




Article

New Control Paradigms for Resources Saving: An Approach for Mobile Robots Navigation

Rafael Socas * , Raquel Dormido  and Sebastián Dormido 

Departamento de Informática y Automática, Universidad Nacional de Educación a Distancia, Juan del Rosal 16, Madrid 28040, Spain; raquel@dia.uned.es (R.D.); sdormido@dia.uned.es (S.D.)

* Correspondence: rsocas@telefonica.net; Tel.: +34-629-578-386

Received: 16 December 2017; Accepted: 11 January 2018; Published: 18 January 2018

Abstract: In this work, an event-based control scheme is presented. The proposed system has been developed to solve control problems appearing in the field of Networked Control Systems (NCS). Several models and methodologies have been proposed to measure different resources consumptions. The use of bandwidth, computational load and energy resources have been investigated. This analysis shows how the parameters of the system impacts on the resources efficiency. Moreover, the proposed system has been compared with its equivalent discrete-time solution. In the experiments, an application of NCS for mobile robots navigation has been set up and its resource usage efficiency has been analysed.

Keywords: Event-based control; Networked Control Systems (NCS); resources efficiency; mobile robots; robot navigation

1. Introduction

In recent years, Networked Control Systems (NCS) have been gaining importance in the control community [1]. NCS are distributed architectures composed of controllers, sensors that can obtain information from the environment, actuators for acting on them, and a communication network that connects all the elements to achieve a common goal. Therefore, NCS is a field which includes different disciplines such as control theory, communications, software engineering and computer science. Typical applications where these control systems are being used are: space or terrestrial explorations, factory automation, remote diagnostics and troubleshooting, hazardous environments, experimental facilities, mobile robots, multi-vehicles networks, aircraft, manufacturing plant monitoring, nursing homes or hospitals, tele-robotics, tele-operation, etc.

The elements of an NCS system are called the agents. These elements use a communication network to exchange the information between them. Depending on the application where the NCS is used, this network can be deployed using wireline or wireless technology. These communication networks use digital technology to transmit the information which has constraints in delays and limited bandwidth. The information packaging and the constraints due to the limited resources of the network produce undesirable effects such as packet losses, variable delays and signal quantization issues among others. These effects may disturb the stability and performance of the system [2].

Therefore, reducing the traffic in the network is a critical aspect. If the number of packets is decreased can be guaranteed a predictable bandwidth, and at the same time, the analysis of the delays of the network is simplified [3]. As a conclusion, an important issue in the design of these control systems is to implement protocols for transmitting the sensor signals, the state of the system and the control information in a more effective way.

Some researchers have investigated the timing issues in NCS [4]. In traditional approaches, the controllers are used under the assumption of perfect communication, and then, the Maximum

Allowable Transfer Interval (MATI) between two subsequent message transmissions that ensures closed loop stability under a network protocol is determined. Try Once Discard (TOD) and Round Robin (RR) are protocols implemented based on this philosophy. On the other hand, the MATI protocol is often deployed in a centralized way, therefore, it is not practical for systems of large-scale.

Other proposals have achieved an important reduction of network resources usage without a significant loss of performance. Two approaches have been raised: Model-Based Networked Control Systems (MBNCS) and Event-Based Control (EBC). The basics of the MBNCS have been developed in [5,6], and they have been considered in networks of coupled systems in [7] using periodic communication. Another approach to deal with this problem has been built on an event-based feedback scheme in NCS [8–11]. In event-based control systems, the agent information is broadcast only when some measures of the state error cross a specified level (the event threshold). This control scheme is decentralized in the sense that an agent can broadcast its state using the local information and in an asynchronous way.

In an EBC system, the impact of noise in the sensors increases the number of events and therefore a degradation of the performance. If the disturbance is known or can be modeled, the controller can be properly set to reduce its effects. However, in most applications it is difficult either to estimate the noise level or to have a reliable model. In these cases, it becomes a hard work to tune the controller in a proper way. In [12,13] these problems have been investigated where a new control scheme has been proposed which are dynamically adjusted depending on the conditions of the environment. In both proposals, the algorithms work with an estimation of the noise previously calculated. In [14], the estimation of the noise and the tune of the controller is made in real time.

The EBC strategies have been widely used to control dynamical processes while decreasing considerably the number of packets that the sensors have to send to the controller over the network. In [15] the events based on state errors have been investigated. In [16–18], similar proposals have been analysed to apply in networked interconnected systems. In these works, the use of a zero-order hold (ZOH) in the controller is a common feature. In [19], an event-triggering in networked systems with probabilistic sensor and actuator fault has been investigated to reduce the computation load. An overview on sampled-data-based event-triggered control and filtering for networked systems has been presented in [20]. In this research, a deep investigation of the sampled data-based event-triggered scheme has been made. In general, the event-based control architectures can be a good solution for the systems with limited resources and they could be a more efficient control scheme than the classical ones [21,22].

In this paper, a new event-based control architecture based on a simple event-based control scheme for NCS environments is presented. Making use of the control strategy implemented a full analysis of different resource consumption is carried out. The use of bandwidth, computational load and energy resources are analysed. Several methods and methodologies to measure the efficiency in the resources consumption are also proposed. The main contributions of this work are the models development of resource consumption and their parametrization. Finally, the ideas presented in this work are applied to an NCS mobile robots system to solve the navigation problem.

The paper is organized as follows. Section 2 presents an overview of the event-based control. In Section 3, the principles of sampling criteria are described. In Section 4, the proposed control strategy is presented. Section 5 shows the resource usage in the proposed system. Section 6 presents the experimental results. Finally, the conclusions and future work are discussed in Section 7.

2. Event-Based Control Overview

The event-based control has motivated the interest of the control community in the last few years, multiple control architectures and new applications have been proposed based on these ideas. In [23], an event-driven sampling method called the area-triggered method has been proposed. In this scheme, sensor data are sent only when the integral of the differences between the current sensor value and the last transmitted one is greater than a given threshold. The proposed system reduces the data

transmission rate and also improves the estimation performance in comparison with the conventional time-driven technique. In [24], a greenhouse climate is controlled by an event-based control system. The system is based on a network of wireless sensors to control the low frequency dynamics of the environment. In this case, the control actions are calculated by considering the events that produce the external disturbances. The proposed system increases the actuators life and allows cost savings by minimizing the wear while maintaining a good performance. In [25], an event-based sampling according to a constant energy of sampling error is investigated. The defined criterion is suitable for applications where the energy of the sampling error should be bounded (e.g., in greenhouse climate monitoring and control or in building automation). Finally, in [26], a fault isolation filter to apply on discrete-time networked control systems based on a particular form of the Kalman filter is proposed. The scheme makes an efficient use of the resources with a good estimation of failures and its effect on the performance. The sampled-data-based event-triggered control schemes is another emerging event-based control technique. The reliable control design for networked control system under event-triggered scheme is investigated in [19]. The key idea of this work is that only the newly sampled sensor measurements that violate specified triggering condition will be transmitted to the controller. The main advantage of this approach is that the proposed event-triggered scheme only needs a supervision of the system state in discrete instants and there is no need to retrofit the existing system. Finally, in [20], an overview and a deep investigation on sampled-data-based event-triggered control and filtering for networked systems has been done. Compared with some existing event-triggered and self-triggered schemes, a sampled-data-based event-triggered scheme can ensure a positive minimum inter-event time and make it possible to jointly design suitable feedback controllers and event-triggered threshold parameters.

In event-based control systems, information is exchanged between the elements (controller, sensors and actuators) depending on the state of the system [27]. When the system variables exceed a certain level an event is generated in the system and the control actions are executed. This means that the activity of the controller and the use of resources to communicate the different elements are restricted to the time intervals in which a control action must inevitably be taken to guarantee the system specifications.

In Figure 1, the basic scheme of an event-based control strategy is presented [28,29].

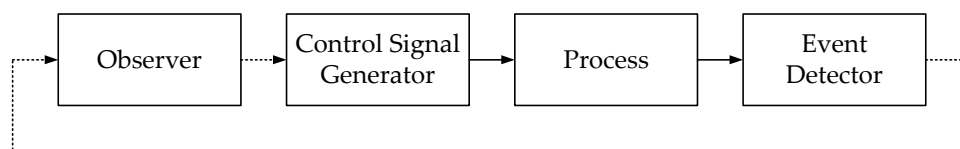


Figure 1. Basic scheme of an event-based control system. Solid arrows represents continuous signal and dashed arrows represents event-based signals.

The control scheme is composed of an event detector, an observer, and a control signal generator. The event detector generates an output signal when an event occurs, it happens when the error signal crosses a threshold. When an events occurs, the observer is updated and it passes the information to the control signal generator. With this information, the control signal generator generates the input signal to control the process. An important aspect of this strategy is that the observer and the control signal generator works in open loop between events.

This control architecture combines feedback and feedforward strategies. When an event is generated there is a feedback action. On the other hand, the feedforward actions happen when the actuators are driven by the control signal generator in open loop between events.

3. Event-Based Sampling Schemes

Different sampling criteria have been proposed in the event-based control schemes [27]. In the event-based sampling methods, the system acts only when the variables of the plant are in

transient state. In the steady state the system does not act and some resources can be saved. On the other hand, in discrete time schemes the sampling is executed periodically (with the period $T = 1/f_s$ where f_s is the sampling frequency) and it does not depend on the state of the system (Figure 2a).

Send-on-delta and integral criterion are the most widely used techniques in event-based control schemes. In the following sections, these methods will be defined and their effectiveness will be discussed.

3.1. Send-On-Delta

The send-on-delta sampling algorithm is the most natural signal-dependent strategy; in the literature, it is also known as level-crossing or deadbands sampling. In the send-on-delta technique, the sensors do not broadcast a new message if the signal remains within a certain level of confidence \bar{e}_S (resolution) (Figure 2b). The sampling criterion is defined as

$$|y(t) - y(t_k)| \geq \bar{e}_S \quad (1)$$

The ratio of events that the send-on-delta algorithm generates can not be calculated in a general way, but its average value N_S may be estimated by the following expression [30,31]:

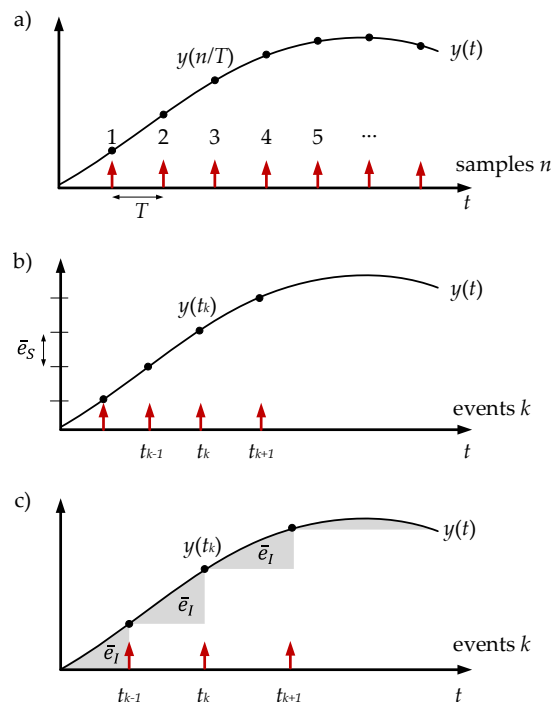


Figure 2. Sampling strategies: (a) periodic sampling; (b) send-on-delta; and (c) integral criterion.

