Time-Critical Wireless Networked Embedded Systems: Feasibility and Experimental Assessment

Francesco Branz, Riccardo Antonello, Luca Schenato, *Fellow, IEEE*, Federico Tramarin, *Member, IEEE*, and Stefano Vitturi, *Senior Member, IEEE*

***Abstract*—In this article, we investigate an innovative solution, to implement high sampling frequency industrial control by means of networked embedded systems connected via WiFi. The basic idea relies on a co-design approach for the control application, which is then able to adapt its sampling period, as well as to tune the Wi-Fi parameters, according to the feedback coming from the network. To this end, we implemented a cross-layer architecture acting at both application and data-link layers, which features a robust frame-delay state estimator, a time-efficient communication policy, and a specific tuning of the critical protocol parameters. Suitable hardware-in-the-loop experiments have been carried out exploiting two different embedded systems available off-the-shelf. The preliminary results, obtained from an extensive experimental campaign, are encouraging since they show that the proposed architecture enables industrial control applications requiring a sampling rate up to 1000 Hz, even in presence of communication impairments.**

***Index Terms*—Kalman filters, networked control systems, wireless networks.**

I. Introduction

NETWORKED embedded systems for industrial control, that rely on wireless networks support, provide several advantages over those based on wired counterparts, like improved flexibility and mobility, reduced maintenance costs, and configuration efforts. Unfortunately, general-purpose and best effort wireless networks, like the IEEE 802.11 Wireless LANs widespread in home/office scenarios, revealed to be not always able to deal with demanding applications, due to their intrinsic unreliability and nondeterminism. This has triggered the design of new, better performing, protocols and/or the modifications and tuning of the existing ones, to cope with the specific application requirements.

Networked embedded systems find a straightforward field of application in cyber–physical systems, that are characterized by an ever-growing interest, where several collaborating computational devices have to handle time, safety, or mission-critical physical entities or processes [1]. This has become even more evident with the efforts toward Industry 4.0 and the rise of the industrial Internet of Things (IIoT) paradigm in smart manufacturing systems, to improve productivity, efficiency, safety, and intelligence [2], [3].

In the above scenario, the advent of low-cost networked embedded systems with steadily increasing computational power and pervasive wireless connectivity has allowed to tackle new industrial applications, such as the edge and fog computing [4], characterized by a tight cooperation among lightweight computing nodes. Within smart factory domains, many control tasks are still critical, both in terms of timeliness and reliability, having stringent computation and networking requirements to ensure the adequate performance level. These results are even more demanding in case applications have to be accomplished by networked computational nodes, possibly distributed within the industrial site. Legacy wireless network protocols may result inadequate for such applications, as pointed out by several analyses and practical experiments, due to their intrinsic unreliability and the lack of determinism, that may compromise performance indicators such as end-to-end latency, network-wide synchronization, and high-speed information delivery.

A. Essential Related Work

To address the aforementioned issues, several industrial wireless protocols have been proposed, based on the IEEE 802.15.4 and 802.15.1 physical layers, to support the required performance figures also, and especially, for the case of networked embedded systems. As an example, the widespread WirelessHart [5], ISA100.11a [6], 6TiSCH [7], and WISA [8] are all

Manuscript received August 4, 2019; revised December 30, 2019 and March 9, 2020; accepted April 19, 2020. Date of publication May 7, 2020; date of current version September 18, 2020. This work was supported in part by the University of Padova under Projects MAgIC SCHE_SID17_01 and INTERACT TRAM_SID19_01. Paper no. TII-19-3612. *(Corresponding author: Francesco Branz.)*

Francesco Branz is with the Department of Industrial Engineering, University of Padova, 35131 Padova, Italy (e-mail: francesco.branz@unipd.it).

Riccardo Antonello and Luca Schenato are with the Department of Information Engineering, University of Padova, 35131 Padova, Italy (e-mail: riccardo.antonello@unipd.it; schenato@dei.unipd.it).

Federico Tramarin is with the Department of Engineering, University of Modena e Reggio Emilia, 41125 Modena, Italy (e-mail: tramarin@ieee.org).

Stefano Vitturi is with the Institute of Electronics and Computer and Telecommunications, National Research Council of Italy (CNR-IEIIT), 35131 Padova, Italy (e-mail: stefano.vitturi@ieiit.cnr.it).

Color versions of one or more of the figures in this article are available online at https://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2020.2992990

able to support real-time control applications, thanks to the adoption of purposely designed time-slotted media access policies (i.e., TDMA) and suitable transmission scheduling. However, although they are found effective typically in process automation applications with modest sampling frequencies [9]–[12], the intrinsic limitations of the underlying physical layers make them unsuitable for the most demanding and critical scenarios.

In this perspective, the widespread IEEE 802.11-based Wi-Fi networks [13] are expected to be capable of sustaining very high data-rates by design, theoretically allowing to tame high-bandwidth control applications with sampling frequency in the order of 1 kHz or more. Meaningful examples, in this respect, are force/current control applications such as those described in [14] and [15]. Moreover, WiFi networks are widespread, and currently available even for low-cost and low-resources embedded systems. For these networks, however, shortcomings come from the legacy channel access and frame delivery policies established by the IEEE 802.11 specifications, that rely on a distributed randomized algorithm [16], which inherently undermine the implementation of time-critical applications.

The Wi-Fi behavior has been comprehensively analyzed, and the protocol potentialities to support demanding industrial control applications can be found in [17]–[19]. For example, a low-frequency control application exploiting Wi-Fi is reported in [20]. The scientific literature also provides suggestions to overcome the unbounded delays and data losses experienced with conventional Wi-Fi architectures [21]–[23]. Indeed, several contributions have been developed to provide high reliability and low latency over Wi-Fi, which include, for instance, the EU-funded project FlexWare [24]–[26], the recent WIA-FA standard [27], and also the commercial solution "industrial WLAN" (iWLAN) [28], [29].

More recently, the development of innovative rate selection algorithms [30] has provided significant improvements to the Wi–Fi performance, allowing to cope with variations of the wireless link quality to maximize timeliness and reliability, while strictly considering the deadline imposed by real-time tasks. A further significant contribution is given by [31], where the authors propose a new data-link layer protocol, based on a TDMA technique conceived to meet demanding real-time constraints, with an improved determinism in channel access and data transfer. Unfortunately, this solution, although appealing, is difficult to implement for commercially available embedded systems, since these latter ones, usually, do not allow to modify their protocol stacks. Moreover, the adoption of a new data-link layer protocol on such devices could negatively impact on their computational capabilities.

II. Contribution

The solutions described above imply significant tuning of the lower protocol levels and/or the design of new data-link layers to amend the legacy Wi-Fi standard, thus obtaining bounded latency and low packet delivery time, as imposed by the typical industrial applications. Indeed, many tasks executed by industrial controllers rely on cyclic schedules, based on constant static periods, regardless of data delivery and network protocols constraints.

This article, conversely, exploits an innovative approach somehow complementary to the aforementioned ones, based on co-design methods, as theoretically presented and formally described in [32]. The basic idea is to dynamically adapt both the application sampling period and the main Wi-Fi parameters, on the basis of the feedback received from the channel. The proposed architecture clearly relies on the estimation of the plant behavior by means of a digital twin approach, where model-predictive control techniques are adopted [33], [34]. The still necessary real-time behavior is hence dealt with by a joint effort of the cooperating system components. This issue has been initially addressed by some recent research works, as for instance in [35] where, however, the problem has only been roughly outlined by the authors.

This article presents a feasibility study based on the implementation and performance assessment of such a co-design approach. As a meaningful application example of networked systems, we address the real-time control of the balance of a two-wheeled robot over Wi-Fi. This goal is achieved by leveraging on the capabilities offered by off-the-shelf available embedded computing systems, with built-in Wi-Fi interfaces.

The industrial application is developed and implemented by exploiting modern rapid control prototyping techniques on two low-cost, widespread, embedded systems, namely Raspberry Pi 3 mod. B and BeagleBone Black Wireless.

