisition system," in *Proceedings of the 2010 International Conference on Intelligent Control and Information Processing (ICICIP 2010)*, Dalian, China, August 2010.

[13] S. Vitturi, L. Peretti, L. Seno, M. Zigliotto, and C. Zunino, "Real-time Ethernet networks for motion control," *Computer standards & interfaces*, vol. 33, no. 5, pp. 465–476, 2011.

[14] L. Muscari, L. Seminara, F. Mastrogiovanni, M. Valle, M. Capurro, and G. Cannata, "Real-time reconstruction of contact shapes for large area robot skin," in *Proceedings of the 2013 IEEE International Conference on Robotics and Automation (ICRA 2013)*, Karlsruhe, Germany, May 2013.