$$N_S = \frac{1}{\overline{\Delta t}} \quad (2)$$

where $\overline{\Delta t}$ is the mean period between events considering the analysis interval (t_0, t_n)

In the send-on-delta algorithm, an event occurs when

$$|y(t_k) - y(t_{k-1})| = \bar{e}_S \quad , \quad k = 1, 2, \dots, n \quad (3)$$

and the interval time between the events $k - 1$ and k is expressed as

$$\Delta t_k = t_k - t_{k-1} = \frac{\bar{e}_S}{|\dot{y}(t_k)|} \quad (4)$$

where $\overline{|\dot{y}(t_k)|}$ is defined as

$$\overline{|\dot{y}(t_k)|} = \frac{1}{\Delta t_k} \int_{t_k}^{t_{k+1}} |\dot{y}(t)| dt \quad (5)$$

at this point, the mean period between events $\overline{\Delta t}$ can be defined as

$$\overline{\Delta t} = \frac{\sum_{k=1}^n \Delta t_k}{n} \quad (6)$$

in [31], the following relationship has been proven

$$\overline{\Delta t} = \frac{\bar{e}_S}{\overline{|\dot{y}(t)|}} \quad (7)$$

where $\overline{|\dot{y}(t)|}$ was defined as

$$\overline{|\dot{y}(t)|} = \frac{1}{t_n - t_0} \int_{t_0}^{t_n} |\dot{y}(t)| dt \quad (8)$$

taking Equations (2) and (7) into account, the mean rate of events N_S can be written as

$$N_S = \frac{\overline{|\dot{y}(t)|}}{\bar{e}_S} \quad (9)$$

In this case, the ratio of events N_S in the system depends on two parameters:

- the resolution \bar{e}_S of the sampling (the event threshold); and
- the mean of the absolute value of the first time-derivative $\overline{|\dot{y}(t)|}$ during the analysis interval.

As presented in Equation (9), the message rate in the send-on-delta strategy is a trade off between the resolution \bar{e}_S and the average slope of the signal $y(t)$.

3.2. Integral Criterion

There are some reasons to apply the event-based integral criterion in control systems. This method has a high efficiency for sampling burst signals. Likewise, this technique is a good solution in applications where a critical problem of sampling process is the accuracy of approximation of a continuous-time signal by a sequence of discrete-time samples. In this algorithm, the sampling criterion (Figure 2c) is defined by Equation (10), where \bar{e}_I is the resolution of the method:

$$\int_{t_k}^t |y(t) - y(t_k)| dt \geq \bar{e}_I \quad (10)$$

then, the mean rate of events N_I can be estimated as follows [32]:

The system generates an event when the following condition is satisfied

$$\int_{t_k}^{t_{k+1}} |y(t) - y(t_k)| dt = \bar{e}_I \quad (11)$$

taking into account the time interval between the events k and $k + 1$ ($\Delta t_k = t_{k+1} - t_k$), in [32], it has been demonstrated that the mean period between events considering the interval (t_0, t_n) is

$$\overline{\Delta t} = \frac{\sum_{k=1}^n \Delta t_k}{n} = \frac{\sqrt{2\bar{e}_I}}{\sqrt{\overline{|\dot{y}(t)|}}} \quad (12)$$

where $\sqrt{\overline{|\dot{y}(t)|}}$ is defined by

$$\sqrt{\overline{|\dot{y}(t)|}} = \frac{1}{t_n - t_0} \int_{t_0}^{t_n} \sqrt{|\dot{y}(t)|} dt \quad (13)$$

Finally, the mean rate of events N_I based on the integral criterion strategy can be expressed as:

$$N_I = \frac{1}{\Delta t} = \frac{\sqrt{|\dot{y}(t)|}}{\sqrt{2\bar{e}_I}} \quad (14)$$

Taking Equation (14) into account, the mean rate of events depends on the mean of the square root of the signal derivative absolute value and the resolution \bar{e}_I used in the algorithm

3.3. Effectiveness of Event-Based Sampling

To study how effective the presented sampling algorithms are, they will be compared with periodic sampling strategies. In the send-on-delta algorithm, the mean rate of events N_S was estimated in Equation (9). If the periodic sampling algorithm is considered, the sampling period T can be obtained by Equation (15) where the accuracy of the algorithm is \bar{e}_P

$$T = \frac{\bar{e}_P}{|\dot{y}(t)|_{max}} \quad (15)$$

Now, the ratio of samples N_P can be calculated as

$$N_P = \frac{1}{T} = \frac{|\dot{y}(t)|_{max}}{\bar{e}_P} \quad (16)$$

If the send-on-delta is compared with the periodic sampling considering the same resolution for both methods ($\bar{e}_S = \bar{e}_P$), the following equation is obtained

$$\frac{N_S}{N_P} = \frac{|\dot{y}(t)|}{|\dot{y}(t)|_{max}} \quad (17)$$

In this case, $N_S = N_P$ if $y(t)$ is a linear signal, for the rest of the continuous-time signal $N_S < N_P$ is fulfilled. The last expression implies that the send-on-delta is more efficient than the periodic sampling algorithm.

Taking into account the integral criterion, the number of samples in the periodic sampling strategy is given by (see [32]):

$$N_P = \frac{\sqrt{2\bar{e}_P}}{\sqrt{|\dot{y}(t)|_{max}}} \quad (18)$$

If both methods (event-based and periodic) have the same accuracy ($\bar{e}_I = \bar{e}_P$), the following equation is obtained from Equations (14) and (18):

$$\frac{N_I}{N_P} = \frac{\sqrt{|\dot{y}(t)|}}{\sqrt{|\dot{y}(t)|_{max}}} \quad (19)$$

which shows that for the integral criterion the event-based strategy is more efficient than the periodic sampling $N_I < N_P$ in a general way.

It can be concluded that the event-based sampling algorithms are more efficient than the periodic strategies. It means that the event-based solution generates less events than the periodic sampling scheme.

4. NCS Control Architectures

In this section, two NCS schemes are discussed. First, the classical discrete-time architecture is presented, and then the event-based solution proposed in this work is investigated in detail. In these NCS, the agents (the controller and the remote node) are connected by a wireless network.

The basic scheme of a discrete-time NCS is presented in Figure 3a. The signals $u[n]$ (control signal) and $y[n]$ (sensor signal) are sampled with the frequency f_s . The signal $u[n]$ is sent to remote node over the communication channel $Ch_1(t)$. In the remote node, this information is used to act over the actuators. The signal $y[n]$ is sent to the controller over the $Ch_2(t)$. Finally, the controller calculates $u[n]$ considering the reference signal $w[n]$ and $y[n]$. Therefore, this architecture exchanges information over the communication channels every period of time defined by $T = 1/f_s$. When the plant is in a steady state, it is not necessary to interchange information between the elements of the system because the plant does not need new control actions. However, in this scheme, the communication channels $Ch_1(t)$ and $Ch_2(t)$, the controller, the actuators and the sensors are busy every period of time T .

The proposed event-based NCS is presented in Figure 3b. As mentioned above, the proposed system follows the basic principles of an event-based control system, being the evaluation of the consumption of resources the main objective of this work. The system is composed of a controller, an event generator (EG) and a memory block (M). In the EG the signals, $y[n]$ and $w[n]$ are compared. If the difference between these signals crosses the event threshold \bar{e} , the system generates an event k and the signal $e[n_k]$ (error signal) is sent to the controller. Different methods can be applied to produce events, in [21], a review of these methodologies is presented. In a general way, the event threshold \bar{e} is defined as a constant value. This parameter can be defined as a function of the noise in the sensors or as a function of other relevant variables of the process to generate the events in a more accuracy way. In [12,14], these ideas have been explored. Therefore, the event threshold has to be set as a trade off between the accuracy of the system and the number of events. The communication channel $Ch_2(t)$ is used to send the signal $e[n_k]$ to the controller. In this case, the RF channel is occupied only when the EG generates events. On the other hand, when the signals $w[n]$ and $y[n]$ are very similar, the plant/process is in a steady state. In this case, no events occur in the system and the channel $Ch_2(t)$ is free. When the signal $u[n_k]$ arrives in the remote node it is stored in the memory M . Therefore, every time the system generates an event, the memory is updated with a new value. In the periods of time between events, the information stored in the memory is used to control the remote node. Besides, when an event is generated in the system, the controller receives the signal $e[n_k]$ and it calculates the signal $u[n_k]$. Afterwards, this control signal is sent to the remote node via $Ch_1(t)$. As result of this, the communication resources $Ch_1(t)$ and $Ch_2(t)$ are used only when the system is generating events.

In general, the communication channels $Ch_1(t)$ and $Ch_2(t)$ use Industrial, Scientific and Medical (ISM) bands which are not exclusive and many agents may be using them at the same time. When other devices are using the same channel, the interferences, the packet dropouts and an excessive delay in the network could affect the performance of the control system. To avoid these effects, a free channel in the radio link between the remote node and the controller has to be selected. For example, if the 2.4 GHz band is used, there are around 126 available channels so there is a high probability of finding interference-free channels in the wireless network.

As a conclusion, the proposed event-based solution has three main advantages:

1. The RF channels are busy only when the system generates events, see Figure 3c.
2. The controller does not need to compute control signals when the plant is in the steady state.
3. The system is protected from Zeno phenomenons. In the proposed control scheme the events are generated in the system only at certain instants of time (Figure 3c). This behaviour sets a minimum time interval between events defined by $1/f_s$. This is a typical mechanism to avoid the Zeno phenomenons in the event-based control systems [33].

In a practical way, to compare the responses of both NCS architectures (discrete-time and event-based), some conditions must be imposed:

- The clocks of the systems are synchronized (Figure 3c). This implies that the events and the samples in the system are generated at the same instant of time.
- The accuracy of both systems has to be the same. In this case, the event threshold has to be set up considering the sampling frequency of the discrete-time system and the sampling criterion in the event-based one.

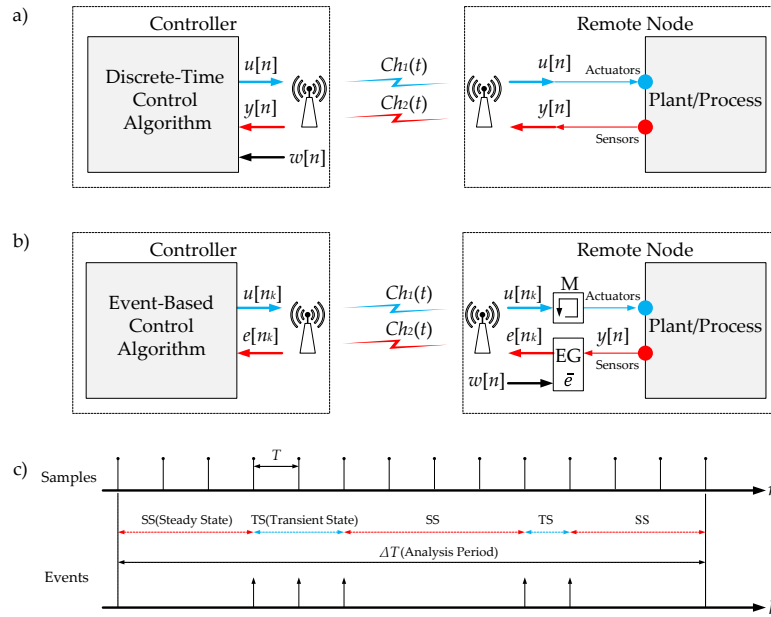


Figure 3. NCS control schemes: (a) discrete-time; (b) event-based; and (c) time diagrams. The samples represent the activity of the discrete-time system; for the event-based solution, the activity is represented by the events.