Experimental assessments are carried out by employing hardware-in-the-loop techniques, where the plant is controlled by an embedded system and the connection with the simulated plant is achieved by means of a real wireless communication link.

The rest of this article is organized as follows. Section III briefly outlines the multiagent robotic system which represents the benchmark application. Section IV addresses the cross-layer architecture we implemented, referring to both the control system and the communication application. Section V introduces the implementation strategies and describes the conducted tests. Section VI deals with the experimental assessment and reports the obtained results. Also, this section provides a critical analysis of the results and discusses some future perspectives. Finally, Section VII concludes the article.

III. Benchmark Application

We take into account the representative case of multiagent robotic systems in complex environments controlled over wireless links. Exploiting embedded boards in these systems is a very appealing solution for the potential compactness, modularity, and resource efficiency. The considered benchmark application is the balance control of a two-wheeled segway-like vehicle [balancing robot; Fig. 1(a)] which can be seen as a meaningful testbed to verify the feasibility of the proposed cross-layer architecture for control at high sampling rates of networked embedded systems over Wi-Fi. This choice is motivated by the fact that keeping the correct upward vertical balance of this, naturally unstable vehicle requires fast and continuous adjustments of wheels positions, according to the observed state. A reliable and very low-latency communication between the controller and the

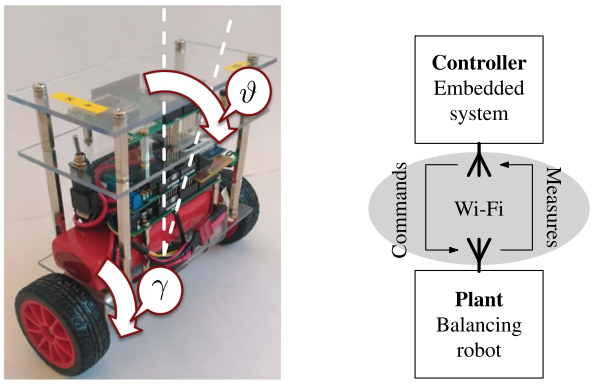


Fig. 1. Benchmark application. (a) Balancing robot prototype. (b) Control system architecture.

robot is hence required to successfully accomplish the control task.

As shown in Fig. 1(b), the controller is connected to the balancing robot via a Wi-Fi network. It is assumed that all the "intelligence" runs on the controller which exchanges process data, such as motor commands and sensor measurements, with the robot over Wi-Fi. This application represents an effective benchmark for the proposed co-design approach and the designed cross-layer implementation, since it makes the underlying Wi-Fi network crucial for any robot action.

The final control goal is the positioning of the robot base towards a specified set-point, while maintaining the robot balance. It is assumed that the robot only moves along a straight path, so that the dynamics is constrained to its sagittal plane. In this way, the robot can be analyzed as a planar one, with generalized coordinates corresponding to the wheel angle, γ (proportional to the robot base position, under no wheel slip conditions), and the robot body inclination, ϑ, with respect to the vertical position. The wheel angle is measured by the magnetic encoders connected to the shaft of the two dc gearmotors installed on the robot. The inclination can be instead inferred from the inertial measurements provided by a micro electro-mechanical systems (MEMS) accelerometer/gyroscope pair, with the aid of a sensor fusion algorithm (e.g., complementary filtering) running locally on the embedded system installed onboard the robot. The dc gearmotors are voltage-controlled with a pulse-width modulation (PWM) driver; the PWM duty-cycle commands are received from the host computer.

IV. Cross-Layer Architecture

The following subsections will provide details about the proposed co-design architecture addressing, specifically, both the control system and the communication applications.

A. Control System

The overall control system is organized around the architecture illustrated in Fig. 1(b). We adopted the algorithm proposed in [32], with the aim of minimizing the effects of communication nonidealities on the control application side. The controller implements both the balance and position algorithms that estimate the system state through a modified Kalman filter, exploiting measurement buffering to realize a static feedback [36]. For the design of both the controller and the estimator, the continuous-time nonlinear dynamics of the robot is first linearized around the upward vertical equilibrium, and then discretized with the exact discretization method with a sampling time $T_s = 1$ ms. The resulting linear model is of the type

$$\begin{cases} x_{k+1} = Ax_k + Bu_k + w_k \\ y_k = Cx_k + v_k \end{cases} \tag{1}$$

where $x_k = [\gamma_k, \vartheta_k, \dot{\gamma}_k, \dot{\vartheta}_k]^T \in \mathbb{R}^4$ is the robot state, $u_k \in \mathbb{R}$ are the motor commands (PWM duty-cycle values, identical for both motors), and y_k are the measurements received from the plant. Since the robot state is completely observable from the wheel angle measurement γ, in the following, it will be assumed that y_k coincides with that quantity. Compared to a solution involving a direct measurement of the whole robot state (by exploiting also the onboard inertial sensors), this design choice has the appreciable advantage of minimizing the amount of information exchanged between the plant and the controller over Wi-Fi. The symbols w_k and v_k denote two gaussian noises that account for, respectively, modeling and measurement errors.

Due to the stochastic behavior of the Wi-Fi network, the measurements received by the controller can be affected by a random delay, or they can even be lost. Therefore, the arrival process can be modeled by the random variable δ_t^k:

$$\delta_t^k = \begin{cases} 1 & \text{if } y_t \text{ is available at estimator at } k > t \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

whose distribution can be chosen, depending on the situation, according to different models [37]–[40]. Based on the arrival sequence, it is possible to define the measurement model at the estimator on the host side as

$$y_t^k = \delta_t^k y_t. \tag{3}$$

With these positions, we may state that the information set available at the estimator at the time instant k is

$$\mathcal{I}^k = \left\{\delta_0^k, \delta_1^k, \ldots, \delta_{k-1}^k, y_0^k, y_1^k, \ldots, y_{k-1}^k, u_0, u_1, \ldots, u_{k-1}\right\}. \tag{4}$$

Within this framework, it has been shown in [36] that the optimal state estimate $\widehat{x}_{k|k-1}^k$ given $\mathcal{I}^k$ can be computed with the following iterative scheme:

$$\widehat{x}_{t|t-1}^k = A\widehat{x}_{t-1|t-1}^k + Bu_{t-1} \tag{5}$$

$$\widehat{x}_{t|t}^k = \widehat{x}_{t|t-1}^k + \delta_t^k L_t(y_t^k - C\widehat{x}_{t|t-1}^k) \tag{6}$$

$$L_t = P_{t|t-1}^k C'(CP_{t|t-1}^k C' + R)^{-1} \tag{7}$$

$$P_{t|t-1}^k = AP_{t-1|t-1}^k A' + Q \tag{8}$$

$$P_{t|t}^k = P_{t|t-1}^k - \delta_t^k P_{t|t-1}^k C'(CP_{t|t-1}^k C' + R)^{-1} CP_{t|t-1}^k \tag{9}$$

where Q and R denote the covariance matrices of, respectively, the process noise w_k and measurement noise v_k in (1).

A key point of this approach is the availability of a finite buffer at the estimator,[1] whose length is denoted by the integer N. Therefore, the iteration starts from

$$\widehat{x}^k_{k-N|k-N-1} = \widehat{x}^{k-1}_{k-N|k-N-1}, P^k_{k-N|k-N-1} = P^{k-1}_{k-N|k-N-1}$$

if $t > N$ or from

$$\widehat{x}^t_{0|-1} = \widehat{x}_0, \quad P^t_{0|-1} = P_0$$

otherwise.

The buffer is used to store all the relevant estimator quantities, i.e., the state prediction $\widehat{x}^k_{t|t-1}$ and estimate $\widehat{x}^k_{t|t}$, the covariance matrices $P^k_{t|t-1}$ and $P^t_{t|t}$ (of the state prediction and the estimation errors respectively), the plant inputs u_t, the received plant outputs y_t, and the arrival sequence δ^k_t.

An important assumption is that N corresponds also to the maximum allowable delay: any measurement older than N estimator sampling periods (i.e., older than $k - N + 1$, since k denotes the current estimator sampling instant) is neglected by the estimator, that is, it is treated as a lost packet. Note that, at each time instant, the estimator requires to be iterated only from the oldest measurement received at that instant: from such starting point, the iteration involves a prediction (open-loop) step if a measurement is not available in the buffer (i.e., $\delta^k_t = 0$), or an estimate (closed-loop) step otherwise (i.e. $\delta^k_t = 1$). The obtained optimal estimator resembles a modified Kalman filter endowed with a buffer.

The estimated state is used to implement a state feedback controller of the type

$$u_k = K\hat{x}_{k|k-1} \tag{10}$$

whose gain matrix $K \in \mathbb{R}^{1\times 4}$ is computed with the linear quadratic (LQ) optimality principle, namely to minimize the following infinite horizon cost function:

$$J = \lim_{S\to\infty} \mathbb{E}\left[\frac{1}{S}\sum_{k=0}^{S-1} x'_k W x_k \; + u'_k U u_k \,\middle|\, \{u_k\}_{k=0}^{S-1}\right]. \tag{11}$$

The weights U and W (semidefinite and definite matrices, respectively) are selected according to the Bryson's rule, namely inversely proportional to the maximum allowed deviations of the weighted quantities from their steady-state values. An additional integral action is included in the control law to achieve robust perfect tracking of a step position reference.

B. Communication Application

The communication between controller and plant exploits the Wi-Fi interfaces of off-the-shelf low-cost embedded systems, equipped with a Linux-based operating system. Given the proposed architecture, we hence leverage on the common Transmission Control Protocol (TCP)/Internet Protocol (IP) stack of the Linux kernel. However, the utmost importance of reducing any source of latency for critical and time-sensitive data delivery fosters a careful protocol selection and tuning. Thus, as a first design choice, the communications involved in the considered industrial control system exploit the User Datagram Protocol (UDP), following a common practice of real-time industrial protocols [41].

[1]Indeed, the solution is optimal only if the buffer has an infinite length. However, for implementation purposes, a finite buffer length version is typically preferred, leading however to a suboptimal solution (see [36]).

Algorithm 1: Access Policy to Buffer of Incoming UDP Packets.

Input: buffer of incoming packets
Output: buffer of last N received packets

```
iPkt ← 0
expectedPktSize ← size of expected packet
loop
  extract packet (receivedPkt) from input buffer
  if receivedPktSize = expectedPktSize then
    outBuffer[iPkt] ← receivedPkt
    iPkt ← iPkt + 1
    if iPkt ≥ N then
      iPkt ← 0
    end if
  else
    exit loop
  end if
end loop
return outBuffer
```

The communication application is implemented in a time-efficient manner, by exploiting non-blocking accesses to the buffer of incoming data, and by extracting only the most recently received packets, as detailed in Algorithm 1. Indeed, N is the estimator buffer size as described in Section IV-A; the receiver routine reads the datagrams starting from the first received (according to the UDP standard FIFO policy) and holds them in an output array of N elements. Nonblocking readings are repeated until the incoming buffer is empty. If the number of incoming packets is greater than N, the ith packet (with $i > N$) replaces the $(i - N)$th packet in the output array and so on, until no more packets are available at the incoming buffer. A check on the incoming packet size ensures that the packet is not incomplete or the UDP buffer is not empty, otherwise the reading loop is aborted. Also, Algorithm 1 does not guarantee that the N packets in the output buffer are ordered from last to first, but it guarantees that the last N packets are not discarded.

Some meaningful Wi-Fi features and parameters can be managed to adequately support the cross-layer architecture. To this regard, some special custom blocks have been developed to implement the following functionalities.

1) Set the Wi-Fi communication standard (IEEE 802.11 g/n) and the related bitrate.
2) Set the number of MAC transmission retries.
3) Retrieve the signal power of the last received packet.
4) Retrieve the noise level of the Wi-Fi channel.
5) Enable/disable the IEEE 802.11 power-save mode.

Since all the embedded systems selected for the tests run the GNU/Linux operating system, these functionalities are implemented by resorting to the *nl80211* library that provides a powerful interface to properly manage the IEEE 802.11 protocol

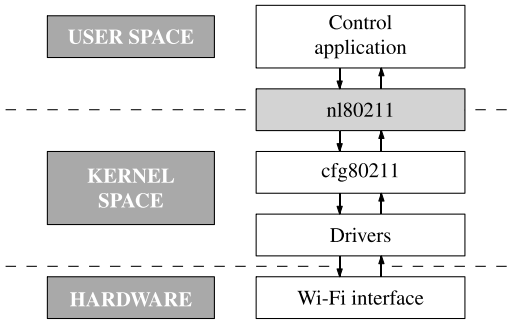


Fig. 2. IEEE 802.11 management from user-space: implementation details.

features from the user space, as shown by the block diagram reported in Fig. 2. Moreover, to further deepen the control over the transmission policies, a set of customized blocks to send and receive UDP packets have been developed, leveraging on standard POSIX socket libraries.

V. Implementation

This section provides a description of the different building blocks that realize the proposed architecture. The cross-layer architecture described in the previous section is implemented and enforced exploiting a rapid control prototyping (RCP) methodology. The experimental assessment is, instead, carried out leveraging on a hardware-in-the-loop scheme.

A. Rapid Control Prototyping

The control system application has been developed as a Simulink block-diagram model, which is then automatically converted into a C-language code that can be compiled and run in real-time on the embedded system. Code generation is automatically managed either by the *Simulink Coder* for general purpose conversions, or by the *Embedded Coder* for optimized conversions targeted to embedded systems. Ad hoc Simulink blocks with custom functionalities can be easily implemented by providing the instructions, in form of *Target Language Compiler* (TLC) files, for the code generation process. This feature is particularly useful for the development of specific blocks that allow to interface the Simulink model with the underlying embedded system hardware.

The code generated for real-time execution on a target running a multitasking operating system has the structure reported in Algorithm 2. A function (*ModelExecutionTask* at line 1) is generated to repeatedly run the model code inside an iterative loop. Each iteration is synchronized with a semaphore (*clockSem*), and starts only when the semaphore is set to true (line 6). The semaphore is periodically set to true ("semaphore give" operation) by the interrupt service routine of a real-time clock enabled at the beginning of the application (line 16). Once the iteration starts, the semaphore value is set to false ("semaphore take" operation), and a single step of the model is evaluated (lines 7–10). An *overrun* condition, namely a situation when the execution of a single iteration exceeds the specified sampling time (equal to the real-time clock period), can be detected by verifying whether the semaphore is already set when the iteration loop is restarted (line 3).

Algorithm 2: Auto-Generated Real-Time Single-Tasking Application (Using Operating System Multitasking Primitives).

```
1:  function ModelExecutionTask
2:    loop
3:      if clockSem = true then
4:        Overrun condition: perform suitable actions.
5:      end if
6:      wait on clockSem
7:      evaluate model step
8:        – evaluate model blocks outputs
9:        – update model blocks states
10:     end evaluate
11:   end loop
12: end Function
13: procedure Main
14:   Initialization
15:   start ModelExecutionTask
16:   start clock that periodically sets
      clockSem ← true
17: end procedure
```

The advantages offered by the RCP methodology in the design and development of a generic control systems are significant. The intrinsic modularity of Simulink models allows to rapidly set up prototype applications that can be promptly tested in laboratory. In addition, the capability to implement custom blocks boosts the possibilities of this technique, allowing to create specific toolboxes focused on the purposes of the research activity. Finally, the portability of the developed software and the acquired know-how over different platforms and projects are greatly enhanced, since blocks of code can be reused just like traditional Simulink blocks.