5. Resources Usage

To measure the performance of a control system, the Integral Absolute Error IAE is applied. This criterion is widely used in continuous-time and discrete-time control systems. The IAE calculates the difference between the output of the system ($y(t)$ for continuous-time or $y[n]$ for discrete-time) and the reference signal ($w(t)$ or $w[n]$) by the following equations:

$$IAE = \int_{t_0}^t |w(t) - y(t)| dt \quad (20)$$

$$IAE = \sum_{n_0}^n |w[n] - y[n]|$$

Therefore, the analysed system has a good performance if the IAE is small.

Another way to evaluate the performance of a control system is by the Integral Absolute Error compared to Periodic loop $IAEP$. This criterion is mainly used in event-based control systems [34,35]. This indicator compares the output of the event-based system $y_{eve}[n]$ with the output of its equivalent discrete-time system $y[n]$, as presented in Equation (21).

$$IAEP = \sum_{n_0}^n |y_{eve}[n] - y[n]| \quad (21)$$

Although the $IAEP$ is a good criterion to measure the performance of an event-based control system, in practice, many researchers [36–40] use the ratio of events N_N . This parameter calculates the activity of an event-based control system versus an equivalent discrete-time control system, the ration of events is given by:

$$N_N = \frac{N_{eve}}{N_{per}} \quad (22)$$

where N_{eve} is the number of events in the analysed system and N_{per} the number of samples in the equivalent discrete-time system. In this case, the number of events can be defined as $N_{eve} = N_S \Delta T$ for the send-on-delta algorithm and $N_{eve} = N_I \Delta T$ for the integral criterion, where ΔT is the analysis period.

In the same way, the number of samples is defined as $N_{per} = N_P \Delta T$ (see Section 3). In this context, if $N_N < 1$ the event-based system has less activity than the equivalent discrete-time system and consequently, it uses less resources. As was previously analysed, the event-based sampling criteria generate less events than the periodic sampling techniques using the same resolution in both methods. It means that $N_P \geq N_{EVE}$ (where $N_{EVE} = N_S$ if the send-on-delta is used or $N_{EVE} = N_I$ if the integral criterion is used (see Equations (17) and (19)). If the analysis period ΔT is taken into account, it can be written $N_{eve} = N_{EVE} \Delta T$ and $N_{per} = N_P \Delta T$, then the ratio of events is given as:

$$N_N = \frac{N_{eve}}{N_{per}} = \frac{N_{EVE}}{N_P} \quad (23)$$

where $0 \leq N_N \leq 1$.

Using the event efficiency η_N , this can be expressed as:

$$\eta_N = (1 - N_N)100 \quad (24)$$

In a general way, if the resolutions in the sampling methods are the same, the activity of the event-based control systems is more efficient than the discrete-time solutions ($0\% \geq \eta_N \geq 100\%$).

In this work, to analyse the resources usage efficiency, the ratio N_R is defined as follows

$$N_R = \frac{(R)_{eve}}{(R)_{dis}} \quad (25)$$

where R indicates which resource is analysed, $(R)_{eve}$ is the usage of the resource R in the event-based control system and $(R)_{dis}$ denotes the usage of the same resource in the equivalent discrete-time control system. In the same way, the efficiency in the usage of the resource R can be expressed as

$$\eta_R = (1 - N_R)100 \quad (26)$$

In the following sections, the usage efficiency of the resources of the control system such as bandwidth, computational load and energy are investigated.

5.1. Bandwidth Usage

The model depicted in Figure 4a has been used to analyse the bandwidth usage.

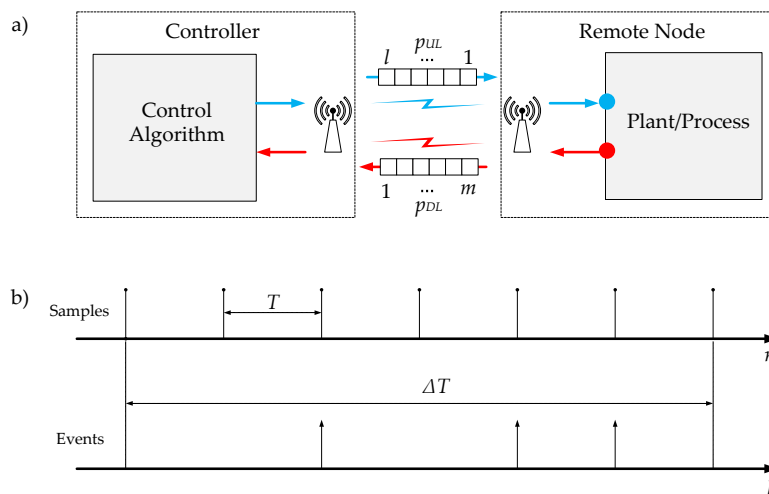


Figure 4. Bandwidth utilization model: (a) control architecture; and (b) time diagrams for discrete-time and event-based schemes.

The information in the uplink direction is $p_{UL} = l$ bits and in the downlink is $p_{DL} = m$ bits. The analysis period ΔT is considered. In this interval time, the discrete-time system generates N_{per} samples and the event-based system N_{eve} events. Then, the bandwidth usage, defined as the number of bits transmitted per second, can be written for the discrete-time system by

$$(BW_{UL})_{dis} = \frac{lN_{per}}{\Delta T}; (BW_{DL})_{dis} = \frac{mN_{per}}{\Delta T} \quad (27)$$

and for the event-based solution is given by

$$(BW_{UL})_{eve} = \frac{lN_{eve}}{\Delta T}; (BW_{DL})_{eve} = \frac{mN_{eve}}{\Delta T} \quad (28)$$

where BW_{UL} and BW_{DL} represent the bandwidth usage in the uplink and in the downlink direction, respectively.

The bandwidth usage ratio for the uplink $N_{BW_{UL}}$ and for the downlink $N_{BW_{DL}}$ can be expressed by the following equations

$$N_{BW_{UL}} = \frac{(BW_{UL})_{eve}}{(BW_{UL})_{dis}} = \frac{N_{eve}}{N_{per}} = N_N \quad (29)$$

$$N_{BW_{DL}} = \frac{(BW_{DL})_{eve}}{(BW_{DL})_{dis}} = \frac{N_{eve}}{N_{per}} = N_N \quad (30)$$

considering $N_{BW_{UL}} = N_{BW_{DL}}$, the bandwidth usage ratio for both directions N_{BW} can be defined as

$$N_{BW} = \frac{N_{eve}}{N_{per}} = N_N \quad (31)$$

and the bandwidth usage efficiency is given by

$$\eta_{BW} = \eta_N \quad (32)$$

As Equation (32) shows, the bandwidth usage efficiency is the same as the event efficiency.

5.2. Computational Load Reduction

To obtain a model of the computational load in the analysed control systems, the scheme depicted in Figure 5 has been used.

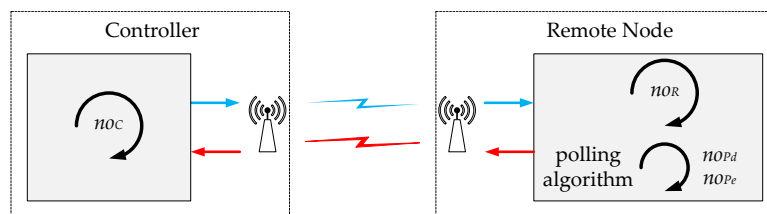


Figure 5. Computational resources model.

The control algorithm is composed of two elements: the algorithm in the controller and the algorithm in the remote node. In the remote node, the algorithm is divided in two blocks, the control algorithm and the polling algorithm. The polling block is used to get the measures from the sensors and in the event-based solution it also generates the events. In the controller, no_C operations are executed each time the algorithm is run. On the other hand, in the remote node, no_R operations are executed in the control algorithm either no_{Pd} when the discrete-time solution is used or no_{Pe} when the event-based technique is selected.

Taking into account the previous assumptions, the computational load in the discrete-time system $(CL)_{dis}$ and in the event-based control system $(CL)_{eve}$ considering the analysis period ΔT can be expressed by

$$(CL)_{dis} = \frac{(no_C + no_R + no_{Pd})N_{per}}{\Delta T} \quad (33)$$

$$(CL)_{eve} = \frac{(no_C + no_R)N_{eve} + no_{Pe}N_{per}}{\Delta T} \quad (34)$$

then, the computational load ratio N_{CL} is given by

$$N_{CL} = \frac{(CL)_{eve}}{(CL)_{dis}} = \frac{(no_C + no_R)N_N + no_{Pe}}{no_C + no_R + no_{Pd}} \quad (35)$$

and the computational load efficiency can be written as

$$\eta_{CL} = (1 - N_{CL})100 \quad (36)$$

In general, if N_N increases, the efficiency η_{CL} decreases. If the computational load of the polling algorithms is smaller than the control algorithms ($no_{Pd} \ll (no_C + no_R)$ and $no_{Pe} \ll (no_C + no_R)$), $N_{CL} \approx N_N$ and $\eta_{CL} \approx \eta_N$. On the other hand, when the load of the polling algorithms increases, the computational load ratio ($N_{CL} \rightarrow 1$) and the event-based control system does not have efficiency in computational load $\eta_{CL} \rightarrow 0\%$. As a conclusion, the computational load efficiency of the system has a high dependence on the computational load of the polling algorithms.

From a practical point of view, to evaluate the computational burden of the system a simple procedure has to be followed. First, it is necessary to distinguish which part of the algorithm is executed periodically and which one is executed eventually when the events occur in the system. Then, the parameters no_C , no_R , no_{Pd} and no_{Pe} can be obtained doing a high level analysis of the control algorithm. In Section 6.3.2 and in Appendix A, this procedure has been applied in the application used as practical example in this work.

5.3. Energy Consumption

In this section, the energy efficiency of the presented event-based control system is investigated. In Figure 6, the simplified energy model of the system is depicted.

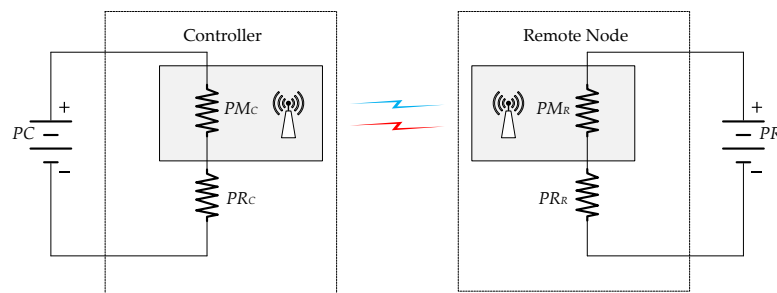


Figure 6. Energy model for the controller and for the remote node.