B. Hardware-in-the-Loop Testbed

A hardware-in-the-loop experiment has been conceived and implemented in order to verify the performance of the control at high sampling rates over Wi-Fi. The test setup resembles exactly the configuration depicted in Fig. 1(b), with the only difference that the dynamics of the real plant, i.e., the balancing robot, is numerically simulated in real-time on a personal computer. The test setup is depicted in Fig. 3.

The control loop contains the real physical link between controller and plant, thus allowing to carefully analyze the effects of communication delays and packet losses on the performance of the control system. In order to obtain realistic and meaningful results from the experimental tests, particular attention has been devoted to derive an accurate model of the balancing robot, that accounts for both the nonlinear dynamics of the multibody mechanism (obtained with a Lagrangian approach), and all the relevant nonidealities of the sensors and actuators (such as actuator saturations, encoders quantizations, sensors noises, and

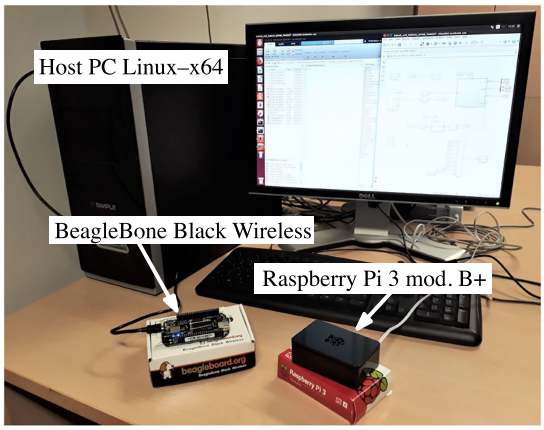


Fig. 3. Test setup.

finite bandwidths). Some slightly simplifying assumptions have been introduced in this model, to ensure the feasibility of the simulation at high sampling rate. In particular, without loss of generality, the robot has been described with a simplified planar model [42].

C. Test Schemes

The tests have been performed by exploiting an IEEE 802.11n network, on Channel 6 (2437 MHz) with 20 MHz of bandwidth. Tests were conducted in an university laboratory crowded with several Wi-Fi networks, some of them actually operating on the same channel (Channel 6) selected for the experiments. Background noise has been measured with a GNU/Linux PC with the same hardware configuration of that used in the tests (referred to as PC #1 in the following), providing a measure of -95 dBm.

The experimental campaign involved different computing devices, as listed below. Specifically, we used two embedded systems and a classic desktop PC, as a reference to compare the performance of the embedded systems.

1) Raspberry Pi 3 mod. B—Broadcom BCM2837 Wi-Fi module;
2) BeagleBone Black Wireless—Texas Instruments WiLink 1835 module;
3) GNU/Linux (PC #1)—Intel Core i5-6400, 2.70GHz, 16GB RAM, Ubuntu 17.04, TP-Link PCI Wi-Fi interface based on the Qualcomm Atheros AR9227 chip.

Multiple tests, as listed below, have been carried out in order to determine the performance of the involved devices, as well as to assess the feasibility of a realistic control-over-Wi-Fi experiment.

Test #1: Measurement of the execution times of the user-space Wi-Fi management routines (based on the *nl80211* library).

This test takes into account the *nl80211*-based functionalities from 1) to 4) as listed in Section V-A. All routines are invoked with a period of 1 ms (i.e., sampling rate of 1000 Hz). The execution times are recorded at each sample time. The test, which is carried out on each machine, lasts for 10 s, thus collecting a total of 10 000 samples each.

Test #2: Assessment of the communication link.

This test is carried out to evaluate the ability to handle the access policy described by Algorithm 1, that is implemented by the controller. As shown in Fig. 4(a), the tested machine (A) sends a sequence of numbered UDP packets to a second machine (B) through the Wi-Fi network (Fig. 4(a)). Machine B is a personal computer set up as a communication relay through the Netcat utility [43] that simply sends the data back on the same network. Machine A receives the frame exploiting the routine described in Algorithm 1. The number of MAC transmission retries was set to one, in order to minimize time delays due to multiple transmissions, whereas the bitrate is not forced to a specific value in these experiments, since each system exploits its own default rate adaptation algorithm. Transmitted packets consist of a 8-byte payload and a 4-byte header that holds a progressive packet identifier. The tests last for 20 s and are carried out at different packet delivery rates (0.1–1 kHz) for each tested device. The Wi-Fi antennas are roughly 1 m apart from each other.

In order to correctly understand the outcomes of Test #2, the communication performance and link quality are assessed by means of both the packet error rate (PER) and the communication latency d. Based on the implementation of the receiver in Fig. 4(a), a lost packet is defined as a packet that during communication is either not received, or discarded by the receiver, since the number of new incoming packets at a certain sampling instant exceeds the buffer length N. Thus, the PER at the kth packet is computed as the moving average of the number of lost packets over the previous M transmissions. The chosen value of M is equal to the estimator buffer length N. Considering the definition of δ_k given with the function in (2) to count packet arrivals, then PER_k at the kth packet is computed as follows:

$$\mathrm{PER}_k = 1 - \sum_{i=k-M+1}^{k} \frac{\delta_i}{M}. \tag{12}$$

The communication latency is computed for each transmitted packet. Each packet contains an identifier, equal to the iteration step number k at which the packet was created and sent. The same packet is received back (by the same device) at the jth iteration step, with $j > k$. The delay of the kth packet, received at the jth time instant, is hence computed as $d_k = j - k$. An infinite delay is assigned to lost packets, so that their actual delay is ignored when computing cumulative delay statistics. Evaluating delays in this way ensures an independence on the exact value of the sampling time used within an experiment, and allows a direct comparison among relative latency values.

Test #3: Control-over-Wi-Fi simulation experiment

This test relies on the architecture shown in Fig. 4(b). The controller is the Raspberry Pi (this board has been chosen since it performs better than the BeagleBone, as will be shown in the next section). The plant is represented by the Linux PC #1 which, according to the classic hardware-in-the-loop policy, executes the simulation of the plant dynamics. The controller periodically executes the control algorithm and sends reference signals to the plant from which it receives state measurements. The initial conditions for angles are $\vartheta_0 = 5^\circ$ and $\gamma_0 = 0^\circ$, whereas initial

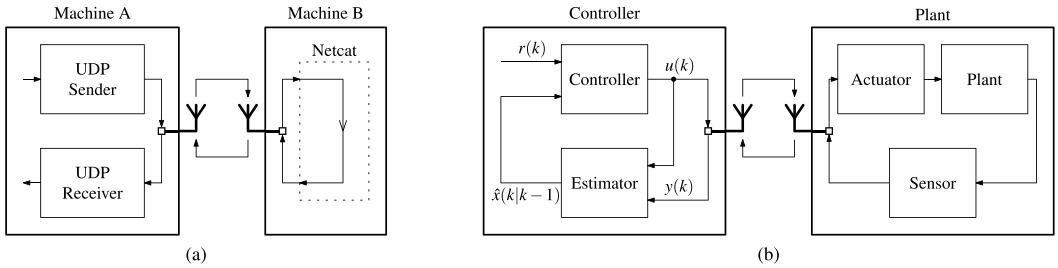


Fig. 4. Experimental tests. (a) Wi-Fi performance experiment (Test #2). (b) Control-over-Wi-Fi experiment (Test #3).