The electric power in the controller can be obtained adding up the power that the modem needs to transmit and receive the information PM_C and the power in the rest of the system of the controller PR_C . When the discrete-time system is considered, the power of the controller is given by

$$PC_{dis} = PM_C N_{per} + PM_C \left(\frac{1 - \beta}{\beta} \right) \quad (37)$$

where β is the power ratio defined by

$$\beta = \frac{PM_C}{PM_C + PR_C} \quad (38)$$

Using the same argument, the power of the controller when the event-based architecture is used can be written as

$$PC_{dis} = PM_C N_{eve} + PM_C \left(\frac{1 - \beta}{\beta} \right) \quad (39)$$

Taking into account the analysis period ΔT , the energy usage in the discrete-time controller $(EC)_{dis}$ and for the event-based controller $(EC)_{eve}$ are given by the following equations

$$(EC)_{dis} = \left(PM_C N_{per} + PM_C \left(\frac{1 - \beta}{\beta} \right) \right) \Delta T \quad (40)$$

$$(EC)_{eve} = \left(PM_C N_{eve} + PM_C \left(\frac{1 - \beta}{\beta} \right) \right) \Delta T \quad (41)$$

and the energy ratio in the controller N_{EC} can be expressed as

$$N_{EC} = \frac{(EC)_{eve}}{(EC)_{dis}} = \frac{N_{eve}\beta + (1 - \beta)}{N_{per}\beta + (1 - \beta)} \quad (42)$$

and the energy efficiency in the controller can be written as

$$\eta_{EC} = (1 - N_{EC})100 \quad (43)$$

Using the same reasoning in the remote node, the energy usage in the discrete-time implementation $(ER)_{dis}$ and in the event-based solution $(ER)_{eve}$ can be written as

$$(ER)_{dis} = \left(PM_R N_{per} + PM_R \left(\frac{1 - \gamma}{\gamma} \right) \right) \Delta T \quad (44)$$

$$(ER)_{eve} = \left(PM_C N_{eve} + PM_C \left(\frac{1 - \gamma}{\gamma} \right) \right) \Delta T \quad (45)$$

where $\gamma = \frac{PM_R}{PM_R + PR_R}$. Then, the energy ratio in the remote system is defined by

$$N_{ER} = \frac{(ER)_{eve}}{(ER)_{dis}} = \frac{N_{eve}\gamma + (1 - \gamma)}{N_{per}\gamma + (1 - \gamma)} \quad (46)$$

and the energy efficiency in the remote node can be written as

$$\eta_{ER} = (1 - N_{ER})100 \quad (47)$$

If the power of the modem in the controller is high ($\beta \rightarrow 1$), the efficiency in energy usage in the proposed event-based system is the event efficiency $N_{EC} \rightarrow N_N$ and $\eta_{EC} \rightarrow \eta_N$. On the other hand, when the power of this device is low ($\beta \rightarrow 0$), the system is not energy efficient, $N_{EC} \rightarrow 1$ and $\eta_{EC} \rightarrow 0\%$. Similar conclusions can be obtained in the remote node considering the parameter γ . In conclusion, the power of the modems determines the energy efficiency of the proposed event-based control scheme.

6. Experimental Results

To check the ideas presented in this work, a test laboratory to investigate wireless control systems has been developed (Figure 7). In this platform, the controller has been implemented in a laptop and

the remote nodes are the mobile robots. For this purpose, mOway mobile robots [41] have been used. Using this platform, an analysis of the behaviour of the event-based control schemes presented in this paper is carried out as well as a comparison with their equivalent discrete-time implementation.

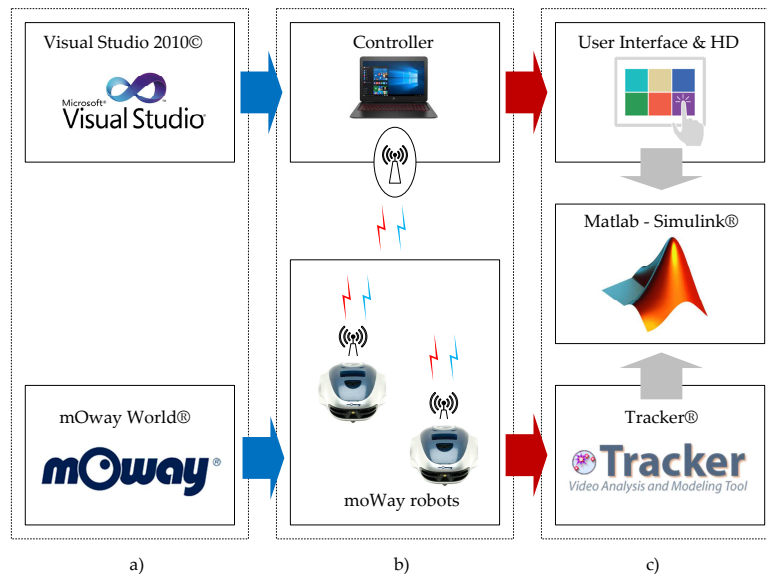


Figure 7. Laboratory for the experiments: (a) development module; (b) mobile robots environment; and (c) analysing tools.

The control algorithms have been programmed in C++ for the controller and in the mOway World environment for the robots (Figure 7a). A radio link interface is used to communicate the controller and the robots (Figure 7b). Finally, other applications such as the Tracker, the Matlab/Simulink and some scripts in the controller can be used to analyse the experimental results (Figure 7c).

The structure and components of the robots are depicted in Figure 8. The robots have four infra-red obstacle sensors with a maximum range of 3 cm and a sensor to measure the battery level (Figure 8a). The wireless communication system (robot RF interface (Figure 8b), and PC RF interface (Figure 8c)) works in the worldwide ISM frequency band at 2.400–2.4835 GHz and uses GFSK modulation. In the system 126 channels can be configured, the rate for each channel is 2 Mbps. The angular speeds of the wheels can be varied from 0 to 10.9 rad/s and their geometrical parameters $L = 6.6$ cm (distance between wheels) and $r = 1.6$ cm (radius of the wheels).

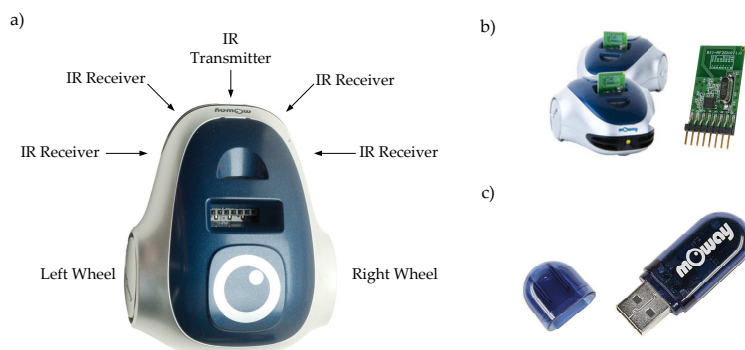


Figure 8. Robot platform: (a) components and sensor distribution; (b) robot wireless interface; and (c) PC wireless interface.

To analyse the event-based control architecture proposed in this work and its resources usage efficiency, some experiments have been set up. In these experiments robot navigation algorithms

will be checked in the laboratory. In the following sections, the proposed navigation algorithms, their implementation in the system and the experimental results will be analysed in detail.

6.1. Navigation Algorithms

In navigation applications for mobile robots, the algorithms such as Go To Goal (GTG), Obstacle Avoidance (OA) and Wall Following (WF) are widely used [42–46]. In these applications, it is also critical to define a precise positioning mechanism to guarantee the convergence of the navigation algorithm [47,48].

In this work, the behaviour and the resource usage of the OA and WF algorithms are investigated in the proposed event-based control architecture. To define the navigation algorithms, the position and the orientation of the obstacle sensors have to be taken into account (Figure 9a). The variables that contain the measurements of the sensors are defined by two indexes as follows: *ll* lateral left, *fl* front left, *fr* front right and *lr* lateral right. The parameter *bl* stores the battery level. In the actuators, the linear speeds of the wheels are *sl* speed left and *sr* speed right. Finally, the parameter *rn* (robot number) identifies the robot in the platform.

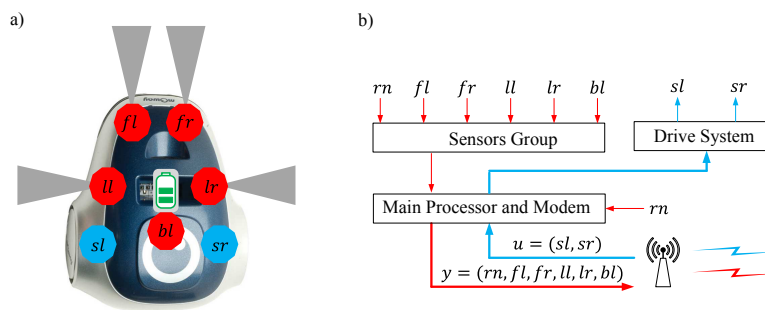


Figure 9. Sensors and actuators of the mOway robot: (a) position and orientation of the IR obstacle sensors (*fl*, *fr*, *ll*, *lr*), the battery sensor (*bl*), and the wheel actuators (*sl*, *sr*); and (b) flow diagram for the communication between sensors, actuators and the RF interface.

The sensor information $y = (rn, ll, fl, fr, lr, bl)$ and the control signals $u = (sl, sr)$ are sent and received by the RF interface, as shown in Figure 9b.

The architecture of the algorithm in the robots is depicted in Figure 10.

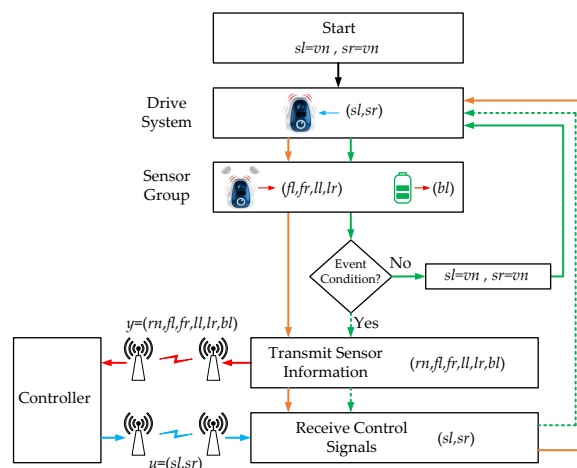


Figure 10. Robot control algorithm. The orange arrows represent the discrete-time solution and the green arrows the event-based implementation. The blocks with continuous arrows are executed each period of time *T* and the blocks with dotted arrows when an event is generated in the system.

The orange arrows represent the discrete-time implementation. The green ones are used for the event-based solution. The discrete-time algorithm works as follows:

1. The navigation speed v_n is assigned to sl and sr .
2. The speeds sl and sr are applied to the actuators.
3. The sensors group gets the information from the sensors.
4. The sensor information y is sent to the controller.
5. The control signals u are received from the controller.
6. Finally, the speeds are applied to the robot wheels.

In the event-based solution, Steps 4 and 5 are executed only if the event condition is fulfilled. Furthermore, depending on which algorithm is executing (OA or WF) the event condition will be different. In the following subsections, these aspects and the controller algorithm will be defined in detail.

6.1.1. Obstacles Avoidance Algorithm

In the event-based implementation of the OA algorithm, the event condition is given by

$$\begin{aligned} & \text{if } ((fl > \bar{e}_{OA}) \text{ OR } (fr > \bar{e}_{OA}) \text{ OR } \dots \\ & \dots (ll > \bar{e}_{OA}) \text{ OR } (lr > \bar{e}_{OA})) \\ & \quad \{event = true\} \\ & \quad \text{else } \{event = false\} \end{aligned} \quad (48)$$

where \bar{e}_{OA} is the event threshold. As shown in Equation (48), if any of the obstacle sensors (fl, fr, ll, lr) exceeds the value of the event threshold (\bar{e}_{OA}), an event is generated in the system and the robot sends the sensor information to the controller. The value of the event threshold determines the accuracy of the algorithm and the number of the events that the robot generates. If the threshold value is high, the number of events decreases at the same time as the accuracy of the algorithm does. On the other hand, there will be an inverse behaviour when the value of the event threshold decreases.