TABLE I
TEST #1: EXECUTION TIMES OF WI-FI MANAGEMENT ROUTINES (AVERAGE OVER 10K SAMPLES)

	Raspberry		**BeagleBone**		**Linux-x64**	
	mean [µs]	std [µs]	mean [µs]	std [µs]	mean [µs]	std [µs]
Set bitrate	-	-	221	121	49	6
Set # of retries	56	31	216	129	50	6
Get RX power	2506	393	661	181	90	7
Get noise	-	-	-	-	99	6

TABLE II
TEST #2: PACKET ERROR RATE (PER) VS. SAMPLING FREQUENCY (AVERAGE OVER 100 RUNS)

	Raspberry		**BeagleBone**		**Linux-x64**	
frequency [Hz]	PER [%]	std [%]	PER [%]	std [%]	PER [%]	std [%]
100	5.8	3.7	14.8	1.8	2.5	1.4
200	4.5	2.5	23.6	2.0	3.0	1.7
250	4.1	1.9	26.2	1.6	3.4	1.5
400	4.9	2.3	28.9	3.0	5.3	1.8
500	7.8	2.7	28.7	5.7	6.2	2.6
1000	8.3	4.8	22.5	7.6	9.0	2.6

velocities are $\dot{\vartheta}_0 = \dot{\gamma}_0 = 0$ deg/s. The reference signal that the plant is required to track is zero for ϑ (i.e., hold to the vertical position) and a step change of 0.1 m for γ, occurring at $t = 10$ s (thus allowing the system to recover from the initial transient due to nonzero initial conditions). Different values of state estimator buffer length, N, have been tested, showing that the optimal performance can be achieved with a value between 4 and 20. Consequently, in the experiments, the buffer length of the state estimator has been set as $N = 20$. The simulation time is 32 s.

VI. RESULTS AND DISCUSSION

A. Test #1: Execution Times

Table I presents the execution times of the routines implementing the functionalities listed in Section V-A from 1) to 4). Within the table, missing data refers to functionalities that are either disabled or not available on the corresponding device.

Unsurprisingly, PC #1 proved the best performance, with execution times under 100μs for all considered functionalities. On the Raspberry, only two functionalities were accessible. In particular, sensing the received power takes more than 2.5 ms, on average, whereas setting the number of transmission retries is far less time-consuming. Finally, the BeagleBone system showed the worst performance, taking hundreds of microseconds for all enabled functionalities.

B. Test #2: Communication Performance

The plots in Fig. 5 report the cumulative distribution function relevant to the latency with which packets sent by Machine A are received back, in a typical polling application. In particular, the left column shows the latency in terms of absolute time (ms), whereas, on the right, the same plots are shown with latency values scaled to the sample time. The curves are averages over 100 runs (20 s of execution for each run), and each of them is relevant to a different value of sampling frequency in the range 100–1000 Hz. The PER values, averaged over 100 test runs, and their standard deviations are listed in Table II.

As in Test #1, PC #1 shows the best performance, with PERs always below 10% and low latency. However, the performance figures of the Raspberry board are also good, with PERs comparable with those of PC #1.

Conversely, the BeagleBone board shows the worst performance, with a behavior strongly dependent on the sampling frequency. This is likely due to the limited processing capability of such board, which does not cope adequately with the increase of the computational burden caused by the much more frequent execution of both Algorithm 1 and the communication task. Also, data dispersion becomes greater at the higher sampling frequencies, reflecting on high uncertainty, particularly for the curve at 1000 Hz (as can be seen in Table II, the packet loss has the highest standard deviation at this frequency). As a consequence, the behavior of the packet delay, in Fig. 5(b), results highly dispersed, leading the curve at 1000 Hz to cross curves relevant to other frequencies.

C. Test #3: Control-over-Wi-Fi

The results of Test #1 and #2 allow to identify the Raspberry Pi 3 mod. B as the better embedded system for running the control routine with the requirements imposed by the control-over-Wi-Fi approach proposed in this article. Thus, this section presents the results of Test #3 with such an embedded system as the control unit. All the results refer to a single simulation

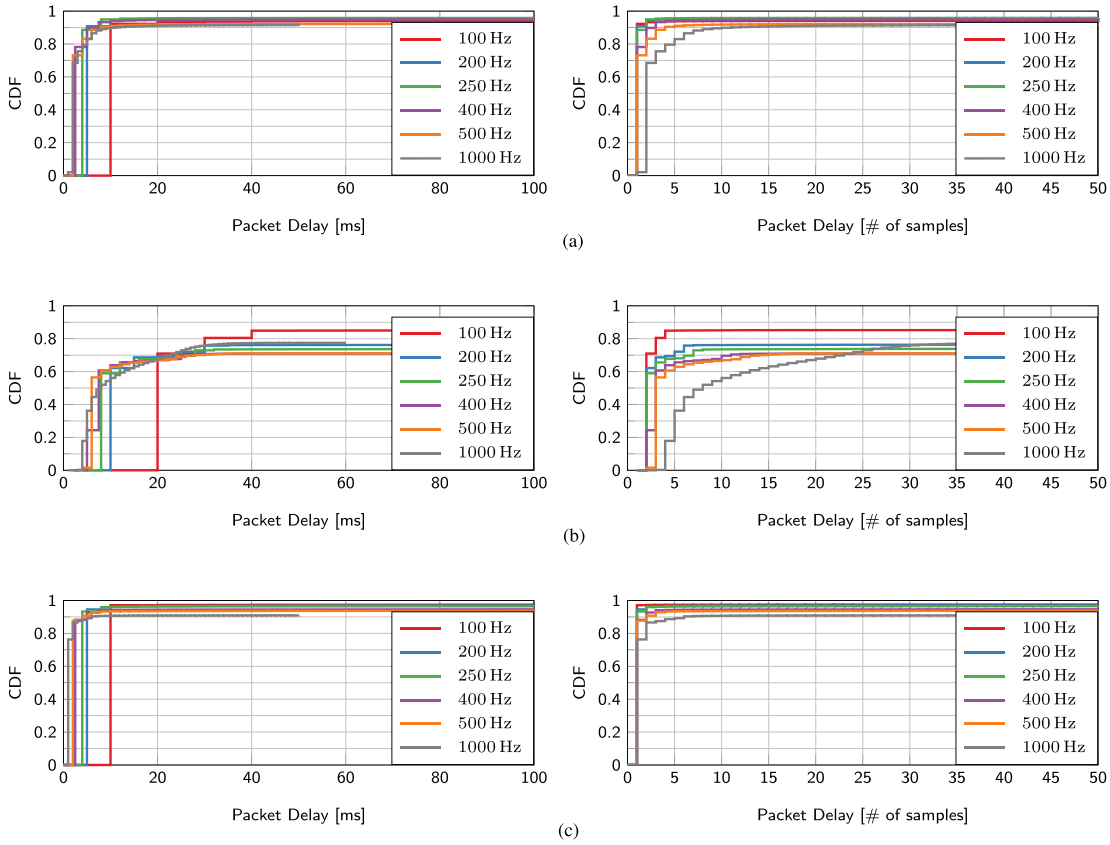


Fig. 5. Wi-Fi performance tests (average over 100 runs); packet delay expressed in terms of absolute time (left) and number of sample times (right). (a) Raspberry Pi, (b) BeagleBone Black, and (c) Linux-x64.

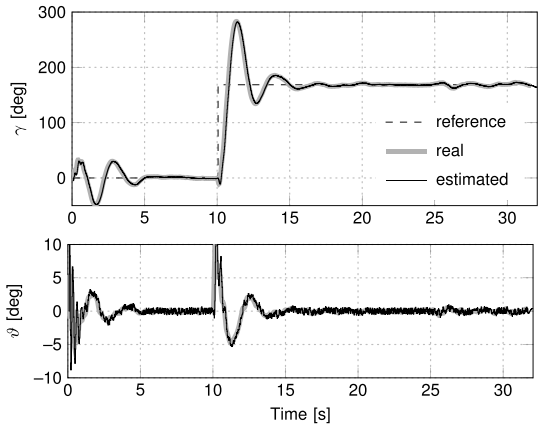


Fig. 6. Assessment of the control-over-Wi-Fi approach: step response.