The algorithm in the controller is shown in Figure 11 and in Equation (49). In this case, the same algorithm is used for the two implementations (discrete-time and event-based).

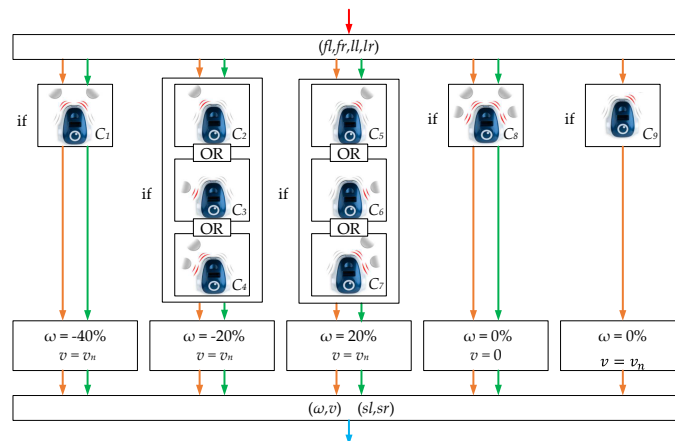


Figure 11. Obstacles avoidance algorithm in the controller. The blocks with orange arrows are executed in the discrete-time implementation and those with green arrows in the event-based one.

$$\begin{aligned} & \text{if } (C_1) \{ \omega = -40\%, v = v_n \} \\ & \text{else if } (C_2 \text{ OR } C_3 \text{ OR } C_4) \{ \omega = -20\%, v = v_n \} \\ & \text{else if } (C_5 \text{ OR } C_6 \text{ OR } C_7) \{ \omega = 20\%, v = v_n \} \\ & \text{else if } (C_8) \{ \omega = 0\%, v = 0 \} \\ & \text{else } (C_9) \{ \omega = 0\%, v = v_n \} \end{aligned} \quad (49)$$

where conditions C_1 – C_9 are defined by Equations (50)–(58)

$$\text{if } (ll = 0 \text{ AND } fl > 0 \text{ AND } fr > 0 \text{ AND } lr = 0) \\ \{C_1 = true\} \quad (50)$$

$$\text{if } (ll = 0 \text{ AND } fl > 0 \text{ AND } fr = 0 \text{ AND } lr = 0) \\ \{C_2 = true\} \quad (51)$$

$$\text{if } (ll > 0 \text{ AND } fl = 0 \text{ AND } fr = 0 \text{ AND } lr = 0) \\ \{C_3 = true\} \quad (52)$$

$$\text{if } (ll > 0 \text{ AND } fl > 0 \text{ AND } fr = 0 \text{ AND } lr = 0) \\ \{C_4 = true\} \quad (53)$$

$$\text{if } (ll = 0 \text{ AND } fl = 0 \text{ AND } fr > 0 \text{ AND } lr = 0) \\ \{C_5 = true\} \quad (54)$$

$$\text{if } (ll = 0 \text{ AND } fl = 0 \text{ AND } fr = 0 \text{ AND } lr > 0) \\ \{C_6 = true\} \quad (55)$$

$$\text{if } (ll = 0 \text{ AND } fl = 0 \text{ AND } fr > 0 \text{ AND } lr > 0) \\ \{C_7 = true\} \quad (56)$$

$$\text{if } (ll > 0 \text{ AND } fl > 0 \text{ AND } fr > 0 \text{ AND } lr > 0) \\ \{C_8 = true\} \quad (57)$$

$$\text{if } (ll = 0 \text{ AND } fl = 0 \text{ AND } fr = 0 \text{ AND } lr = 0) \\ \{C_9 = true\} \quad (58)$$

In Figure 11, the orange arrows represent the discrete-time solution and the green arrows the proposed event-based control algorithm. This algorithm is parametrized by the linear speed v and the angular speed ω . These magnitudes are transformed into wheel speeds sl and sr by

$$sl = v - \omega \frac{L}{2} \omega_{max} \quad (59)$$

$$sr = v + \omega \frac{L}{2} \omega_{max} \quad (60)$$

where ω is expressed as a percentage of the angular speed and ω_{max} is the maximum angular speed of the robot. Notice that, in this control algorithm, condition C_9 is only executed in the discrete-time implementation.

As presented in Figure 11, the algorithm receives from the robot the sensor information (fl, fr, ll, lr). In the discrete-time implementation, this information is received every period of time $T = 1/f_s$ and in the event-based solution when an event is generated in the robot. Depending on the values of the sensors (conditions C_1 to C_9), different control actions are taken. In this implementation, $\omega > 0$ represents a turn of the robot to the left, on the contrary $\omega < 0$ does the robot turns to the right. For example, if the right sensors detect an obstacle (the conditions C_5 OR C_6 OR C_7 is fulfilled) the robot must turn to the left. In this case, the control law is defined as $\omega = 20\%$ (turn to left) and $v = v_n$ (maintain constant linear velocity).

6.1.2. Wall Following Algorithm

Two variants of the WF algorithm are usually implemented: clockwise (CW) or counter-clockwise (CCW) [45]. In this work, the second option has been selected. The event condition in the algorithm is defined by

$$\begin{aligned} & \text{if } ((fl > 0) \text{ OR } (abs(ll - w) > \bar{e}_{WF})) \\ & \quad \{event = true\} \\ & \text{else } \{event = false\} \end{aligned} \quad (61)$$

where \bar{e}_{WF} represents the event-threshold and the parameter w represents the target distance between the robot and the wall. As presented in Equation (61), if there is an obstacle in front of the robot ($fl > 0$) or the distance from the robot to the wall exceeds the event threshold ($abs(ll - w) > \bar{e}_{WF}$), an event is generated in the robot.

The controller algorithm, which is the same in both architectures (discrete-time and event-based), is defined by

$$\begin{aligned} & \text{if } ((fl > 0)) \\ & \quad \{sl = v_n, sr = 0\} \\ & \text{else } \{sl = (ll - w)v_w + v_n, sr = v_n\} \end{aligned} \quad (62)$$

where v_w represents the approach speed to the wall. In this case, the control law is the same for both implementations (discrete-time and event-based). In this algorithm, when there is an obstacle in front of the robot ($fl \geq 0$), the left wheel rotates with a constant speed ($sl = v_n$) and the right wheel stops ($sr = 0$). In any other situation, the speed of the right wheel remains constant ($sr = v_n$) and the right wheel is modulated according to v_w to maintain a constant distance w to the wall.

6.2. System Activity

In this work, two experiments have been set up, the OA algorithm has been checked in experiment 1 and the FW in experiment 2. In both experiments, the discrete-time architecture was implemented in robot number 1 ($rn = 1$) and the event-based one in robot number 2 ($rn = 2$).

The discrete-time system works with a sampling frequency of 10 Hz, and the event-based system uses the send-on-delta sampling method. The obstacles sensors have a maximum change rate (mcr) of 3 cm/s. To compare the efficiency of both systems (discrete-time and event-based), they must have the same accuracy. In other words, the precision of the discrete-time system \bar{e}_p has to be the same as \bar{e}_s (see Section 3.3). In this case, $\bar{e}_p = mcr / f_s$ and \bar{e}_s is defined as \bar{e}_{OA} for the OA algorithm and as \bar{e}_{WF} for the WF algorithm. Taking into account the previous assumptions, the event thresholds were set up to $\bar{e}_{OA} = \bar{e}_{WF} = \bar{e}_p = mcr / f_s = 0.3$ cm.

The parameters of the algorithms are presented in Tables 1 and 2.

Table 1. Experiment 1: obstacles avoidance algorithm.

Architecture	rn	v_n (cm/s)	\bar{e}_{OA} (cm)
Discrete-time	1	12	-
Event-based	2	12	0.3

Table 2. Experiment 2: wall following algorithm.

Architecture	rn	v_n (cm/s)	w (cm)	v_w (1/s)	\bar{e}_{WF} (cm)
Discrete-time	1	12	1.5	-	-
Event-based	2	12	1.5	6	0.3

In the experiments, the responses of the systems were analysed during fifteen minutes ($\Delta T = 15$ min). To analyse the activity of the control scheme, the number of samples N_{per} in the discrete-time system and the number of events N_{eve} in the proposed architecture have been measured.

In Figures 12 and 13, some snapshots of the experiment are shown. In both experiments, the robots show a stable behaviour. The control scheme solve the navigation problem in the discrete-time solution $rn = 1$ as in the event-based implementation $rn = 2$.

In Figure 14, the activity of the systems is presented.

In both experiments, the number of samples N_{per} in the discrete-time architecture is the same (Figure 14a,b). On the other hand, the number of events N_{eve} is always smaller than the number of samples N_{per} in Equation (17) (Figure 14c,d), as demonstrated in Section 3.3. In this case, the event efficiency η_N for the 15 min experiment is 63% for the OA and 17% for WF algorithm, as presented in Figure 14e,f and in Table 3.

Table 3. Algorithm activity by architecture, number of samples, number of events, ratio of events and event efficiency.

Algorithm	N_{per}	N_{eve}	N_N	η_N
Obstacles Avoidance	9544	3548	0.37	63%
Wall Following	9544	7889	0.83	17%

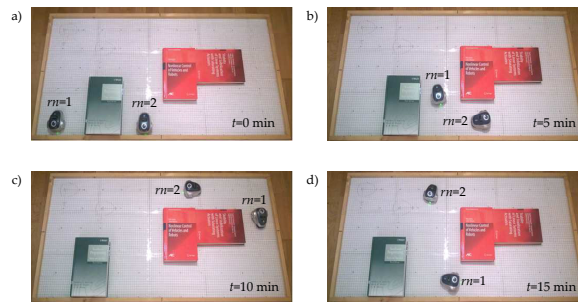


Figure 12. Snapshots of experiment 1. Positions of the robots at: (a) $t = 0$ min; (b) $t = 5$ min; (c) $t = 10$ min; and (d) $t = 15$ min.



Figure 13. Snapshots of experiment 2. Positions of the robots at: (a) $t = 0$ min; (b) $t = 5$ min; (c) $t = 10$ min; and (d) $t = 15$ min.

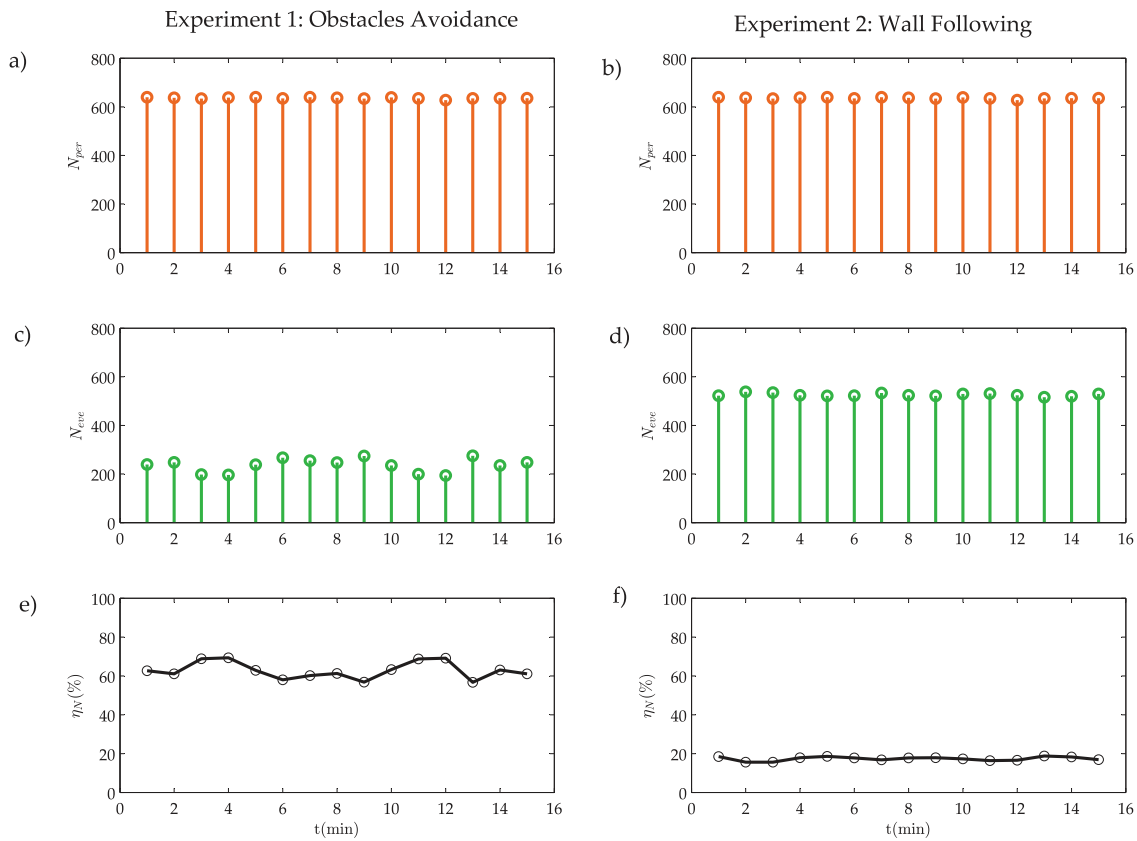


Figure 14. Activity of the systems: number of samples N_{per} (a,b); number of events N_{eve} (c,d); and event efficiency η_N (e,f).

6.3. Resource Efficiency

In this section, the RF bandwidth, the computational load and the energy consumption in both architectures have been analysed. At the same time, the experimental results are discussed.

6.3.1. RF bandwidth

In each sensor and in each actuator, the transmitted information is a 8 bit code. Taking into account this assumption, the RF uplink (from controller to robot) uses 16 bits and the RF downlink (from robot to controller) needs 48 bits to send a packet of information (see Figure 10). Some additional bits also have to be included to manage the radio interface. In this case, the downlink is the critical link because it needs most of the bandwidth.

The downlink (DL) bandwidth for the two experiments is depicted in Figure 15a,b.

As demonstrated in Section 5.1, the efficiency in the bandwidth usage η_{BW} (Figure 15e,f) is the same as the event efficiency η_N (32) (Figure 15c,d). The average bandwidths for each architecture ($(\overline{BW}_{DL})_{dis}$, $(\overline{BW}_{DL})_{eve}$), the bandwidth usage ratio N_{BW} and the bandwidth usage efficiency η_{BW} are presented in Table 4.

Table 4. Bandwidth efficiency.

Algorithm	$(\overline{BW}_{DL})_{dis}$	$(\overline{BW}_{DL})_{eve}$	N_{BW}	η_{BW}
Obstacles Avoidance	509 bps	189 bps	0.37	63%
Wall Following	509 bps	421 bps	0.83	17%

Taking into account these results, the bandwidth efficiency is 63% for the OA algorithm and 17% for the WF one.

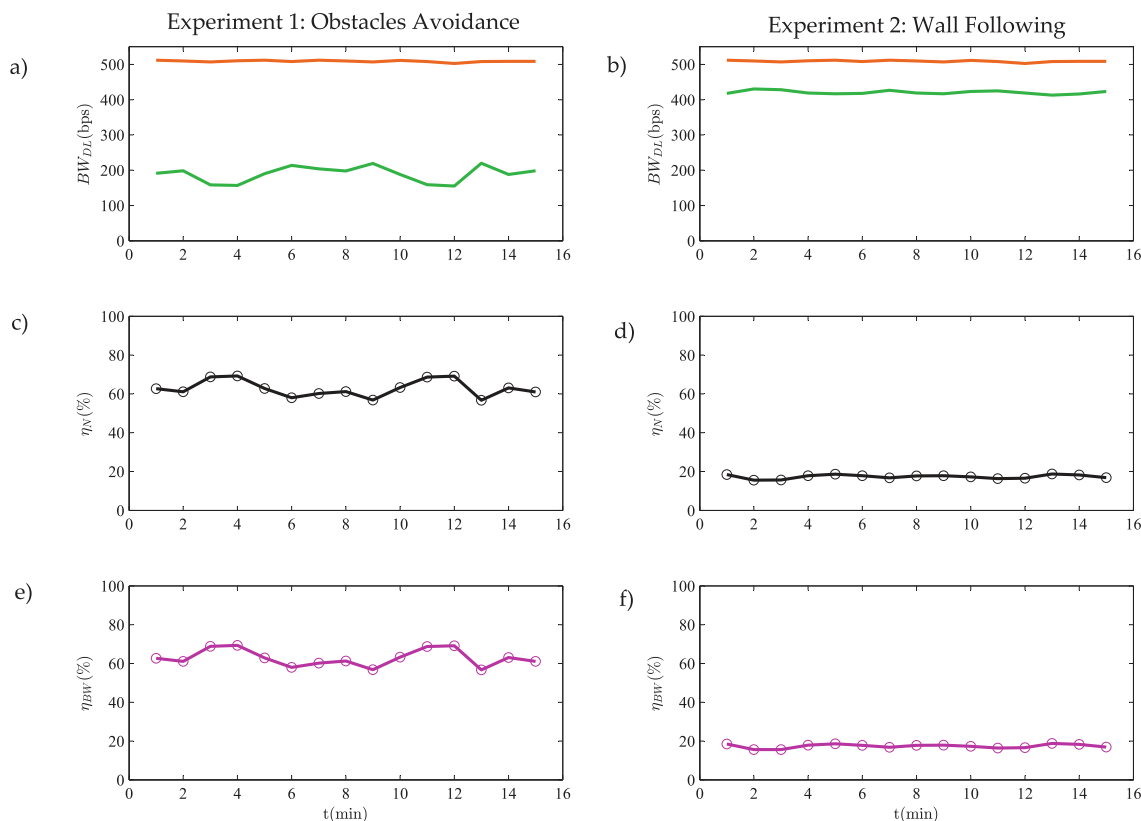


Figure 15. Bandwidth efficiency: (a,b) download bandwidth, where orange line represents the discrete-time architecture and the green line the event-based architecture; (c,d) event efficiency η_N ; and (e,f) download bandwidth efficiency η_{BW} .

6.3.2. Computational Load

To estimate the computational load in both experiments, a high level analysis of the algorithms has been performed. In Table 5, the number of operations for each algorithm is presented.

Table 5. Algorithm operations.

a) OA Algorithm				
Architecture	no_C (ops)	no_R (ops)	no_{Pd} (ops)	no_{Pe} (ops)
Discrete-time	11	4	5	-
Event-based	11	10	-	3
b) WF Algorithm				
Architecture	no_C (ops)	no_R (ops)	no_{Pd} (ops)	no_{Pe} (ops)
Discrete-time	5	4	5	-
Event-based	5	10	-	3

The parameters no_C , no_R , no_{Pd} and no_{Pe} have been calculated taking into account the high level representation of the algorithms of the robots and the controllers (see Figure 5 and Appendix A). Considering Table 5 and the activity of the system (N_{per} and N_{eve}), the computational load for each experiment can be calculated by Equations (33)–(35).

The results for experiments 1 and 2 are depicted in Figure 16.

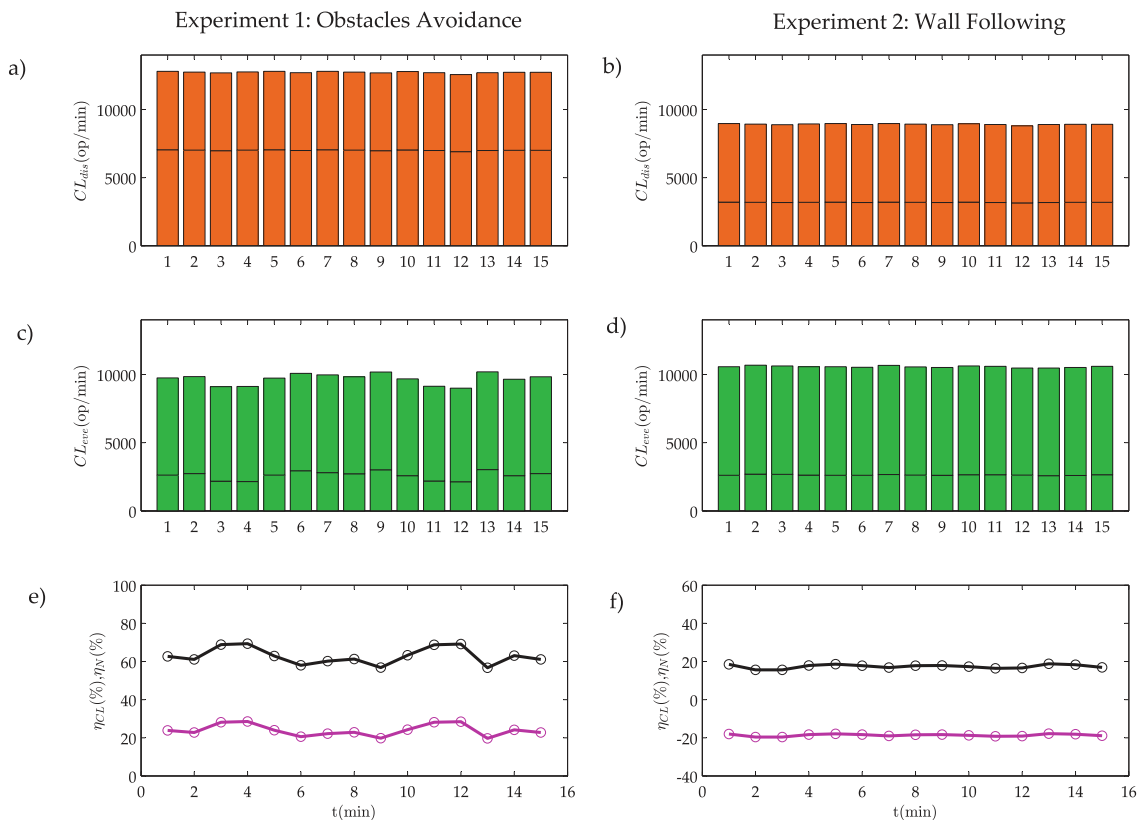


Figure 16. Computational load: (a,b) Computational load in the discrete-time implementation, where the bottom bars represent the controller and the top bars the robot; (c,d) computational load in the event-based implementation, where the bottom bars represent the controller and the top bars the robot; and (e,f) event efficiency η_N (black line) and computational load efficiency η_{CL} (pink line).

In experiment 1, the computational load is lower in the event-based implementation than in the discrete-time one (Figure 16a,c). In this case, the computational load efficiency η_{CL} reaches 24% (Figure 16e and Table 6). On the other hand, in experiment 2, the computational load in the event-based solution is higher than the discrete-time one (Figure 16b,d). In this case, the computational load efficiency is -19% (Figure 16f and Table 6).