TABLE III
TEST #3: STEP RESPONSE PARAMETERS (AVERAGE OVER 10 RUNS)

	Wi-Fi		Ethernet	
	mean	std	mean	std
M_p [%]	84.18	12.10	72.88	8.41
t_r [s]	0.69	0.04	0.71	0.001
t_s [s]	9.29	5.91	4.17	0.47

TABLE IV
TEST #3: EXECUTION TIMES (AVERAGE OVER 10 RUNS)

	mean [µs]	std [µs]	max [µs]	min [µs]
RPi (control routine)	274	117	1006	102
PC #1 (plant simulation)	109	28	301	9

run, which was selected as representative of the typical system behavior.

The test is performed by evaluating the response of the control system to a step variation of the robot position reference. The sampling rate was set to 1000 Hz. A typical outcome is shown in Fig. 6, where both the wheel and tilt angles are included. Table III reports the statistics on the main step response parameters (overshoot, M_p, rise time, t_r, and settling time, t_s) obtained over 10 runs of the same experiment. The same table provides a comparison with the results obtained running the same test over an Ethernet connection.

The execution times of both the control and plant simulation routines are reported in Table IV (statistics over 10 runs). It can be noticed that the execution time of the control routine on the selected embedded system is, on average, well below the 1 ms sampling time threshold.

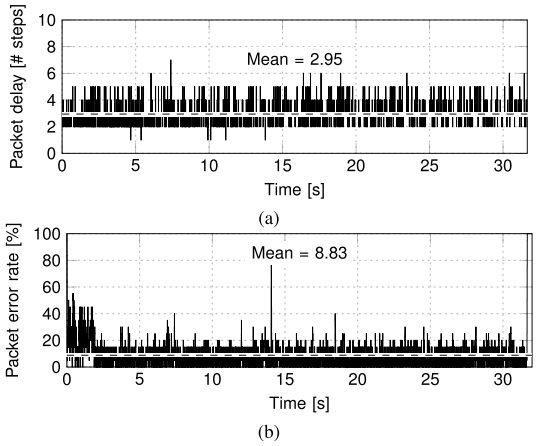


Fig. 7. Control over Wi-Fi test: communication performance.

TABLE V
TEST #3: COMMUNICATION PERFORMANCE (AVERAGE OVER 10 RUNS)

	Wi-Fi		Ethernet	
	mean	std	mean	std
packet delay [# steps]	2.68	0.69	1.37	0.36
PER [%]	10.88	12.94	1.90	9.84

The communication performance for the step response test of Fig. 6 is presented in Fig. 7(a) and (b). For the single-run test of Fig. 6, the average packet delay, scaled to the sample time, is equal to 2.95, whereas the packet loss is equal to 8.83%. The communication statistics, as computed over 10 runs, are summarized in Table V. The Table also reports results obtained using a legacy Ethernet connection. They have been derived from an additional experiment carried out to provide a mean to compare the performance of the wireless connection with that of a wired one, which can be considered as a "reference communication system." Actually some packets are lost even when Ethernet is used, probably as an effect of the computational burden. Finally, it is worth observing that the statistics of the Raspberry Pi show a certain difference between Test #2 and Test #3. Again, we believe this is due to the greater computation effort required during Test #3 compared to Test #2, which is much simpler and does not imply the execution of the control routine.

D. Discussion and Future Perspectives

The results of Test #3 are definitely encouraging, since they show the ability of the system to ensure an adequate level of performance, as well as to cope with the uncertainty of the communication medium. However, since the Raspberry Pi board did not allow to dynamically modify most of its communication parameters, the potentiality of the communication and control co-design could not be adequately exploited. Thus, it may be stated that the obtained results allowed to assess both the feasibility and the effectiveness of the proposed control scheme, paving the way to a complete implementation of the co-design approach.

Hence, considerable improvements are legitimately expected concerning the co-design technique. Indeed, the availability of both the "Set bitrate" and "Get noise" features leads to a twofold positive impact. On the one hand, it is possible to implement effective adaptive rate control strategies, which ensure the deadlines associated with packet transmission are met with a given probability [30], [44]. This makes the underlying communication system able to guarantee a predictable level of performance to the control application, which can thus be designed in a more straightforward and systematic way. On the other hand, and more importantly from the co-design perspective, the control law can be dynamically adapted as a function of the communication channel status [45], [46]. For example, a more accurate knowledge of the probability distribution of the transmission delays can be effectively exploited to design the estimator buffer size (N), in such a way to not exceed a tolerated level of packet dropouts. In the same direction, in [32], it is shown how to continuously adjust the LQ controller gains and transmission rate, based on of the monitored SNR, to minimize the overall LQ cost function. Although results are presented for a numerically simulated scenario involving a single wireless communication link in the control loop, the proposed strategy is intriguing and highlights a research field that, to the best of the authors' knowledge, is still largely unexplored.

Other interesting developments are expected from the continuous evolution of industrial communication systems that will provide further functionalities and better performance to the control applications. Indeed, the Time Sensitive Networking (TSN) family of standards [47], which natively enables real-time and determinism properties on Ethernet networks, will be likely extended to wireless networks, such as WiFi and 5G [48].

Particularly, the new features introduced by IEEE 802.11ax (the most recent amendment to the IEEE 802.11 wireless LAN), as well as the forthcoming IEEE 802.11be amendment, envisage a profitable coupling with TSN, which will lead to definitely better WiFi performance in industrial scenarios.

TSN is an appealing opportunity for the arising 5G wireless technologies as well. These are defined by a complex set of specifications that aims to provide, besides the general-purpose enhanced mobile broadband (eMBB) service, the support for other two service classes, namely massive machine-type communications (mMTCs) and ultrareliable low-latency communications (URLLCs), characterized by diverse features [49]. Specifically, mMTCs have been designed to support a massive number of (possibly industrial) IoT devices sporadically sending small data payloads. Conversely, URLLCs (that still adopts small payload packets) is meant for transmissions with very low latency and high reliability, in agreement with transmission patterns defined by the application, thus clearly targeting industrial usage. In the described scenario, some significant research works are dealing with the topic of co-design of real-time wireless control over the URLLCs provided by 5G [50]–[52]. However, the performance figures of 5G technologies, to the best of the authors' knowledge, have not been yet adequately assessed in industrial environments, especially in terms of both reliability and latency, at least from an experimental point of view. Only few simulation studies are available. As highlighted in [53], the

achievable performance in terms of round-trip time and service time, that are typical of industrial scenarios, are not (yet) able to cope with the requirements of applications like those presented in this work.

VII. Conclusion

This article addressed the control of time-critical industrial systems at high sampling frequencies up to 1000 Hz over Wi-Fi, in which both the increasing computing capabilities of low-cost networked embedded systems and a suitable control strategy were exploited to cope with unbounded latency and nondeterminism of Wi-Fi networks.

The feasibility of the proposed co-design approach was demonstrated and an extensive performance assessment was carried out. The outcomes of the experimental campaign, although preliminary, highlighted the effectiveness of the approach. As an immediate prosecution of the activities described in this article, we will focus on the implementation of advanced hardware-in-the-loop experiments, like those involving the real balancing robot. Furthermore, the opportunity discussed in the Sec. VI to improve the potentiality of the proposed cross-layer approach will allow to address more complex applications in real industrial environments.

References

[1] R. Poovendran *et al.*, "Special issue on cyber-physical systems [scanning the issue]," *Proc. IEEE*, vol. 100, no. 1, pp. 6–12, Jan. 2012.

[2] S. Vitturi, C. Zunino, and T. Sauter, "Industrial communication systems and their future challenges: Next-generation ethernet, IIoT, and 5G," *Proc. IEEE*, vol. 107, no. 6, pp. 944–961, Jun. 2019.

[3] H. Xu, W. Yu, D. Griffith, and N. Golmie, "A survey on industrial internet of things: A cyber-physical systems perspective," *IEEE Access*, vol. 6, pp. 78238–78259, 2018.

[4] L. Chen, P. Zhou, L. Gao, and J. Xu, "Adaptive fog configuration for the industrial internet of things," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4656–4664, Oct. 2018.

[5] J. Song *et al.*, "WirelessHART: Applying wireless technology in real-time industrial process control," *Proc. IEEE Real-Time Embedded Technol. Appl. Symp.*, 2008, pp. 377–386.

[6] "ISA100," [Online]. Available: http://www.isa.org/isa100

[7] D. Dujovne, T. Watteyne, X. Vilajosana, and P. Thubert, "6TiSCH: Deterministic IP-enabled industrial internet (of things)," *IEEE Commun. Mag.*, vol. 52, no. 12, pp. 36–41, Dec. 2014.

[8] R. Steigmann and J. Endresen, "Introduction to WISA: WISA – Wireless interface for sensors and actuators," White paper, ABB, 2006.

[9] X. Zhu *et al.*, "Measuring wirelesshart against wired fieldbus for control," in *Proc.IEEE 10th Int. Conf. Ind. Inform.*, Jul. 2012, pp. 270–275.

[10] S. Petersen and S. Carlsen, "Wirelesshart versus ISA100.11a: The format war hits the factory floor," *IEEE Ind. Electron. Mag.*, vol. 5, no. 4, pp. 23–34, Dec. 2011.

[11] G. Patti, G. Alderisi, and L. L. Bello, "Introducing multi-level communication in the IEEE 802.15.4e protocol: The multichannel-LLDN," in *Proc. IEEE Emerg. Technol. Factory Automat.*, Sep. 2014, pp. 1–8.

[12] G. Alderisi, G. Patti, O. Mirabella, and L. L. Bello, "Simulative assessments of the IEEE 802.15.4e DSME and TSCH in realistic process automation scenarios," in *Proc. IEEE 13th Int. Conf. Ind. Inform.*, Jul. 2015, pp. 948–955.

[13] *IEEE standard for information technology–Telecommunications and information exchange between systems local and metropolitan area networks-specific requirements – Part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications*, IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012), Dec. 2016, pp. 1–3534.

[14] Q. Xu, "Design and smooth position/force switching control of a miniature gripper for automated microhandling," *IEEE Trans. Ind. Informat.*, vol. 10, no. 2, pp. 1023–1032, May 2014.

[15] S. Sakaino, T. Sato, and K. Ohnishi, "Precise position/force hybrid control with modal mass decoupling and bilateral communication between different structures," *IEEE Trans. Ind. Informat.*, vol. 7, no. 2, pp. 266–276, May 2011.

[16] S. Vitturi, F. Tramarin, and L. Seno, "Industrial wireless networks: The significance of timeliness in communication systems," *IEEE Ind. Electron. Mag.*, vol. 7, no. 2, pp. 40–51, Jun. 2013.

[17] D. M. Anand, J. R. Moyne, and D. M. Tilbury, "Performance evaluation of wireless networks for factory automation applications," in *Proc. IEEE Int. Conf. Automat. Sci. Eng. Conf.*, 2009, pp. 340–346.

[18] F. Tramarin, S. Vitturi, M. Luvisotto, and A. Zanella, "On the use of IEEE 802.11n for industrial communications," *IEEE Trans. Ind. Informat.*, vol. 12, no. 5, pp. 1877–1886, Oct. 2016.

[19] F. Tramarin, A. K. Mok, and S. Han, "Real-time and reliable industrial control over wireless LANs: Algorithms, protocols, and future directions," in *Proc. IEEE*, vol. 107, no. 6, pp. 1027–1052, Jun. 2019.

[20] S. Tozlu, M. Senel, W. Mao, and A. Keshavarzian, "Wi-Fi enabled sensors for internet of things: A practical approach," *IEEE Commun. Mag.*, vol. 50, no. 6, pp. 134–143, Jun. 2012.

[21] I. Dominguez-Jaimes, L. Wisniewski, H. Trsek, and J. Jasperneite, "Link-layer retransmissions in IEEE 802.11g based industrial networks," in *Proc. IEEE Int. Workshop Factory Commun. Syst.*, May 2010, pp. 83–86.

[22] G. Patti, G. Alderisi, and L. Lo Bello, "SchedWiFi: An innovative approach to support scheduled traffic in ad-hoc industrial IEEE 802.11 networks," in *Proc. 20th IEEE Conf. Emerg. Technol. Factory Automat.*, Sep. 2015, pp. 1–9.

[23] R. Costa, P. Portugal, F. Vasques, C. Montez, and R. Moraes, "Limitations of the IEEE 802.11 DCF, PCF, EDCA and HCCA to handle real-time traffic," in *Proc. IEEE 13th Int. Conf. Ind. Inform.*, Jul. 2015, pp. 931–936.

[24] EU Funded FP7 Research Program. flexWARE. [Online]. Available: http://www.flexware.at

[25] G. Gaderer, P. Loschmidt, and A. Mahmood, "A novel approach for flexible wireless automation in real-time environments," in *Proc. IEEE Int. Workshop Factory Commun. Syst.*, May 2008, pp. 81–84.

[26] T. Sauter, J. Jasperneite, and L. L. Bello, "Towards new hybrid networks for industrial automation," in *Proc. IEEE Int. Conf. Emerg. Technol. Factory Automat.*, 2009, pp. 1141–1148.

[27] W. Liang *et al.*, "WIA-FA and its applications to digital factory: A wireless network solution for factory automation," *Proc. IEEE*, vol. 107, no. 6, pp. 1053–1073, Jun. 2019.

[28] G. Santandrea. "A PROFINET IO application implemented on wireless LAN, WFCS 2006 Industry day," 2006. [Online]. Available: http://wfcs2006.ieiit.cnr.it/indday/siemens.pdf

[29] N. Ormond, "Welcome to the wireless world," *Comput. Control Eng. J.*, vol. 17, no. 1, pp. 28–31, Feb. 2006.

[30] M. Luvisotto, F. Tramarin, and S. Vitturi, "A learning algorithm for rate selection in real-time wireless LANs," *Comput. Netw.*, vol. 126, pp. 114–124, 2017.

[31] Y.-H. Wei, Q. Leng, S. Han, A. K. Mok, W. Zhang, and M. Tomizuka, "RT-WiFi: Real-time high-speed communication protocol for wireless cyber-physical control applications," in *Proc. IEEE 34th Real-Time Syst. Symp.*, 2013, pp. 140–149.

[32] M. Pezzutto, F. Tramarin, L. Schenato, and S. Dey, "SNR-triggered communication rate for LQG control over Wi-Fi," in *Proc. IEEE Conf. Decis. Control*, 2018, pp. 1725–1730.

[33] Q. Qi and F. Tao, "Digital twin and big data towards smart manufacturing and industry 4.0: 360 degree comparison," *IEEE Access*, vol. 6, pp. 3585–3593, 2018.

[34] G. S. Martínez, S. Sierla, T. Karhela, and V. Vyatkin, "Automatic generation of a simulation-based digital twin of an industrial process plant," in *Proc. 44th Annu. Conf. IEEE Ind. Electron. Soc.*, Oct 2018, pp. 3084–3089.

[35] F. Branz, R. Antonello, F. Tramarin, T. Fedullo, S. Vitturi, and L. Schenato, "Embedded systems for time–critical applications over Wi-Fi: Design and experimental assessment," in *Proc. IEEE Ind. Inform.*, 2019, pp. 1756–1759.

[36] L. Schenato, "Optimal estimation in networked control systems subject to random delay and packet drop," *IEEE Trans. Autom. Control*, vol. 53, no. 5, pp. 1311–1317, Jun. 2008.

[37] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, M. I. Jordan, and S. S. Sastry, "Kalman filtering with intermittent observations," *IEEE Trans. Autom. Control*, vol. 49, no. 9, pp. 1453–1464, Sep. 2004.

[38] M. Huang and S. Dey, "Stability of Kalman filtering with Markovian packet losses," *Automatica*, vol. 43, no. 4, pp. 598–607, 2007.