Table 6. Computational load efficiency.

Algorithm	$(\overline{CL})_{dis}$	$(\overline{CL})_{eve}$	N_{CL}	η_{CL}
Obstacles Avoidance	12,725 ops/min	9674 ops/min	0.76	24%
Wall Following	8907 ops/min	10,570 ops/min	1.19	-19%

As discussed in Section 5.2, the computational load has a high dependence on the ratio of events N_N and the polling algorithms. In these experiments, the polling algorithms have a small computational load (see Table 5), but the ratio of events is very large especially in the WF algorithm (see Table 3). This is the main reason why the efficiency in WF algorithm is negative. Therefore, the proposed system does not present good results for computational load in the WF algorithm. In this case, the solution could be to modify the event threshold \bar{e}_{WF} to improve the efficiency with the inconvenience of reducing the accuracy of the system.

6.3.3. Energy Consumption

In these experiments, the controller has been implemented in a laptop with Windows OS. In this system, it is very complicated to measure the consumed energy by the controller and therefore this consumption has not been considered. On the other hand, in the mOway robot measuring this energy is easy by using the battery level sensor *bl*.

The results of the experiments are depicted in Figure 17.

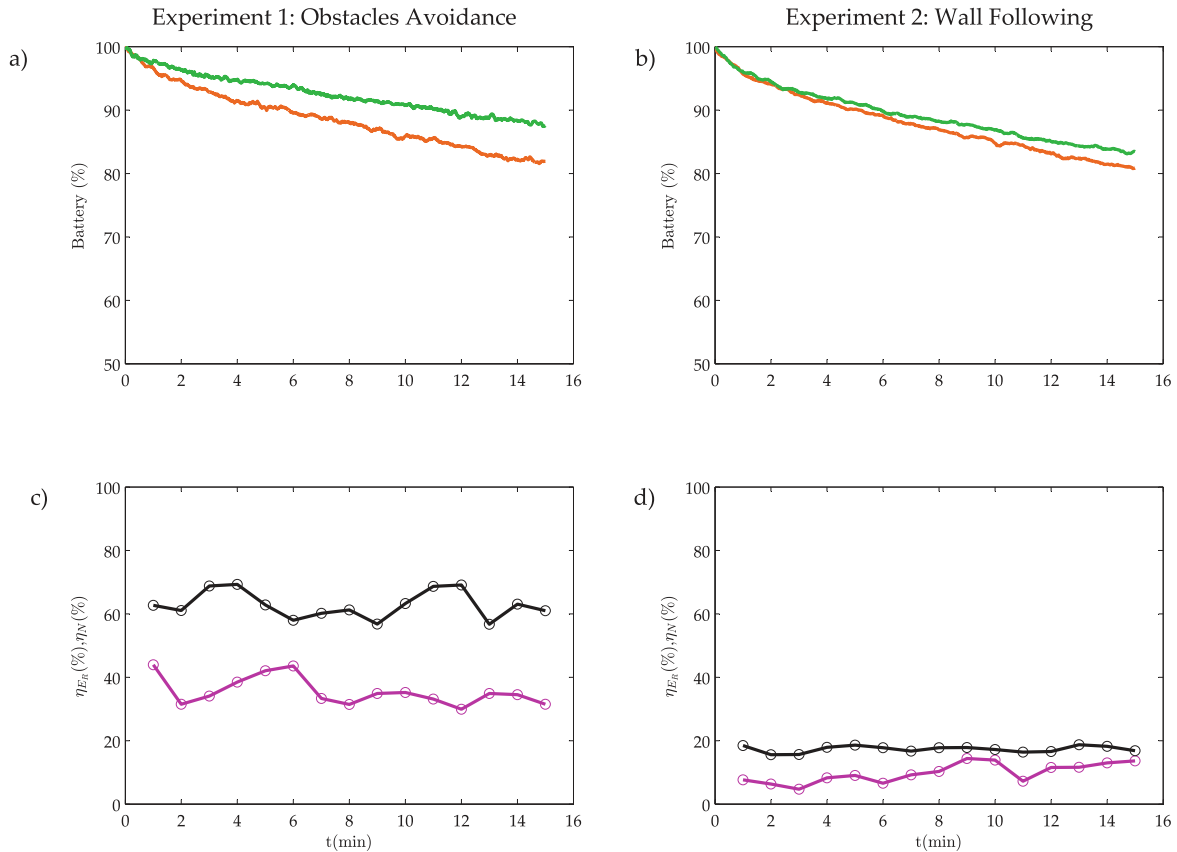


Figure 17. Energy consumption: (a,b) battery level, where the orange line represents the battery level in the discrete-time robot, and the green line in the event-based robot; and (c,d) event efficiency η_N (black line) and robot energy efficiency η_{CL} (pink line).

As shown in Figure 17a,b, when both experiments end, the battery level in the proposed event-based architecture is higher than in the discrete-time implementation, which means that the proposed system uses less energy than the classical discrete-time solution. The energy used in the robot can be estimated in the discrete-time solution as

$$(ER)_{dis} = 100\% - (Battery(\%))_{dis} \quad (63)$$

and in the event-based architecture by

$$(ER)_{eve} = 100\% - (Battery(\%))_{eve} \quad (64)$$

where $(Battery(\%))_{dis}$ denotes the level of battery measured in sensor *bl* for the discrete-time solution and $(Battery(\%))_{eve}$ for the event-based one.

In these experiments, the energy ratio in the robot can be obtained directly by $N_{ER} = \frac{(ER)_{eve}}{(ER)_{dis}}$ and energy efficiency by $\eta_{ER} = (1 - N_{ER})100$. The results are presented in Figure 17c,d and Table 7.

Table 7. Energy efficiency.

Algorithm	$(ER)_{dis}$	$(ER)_{eve}$	N_{ER}	η_{ER}
Obstacles Avoidance	18.09%	12.48%	0.69	31%
Wall Following	19.18%	16.56%	0.86	14%

In this case, the energy efficiency is 31% for experiment 1 and 14% for experiment 2. The energy consumed by the robot could be directly obtained using the measures of the sensor *bl* and this has allowed to calculate the energy efficiency directly. On the other hand, using the model developed in Section 5.3, the power ratio γ can be obtained. In these experiments $\gamma < 1\%$, this means that the modem of the robot consumes less than 1% of the robots energy. This value is extremely small because the modem has very little power (< 1 mW) and consequently a short range (less than 20 m).

As analysed previously, if the power of the modem increases, the efficiency also increases. In these examples, if the power of the modem is increased (e.g., 100 mW which implies a range of 1 km, $\gamma = 63\%$) the efficiency in the OA algorithm increases from 31% to 62% and in the WF algorithm from 14% to 17%.

7. Conclusions and Future Work

The event-based control architectures presented in this work can be an alternative to the classical discrete-time control systems. The features of these new control schemes help to manage the resources of the system under optimal conditions because they present high efficiency in the resource usage. In a general way, by using a sampling criterion such as the send-on-delta or the integral criterion, the activity of the presented solution can be reduced. In this paper, some new methods have been proposed to analyse the resources usage and the criteria to minimize their consumption. Thus, it can be concluded that these new schemes use fewer resources such as bandwidth, computational load and energy than the classical ones. The ideas presented in this work have been applied to an NCS for mobile robots as a practical approach. The navigation problem was solved using this new control paradigm and it has been compared with a classical discrete-time solution. Finally, the experimental results have demonstrated the stability and the efficiency in the resources usage of the proposed event-based control architecture if it is compared with classical control schemes.

In this work, the effects of the delays, the packet dropouts and the packet disorders in the network have not been analysed. To avoid these undesirable effects that the network produces in the proposed control system a free RF channel is selected to communicate the robots and the controller. This reduces the interference that other agents using the same frequency can produce. To improve the proposed system, a logic-like trigger [49] or other similar ideas could be applied in a simple way. To apply these methods, two main aspects must be previously developed. First, it is necessary to find a stochastic model of the mobile robots. Then, an exhaustive analysis of the network with different levels of congestion must be performed. With this model, an estimation of the delay between the packets and the volume of the disordered packets can be obtained. Finally, once these models have been developed a network-based H_∞ filtering using a logic jumping-like trigger could be applied.

As a future work, the proposed strategy and the ideas presented in this paper will be analysed in systems such as Unmanned Aerial Vehicles (UAVs), Autonomous Underwater Vehicles (AUVs) or Legged Mobile Robots (LMRs). Furthermore, new strategies to ensure the stability and the efficiency of these systems will be investigated.

Acknowledgments: This work was supported in part by the UNED project GID2016-6-1, the Spanish Ministry of Economy and Competitiveness under Projects DPI2014-55932-C2-2-R and ENE2015-64914-C3-2-R and FEDER funds.

Author Contributions: Rafael Socas designed the system, performed the experiments, analysed the results and prepared the first draft of the manuscript. Raquel Dormido performed the theoretical conception and planning of the system. The manuscript was revised and modified by Sebastián Dormido.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Pseudocodes of Control Algorithms

In this appendix, the pseudocodes of the following algorithms are presented:

- Algorithm 1: Code for the discrete-time robot, which is used for Obstacle Avoidance and Wall Following navigation algorithms.
- Algorithm 2: Code for the event-based robot, which is used for Obstacle Avoidance and Wall Following navigation algorithms.
- Algorithm 3: Code for the Obstacle Avoidance navigation algorithm, which is used in both architectures (discrete-time and event-based).
- Algorithm 4: Code for the Wall Following navigation algorithm, which is used in both architectures (discrete-time and event-based).

Algorithm A1 Pseudocode in the discrete-time robot

```

assign  $sl \leftarrow v_n$ 
assign  $sr \leftarrow v_n$ 
while (true) do
{
1: assign left wheel speed  $\leftarrow sl$ 
2: assign right wheel speed  $\leftarrow sr$ 
3: measure obstacle front left sensor  $\rightarrow fl$ 
4: measure obstacle front right sensor  $\rightarrow fr$ 
5: measure obstacle lateral left sensor  $\rightarrow ll$ 
6: measure obstacle lateral right sensor  $\rightarrow lr$ 
7: measure battery level  $\rightarrow bl$ 
8: transmit sensor information ( $rn, fl, fr, ll, lr, bl$ )
9: receive control information ( $sl, sr$ )
}

```

Algorithm A2 Pseudocode in the event-based robot

```

assign  $sl \leftarrow v_n$ 
assign  $sr \leftarrow v_n$ 
while (true) do
{
1: assign left wheel speed  $\leftarrow sl$ 
2: assign right wheel speed  $\leftarrow sr$ 
3: measure obstacle front left sensor  $\rightarrow fl$ 
4: measure obstacle front right sensor  $\rightarrow fr$ 
5: measure obstacle lateral left sensor  $\rightarrow ll$ 
6: measure obstacle lateral right sensor  $\rightarrow lr$ 
7: measure battery level  $\rightarrow bl$ 
8: if (event condition==true)
{
event_code()
}
else
{
9: assign  $sl \leftarrow v_n$ 
10: assign  $sr \leftarrow v_n$ 
}
}

event_code()
{
1: transmit sensor information ( $rn, fl, fr, ll, lr, bl$ )
2: receive control information ( $sl, sr$ )
3: return()
}

```

Algorithm A3 Pseudocode of the OA algorithm in the controller

- 1: **receive** obstacle sensor information (fl, fr, ll, lr)
 - 2: **check** condition C_1
 - 3: **check** conditions C_2, C_3, C_4
 - 4: **check** conditions C_5, C_6, C_7
 - 5: **check** condition C_8
 - 6: **check** condition C_9 // only in the discrete-time scheme
 - 7: **calculate** ω
 - 8: **calculate** v
 - 9: **transform** (ω, v) to sl
 - 10: **transform** (ω, v) to sr
 - 11: **transmit control information** (sl, sr)
-

Algorithm A4 Pseudocode of the FW algorithm in the controller

- 1: **receive** obstacle sensor information (fl, fr, ll, lr)
 - 2: **check** condition $(fl > 0)$
 - 3: **calculate** sl
 - 4: **calculate** sr
 - 5: **transmit** control information (sl, sr)
-

References

1. Guinaldo, M.; Sánchez, J.; Dormido, S. Control en red basado en eventos: De lo centralizado a lo distribuido. *Revista Iberoamericana de Automática e Informática Industrial RIAI* **2017**, *14*, 16–30.
2. Mansano, R.K.; Godoy, E.P.; Porto, A.J. The Benefits of Soft Sensor and Multi-Rate Control for the Implementation of Wireless Networked Control Systems. *Sensors* **2014**, *14*, 24441–24461.
3. Roohi, M.H.; Ghaisari, J.; Izadi, I.; Saidi, H. Discrete-time event-triggered control for wireless networks: Design and network calculus analysis. In Proceedings of the 2015 International Conference on Event-Based Control, Communication, and Signal Processing (EBCCSP), Krakow, Poland, 17–19 June 2015; pp. 1–8.
4. Cloosterman, M.B.; Hetel, L.; Van de Wouw, N.; Heemels, W.; Daafouz, J.; Nijmeijer, H. Controller synthesis for networked control systems. *Automatica* **2010**, *46*, 1584–1594.
5. Montestruque, L.A.; Antsaklis, P.J. State and output feedback control in model-based networked control systems. In Proceedings of the 41st IEEE Conference on Decision and Control, Las Vegas, NV, USA, 10–13 December 2002; Volume 2, pp. 1620–1625.
6. Montestruque, L.A.; Antsaklis, P.J. On the model-based control of networked systems. *Automatica* **2003**, *39*, 1837–1843.
7. Sun, Y.; El-Farra, N.H. Quasi-decentralized model-based networked control of process systems. *Comput. Chem. Eng.* **2008**, *32*, 2016–2029.
8. Tabuada, P. Event-triggered real-time scheduling of stabilizing control tasks. *IEEE Trans. Autom. Control* **2007**, *52*, 1680–1685.
9. Wang, X.; Lemmon, M.D. Event-triggering in distributed networked control systems. *IEEE Trans. Autom. Control* **2011**, *56*, 586–601.
10. Garcia, E.; Antsaklis, P.J. Decentralized model-based event-triggered control of networked systems. In Proceedings of the 2012 American Control Conference (ACC), Montreal, QC, Canada, 27–29 June 2012; pp. 6485–6490.
11. Guinaldo, M.; Dimarogonas, D.V.; Johansson, K.H.; Sánchez, J.; Dormido, S. Distributed event-based control strategies for interconnected linear systems. *IET Control Theory Appl.* **2013**, *7*, 877–886.
12. Romero, J.A.; Pascual, N.J.; Peñarocha, I.; Sanchis, R. Event-Based PI controller with adaptive thresholds. In Proceedings of the 2012 4th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), St. Petersburg, Russia, 3–5 October 2012; pp. 219–226.

13. Molin, A.; Hirche, S. Adaptive event-triggered control over a shared network. In Proceedings of the 2012 IEEE 51st IEEE Conference on Decision and Control (CDC), Maui, HI, USA, 10–13 December 2012; pp. 6591–6596.
14. Socas, R.; Dormido, S.; Dormido, R. Event-based controller for noisy environments. In Proceedings of the 2014 Second World Conference on Complex Systems (WCCS), Agadir, Morocco, 10–12 November 2014; pp. 280–285.
15. Anta, A.; Tabuada, P. To sample or not to sample: Self-triggered control for nonlinear systems. *IEEE Trans. Autom. Control* **2010**, *55*, 2030–2042.
16. Wang, X.; Lemmon, M.D. Event-triggering in distributed networked systems with data dropouts and delays. In *International Workshop on Hybrid Systems: Computation and Control*; Springer: Berlin, Germany, 2009; pp. 366–380.
17. Mazo, M.; Tabuada, P. Decentralized event-triggered control over wireless sensor/actuator networks. *IEEE Trans. Autom. Control* **2011**, *56*, 2456–2461.
18. Dimarogonas, D.V.; Johansson, K.H. Event-triggered control for multi-agent systems. In Proceedings of the 48th IEEE Conference on Decision and Control, 2009 Held Jointly with the 2009 28th Chinese Control Conference, Shanghai, China, 15–18 December 2009; pp. 7131–7136.
19. Liu, J.; Yue, D. Event-triggering in networked systems with probabilistic sensor and actuator faults. *Inf. Sci.* **2013**, *240*, 145–160.
20. Zhang, X.M.; Han, Q.L.; Zhang, B.L. An overview and deep investigation on sampled-data-based event-triggered control and filtering for networked systems. *IEEE Trans. Ind. Inform.* **2017**, *13*, 4–16.
21. Dormido, S.; Sánchez, J.; Kofman, E. Muestreo, control y comunicación basados en eventos. *Revista Iberoamericana de Automática e Informática Industrial RIAI* **2008**, *5*, 5–26.
22. Socas, R.; Dormido, S.; Dormido, R. Event-based control strategy for the guidance of the Aerosonde UAV. In Proceedings of the 2015 European Conference on Mobile Robots (ECMR), Lincoln, UK, 2–4 September 2015; pp. 1–6.
23. Nguyen, V.H.; Suh, Y.S. Networked estimation with an area-triggered transmission method. *Sensors* **2008**, *8*, 897–909.
24. Pawlowski, A.; Guzman, J.L.; Rodríguez, F.; Berenguel, M.; Sánchez, J.; Dormido, S. Simulation of greenhouse climate monitoring and control with wireless sensor network and event-based control. *Sensors* **2009**, *9*, 232–252.
25. Miskowicz, M. Efficiency of event-based sampling according to error energy criterion. *Sensors* **2010**, *10*, 2242–2261.
26. Li, S.; Sauter, D.; Xu, B. Fault isolation filter for networked control system with event-triggered sampling scheme. *Sensors* **2011**, *11*, 557–572.
27. Sánchez, J.; Visioli, A.; Dormido, S. Event-based PID control. In *PID Control in the Third Millennium*; Springer: Berlin, Germany, 2012; pp. 495–526.
28. Åström, K.J. Event based control. *Anal. Des. Nonlinear Control Syst.* **2008**, *3*, 127–147.
29. Lunze, J.; Lehmann, D. A state-feedback approach to event-based control. *Automatica* **2010**, *46*, 211–215.
30. Miskowicz, M. Send-on-delta concept: An event-based data reporting strategy. *Sensors* **2006**, *6*, 49–63.
31. Miskowicz, M. Analytical approximation of the uniform magnitude-driven sampling effectiveness. In Proceedings of the 2004 IEEE International Symposium on Industrial Electronics, Ajaccio, France, 4–7 May 2004; Volume 1, pp. 407–410.
32. Miskowicz, M. Asymptotic effectiveness of the event-based sampling according to the integral criterion. *Sensors* **2007**, *7*, 16–37.
33. Zhu, W.; Jiang, Z.P. Event-based leader-following consensus of multi-agent systems with input time delay. *IEEE Trans. Autom. Control* **2015**, *60*, 1362–1367.
34. Vasyutynskyy, V.; Kabitzsch, K. Simple PID control algorithm adapted to deadband sampling. In Proceedings of the 2007 ETFA. IEEE Conference on Emerging Technologies and Factory Automation, Patras, Greece, 25–28 September 2007; pp. 932–940.
35. Otanez, P.G.; Moyne, J.R.; Tilbury, D.M. Using deadbands to reduce communication in networked control systems. In Proceedings of the 2002 American Control Conference, Anchorage, AK, USA, 8–10 May 2002; Volume 4, pp. 3015–3020.

36. Miskowicz, M. Improving the performance of the networked control system using event-triggered observations. *PDS'2004* **2004**, *37*, 53–58.
37. Pawlowski, A.; Guzmán, J.L.; Rodríguez, F.; Berenguel, M.; Sánchez, J.; Dormido, S. The influence of event-based sampling techniques on data transmission and control performance. In Proceedings of the 2009 ETFA 2009. IEEE Conference on Emerging Technologies & Factory Automation, Mallorca, Spain, 22–25 September 2009; pp. 1–8.
38. Lian, F.L.; Yook, J.K.; Tilbury, D.M.; Moyne, J. Network architecture and communication modules for guaranteeing acceptable control and communication performance for networked multi-agent systems. *IEEE Trans. Ind. Inform.* **2006**, *2*, 12–24.
39. Nguyen, V.H.; Suh, Y.S. A modified multirate controller for networked control systems with a send-on-delta transmission method. In *Advanced Intelligent Computing Theories and Applications. With Aspects of Theoretical and Methodological Issues*; Springer: Berlin, Germany, 2007; pp. 304–315.
40. Yook, J.K.; Tilbury, D.M.; Soparkar, N.R. Trading computation for bandwidth: Reducing communication in distributed control systems using state estimators. *IEEE Trans. Control Syst. Technol.* **2002**, *10*, 503–518.
41. Valera, A.; Soriano, A.; Vallés, M. Plataformas de Bajo Coste para la Realización de Trabajos Prácticos de Mecatrónica y Robótica. *Revista Iberoamericana de Automática e Informática Industrial RIAI* **2014**, *11*, 363–376.
42. Siegwart, R.; Nourbakhsh, I.R.; Scaramuzza, D. *Introduction to Autonomous Mobile Robots*; MIT Press: Cambridge, MA, USA, 2011.
43. Nehmzow, U. *Robot Behaviour: Design, Description, Analysis and Modelling*; Springer Science & Business Media: Berlin, Germany, 2008.
44. Nehmzow, U. *Mobile Robotics: A Practical Introduction*; Springer Science & Business Media: Berlin, Germany, 2012.
45. Socas, R.; Dormido, S.; Dormido, R.; Fabregas, E. Event-based control strategy for mobile robots in wireless environments. *Sensors* **2015**, *15*, 30076–30092.
46. Socas, R.; Dormido, S.; Dormido, R. Optimal Threshold Setting for Event-Based Control Strategies. *IEEE Access* **2017**, *5*, 2880–2893.
47. Socas, R.; Dormido, S.; Dormido, R.; Fábregas, E. 3D positioning algorithm for low cost mobile robots. In Proceedings of the 2015 12th International Conference on Informatics in Control, Automation and Robotics (ICINCO), Colmar, France, 21–23 July 2015; Volume 2, pp. 5–14.
48. Socas, R.; Dormido, S.; Dormido, R.; Fabregas, E. Improving the 3D positioning for low cost mobile robots. In Proceedings of the Informatics in Control, Automation and Robotics 12th International Conference, Colmar, France, 21–23 July 2015; *Revised Selected Papers*; Springer: Berlin, Germany, 2016; pp. 97–114.
49. Zhang, X.M.; Han, Q.L. Network-based H_∞ filtering using a logic jumping-like trigger. *Automatica* **2013**, *49*, 1428–1435.



© 2018 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).