[39] E. R. Rohr, D. Marelli, and M. Fu, "Kalman filtering with intermittent observations: On the boundedness of the expected error covariance," *IEEE Trans. Autom. Control*, vol. 59, no. 10, pp. 2724–2738, Oct. 2014.

[40] H. Gao, X. Meng, and T. Chen, "Stabilization of networked control systems with a new delay characterization," *IEEE Trans. Autom. Control*, vol. 53, no. 9, pp. 2142–2148, Oct. 2008.
[41] S. Vitturi, "On the use of Ethernet at low level of factory communication systems," *Comput. Standards Interfaces*, vol. 23, no. 4, pp. 267–277, Sep. 2001.
[42] R. Antonello and L. Schenato, "Longitudinal state-space control of the balancing robot," University of Padova, Dept. of Information Engineering, Tech. Rep., 2017. [Online]. Available: http://automatica.dei.unipd.it/tl_files/utenti/lucaschenato/SEGWAY_GUID E.pdf
[43] *Hobbit*. (2007) Netcat 1.10. [Online]. Available: http://nc110.sourceforge.net/
[44] F. Tramarin, S. Vitturi, and M. Luvisotto, "A dynamic rate selection algorithm for IEEE 802.11 industrial wireless LAN," *IEEE Trans. Ind. Informat.*, vol. 13, no. 2, pp. 846–855, Apr. 2017.
[45] A. Saifullah *et al.*, "Near optimal rate selection for wireless control systems," *ACM Trans. Embed. Comput. Syst.*, vol. 13, no. 4, pp. 231–240, Apr. 2014. [Online]. doi: 10.1145/2584652.
[46] P. Park, J. Arajo, and K. H. Johansson, "Wireless networked control system co-design," in *Int. Conf. Netw. Sens. Control*, Apr. 2011, pp. 486–491.
[47] L. Lo Bello and W. Steiner, "A perspective on IEEE time-sensitive networking for industrial communication and automation systems," *Proc. IEEE*, vol. 107, no. 6, pp. 1094–1120, Jun. 2019.
[48] D. Cavalcanti, J. Perez-Ramirez, M. M. Rashid, J. Fang, M. Galeev, and K. B. Stanton, "Extending accurate time distribution and timeliness capabilities over the air to enable future wireless industrial automation systems," *Proc. IEEE*, vol. 107, no. 6, pp. 1132–1152, Jun. 2019.
[49] P. Popovski, K. F. Trillingsgaard, O. Simeone, and G. Durisi, "5G wireless network slicing for EMBB, URLLC, and MMTC: A communication-theoretic view," *IEEE Access*, vol. 6, pp. 55765–55779, 2018.
[50] B. Chang, "URLLC design for real-time control in wireless control systems," in *Proc. IEEE 5G World Forum*, Jul. 2018, pp. 437–439.
[51] B. Chang, G. Zhao, Z. Chen, L. Li, and M. A. Imran, "Packet-drop design in URLLC for real-time wireless control systems," *IEEE Access*, vol. 7, pp. 183081–183090, 2019.
[52] B. Chang, L. Zhang, L. Li, G. Zhao, and Z. Chen, "Optimizing resource allocation in URLLC for real-time wireless control systems," *IEEE Trans. Veh. Technol.*, vol. 68, no. 9, pp. 8916–8927, Sep. 2019.
[53] X. Jiang, M. Luvisotto, Z. Pang, and C. Fischione, "Latency performance of 5G new radio for critical industrial control systems," in *Proc. 24th IEEE Int. Conf. Emerg. Technol. Factory Automat.*, Sep. 2019, pp. 1135–1142.

Francesco Branz received the Ph.D. degree from the Center of Studies and Activities for Space "G. Colombo," University of Padova, Padova, Italy, in 2016, with a thesis on modeling and control of dielectric elastomer actuators for space robotics.

He held a Postdoctoral position with the Department of Information Engineering, University of Padova until December 2019, with focus on the implementation of control over Wi-Fi. He is currently a Lecturer in Attitude Control of Satellites and Aircraft Systems with the Department of Industrial Engineering, University of Padova. His current research interests include CubeSat technologies for attitude simulation and control, close proximity navigation, docking mechanisms, and miniature pointing systems.

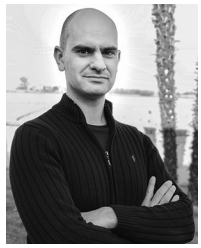
Riccardo Antonello received the Laurea degree in computer engineering in 2002 and the Ph.D. degree in automatic control from the University of Padova, Italy, in 2006.

He has been a Research Associate with the Department of Mechanical and Structural Engineering, University of Trento, Italy, from 2006 to 2010, and then at the Department of Management and Engineering, University of Padova, from 2010 to 2015. Since 2015, he has been with the Department of Information Engineering, University of Padova, as a Laboratory Assistant. His current research interests include control systems, real-time embedded systems, electric drives, and mechatronics.

Luca Schenato (Fellow, IEEE) received the Dr. Eng. degree in electrical engineering from the University of Padova, Padova, Italy, in 1999, and the Ph.D. degree in electrical engineering and computer sciences from the UC Berkeley, in 2003.

He was a Postdoctoral Fellow in 2004 and a Visiting Professor in 2013–2014 with UC Berkeley, Berkeley, CA, USA. Currently, he is currently an Associate Professor with the Information Engineering Department, University of Padova. His current interests include networked control systems, multiagent systems, wireless sensor networks, smart grids, and cooperative robotics.

Dr. Schenato has been awarded the 2004 Researchers Mobility Fellowship by the Italian Ministry of Education, University and Research (MIUR), the 2006 Eli Jury Award in UC Berkeley, and the EUCA European Control Award in 2014. He served as Associate Editor for IEEE TRANSACTIONS ON AUTOMATIC CONTROL from 2010 to 2014 and he is currently a Senior Editor for IEEE TRANSACTIONS ON CONTROL OF NETWORK SYSTEMS and Associate Editor for *Automatica*.



Federico Tramarin (Member, IEEE) received the Dr. Eng. degree in electronic engineering, and the Ph.D. degree in information engineering from the University of Padova, Italy, in 2008 and 2012, respectively.

From 2013 to 2018, he held a Postdoctoral position with the Institute of Electronics and Computer and Telecommunications (IEIIT), National Research Council of Italy (CNR). He was an Assistant Professor with the Department of Management and Engineering, University of Padova, Italy, from 2018 to 2020. Currently, he is an Associate Professor with the "Enzo Ferrari" Engineering Department, University of Modena and Reggio Emilia, Italy. His currentresearch interests include performance analysis and measurements on network systems, with particular focus on real-time industrial systems, wired and wireless communications, and cyber–physical systems.

Dr. Tramarin currently serves in the Editorial Board of several renowned international journals. He is a member of the IEEE Industrial Electronics Society (IES) Technical Committee on Factory Automation, and a TPC Member of several IEEE international conferences. He was also actively involved in the standardization activities of the IEEE 61158 standard for industrial hard real-time communication.

Stefano Vitturi (Senior Member, IEEE) received the Laurea degree in electronics engineering (summa cum laude) from the University of Padova, Padova, Italy, in 1984.

He is currently a Director of Research with the Institute of Electronics and Computer and Telecommunications (IEIIT), National Research Council of Italy (CNR). From 1985 to 1999, he has worked at the Control and Data Acquisition System of RFX, a nuclear fusion experiment lab located in Padova, Italy. From 1999 to 2000, he temporarily moved to the "Galileo Scientific Park," a technological transfer institution located in Padova, where he coordinated research as well as industrial activities. In 2002, he joined the IEIIT, CNR, where he founded the Padova territorial site, located at the Department of Information Engineering, University of Padova. He has authored and coauthored more than 120 scientific publications in referred international journals, proceedings of international conferences, and book chapters. His current research interests include industrial communication systems, real-time communication networks (wired and wireless), and implementation and performance analysis of devices conforming to the most popular industrial communication protocols.

Dr. Vitturi is currently serving as Associate Editor for the IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS.