Sensors **2013**, *13*, 9396-9413; doi:10.3390/s130709396

OPEN ACCESS

sensors

ISSN 1424-8220
www.mdpi.com/journal/sensors

Article

A Mobile Robots Experimental Environment with Event-Based Wireless Communication

María Guinaldo [1,*], Ernesto Fábregas [1], Gonzalo Farias [2], Sebastián Dormido-Canto [1], Dictino Chaos [1], José Sánchez [1] and Sebastián Dormido [1]

[1] Department of Computer Science and Automatic Control, UNED, Juan del Rosal 16, Madrid 28040, Spain; E-Mails: efabregas@bec.uned.es (E.F.); sebas@dia.uned.es (S.D-C.); dchaos@dia.uned.es (D.C.); jsanchez@dia.uned.es (J.S.); sdormido@dia.uned.es (S.D.)
[2] School of Electrical Engineering, Pontificia Universidad Católica de Valparaiso, Av. Brasil 2147, Valparaiso, Chile; E-Mail: gonzalo.farias@ucv.cl

* Author to whom correspondence should be addressed; E-Mail: mguinaldo@dia.uned.es; Tel.: +34-91-398-7147; Fax: +34-91-398-7690.

Received: 15 May 2013; in revised form: 12 July 2013 / Accepted: 18 July 2013 / Published: 22 July 2013

Abstract: An experimental platform to communicate between a set of mobile robots through a wireless network has been developed. The mobile robots get their position through a camera which performs as sensor. The video images are processed in a PC and a Waspmote card sends the corresponding position to each robot using the ZigBee standard. A distributed control algorithm based on event-triggered communications has been designed and implemented to bring the robots into the desired formation. Each robot communicates to its neighbors only at event times. Furthermore, a simulation tool has been developed to design and perform experiments with the system. An example of usage is presented.

Keywords: distributed control systems; mobile robots; multi-agent systems; wireless communications; embedded systems

1. Introduction

Significant developments in the fields of communication technology, wireless technology, embedded devices, and many others, have enabled the development of autonomous air, ground, or underwater vehicles. Groups of such vehicles, referred to as agents, can be utilized to solve a variety of problems very efficiently, such as for example exploration and monitoring tasks.

One of the main problems in these systems is when the control task is centralized. This centralization demands a very powerful communication network to send state information in a timely manner and detailed models of the mobile robot interactions. Both of these requirements can greatly limit the scalability of centralized approaches in networked control systems. An alternative way is to use a distributed approach [1,2], where each mobile robot decides its actions based only on his own state and the direct neighbors' information, while guaranteeing that the whole group fulfills a common group objective. One example of a group objective for multi-agent systems is state agreement or consensus, *i.e.*, all agents are supposed to converge to a common point or state. Such consensus problems have a variety of applications in flocking, attitude synchronization in satellite swarms, distributed sensor networks, congestion control in communication networks, or formation control [3]. We are particularly interested in the last field of application since achieving a stable formation is analogous to reaching consensus. In this consensus problem the agents communicate through a network exchanging information to reach the desired formation. Specifically, we propose to incorporate wireless links between the mobile robots to support the communications, the embedded systems, and the control.

Moreover, all real networks have bandwidth limitations that can cause delays in message delivery producing a major impact on overall system stability [4]. In this sense, it is desirable to know how frequently mobile robots should communicate such that the system maintains a desired level of performance. In this light, event-based communications are an alternative to time-scheduled control [5], and they have been in the focus of many researchers in recent years such as [6–11], to mention just a few. In the literature, several simplifications are usually done in both the robot dynamics and the network models of a multi-agent robot system. However, a real robot has its own physical parts, including sensors, processors, actuators, and communication devices. Therefore, its limitations must be considered, such as those related to computational power, sensor errors, and actuating and communicating devices.

This paper presents a multi-agent approach through the use of a decentralized control with event-triggered communication in a mobile robot formation control with a consensus algorithm. The decentralized control is less vulnerable to a node failure and easier to scale than the centralized approaches presented in [12–14]. In [12] the formation control of a multi-Khepera robot system is solved without communication between robots. However, each robot executes an algorithm periodically that requires the knowledge of the complete graph of the formation, so it is computationally intensive for large number of robots. Centralized approaches are proposed in [13,14]. In [13] all complexity of sensing, control, and computation is relegated to the leader of a group of robots, and [13] treats the robots formation as a single virtual rigid structure.

A decentralized approach for cooperative control is presented in [15] and a detailed hardware description is given and several experimental examples of multi-vehicle control for TXT-1 monster trucks from Tamiya, Inc, are given for both communication and vision formation control. In [16] a hardware platform for a single vehicle is developed and a PID event-based control is implemented. However, none of these works consider distributed event-based control in a multi-robot environment. Hence, the main contribution of this paper is the implementation of distributed event-triggered algorithms in a real mobile robots platform since, to the best knowledge of the authors, all the results in this field are mainly theoretical and only present simulation results. The experimental results show the goodness of the event-trigger approach compared to time-driven control, reducing the communication between agents. Furthermore, in contrast to other works, as for example [12,15], in which a costly platform for indoor or outdoor experimentation is used, we give a detailed description of our low-cost hardware platform, and how it was implemented in LEGO robots, in order to help readers who are considering building a similar system for research, educational purposes [17], or to mimic real world exploratory applications [18].

An interactive simulation, which is available at *http://lab.dia.uned.es/mass*, is also presented as a complement to the experimental setup. This simulation helped in the design and trial of different experiments with variable system and network conditions.

The paper is organized as follows. Section 2 describes in detail the experimental environment and its components. Section 3 gives a background on event-based control and multi-agent systems, and the extension of these to the formation control of a multi-robot system. Section 4 describes the simulation tool and an example of usage. Section 5 discusses experimental results and compares them to simulation results and experiences based on time-driven control. Finally, Section 6 ends the paper with conclusions and suggestions of future works.

2. Test Platform Description

We next describe the experimental environment designed with low-cost components. This may help readers to build similar systems. It can also be scaled up or down to equip bigger or smaller robots, respectively [19,20].

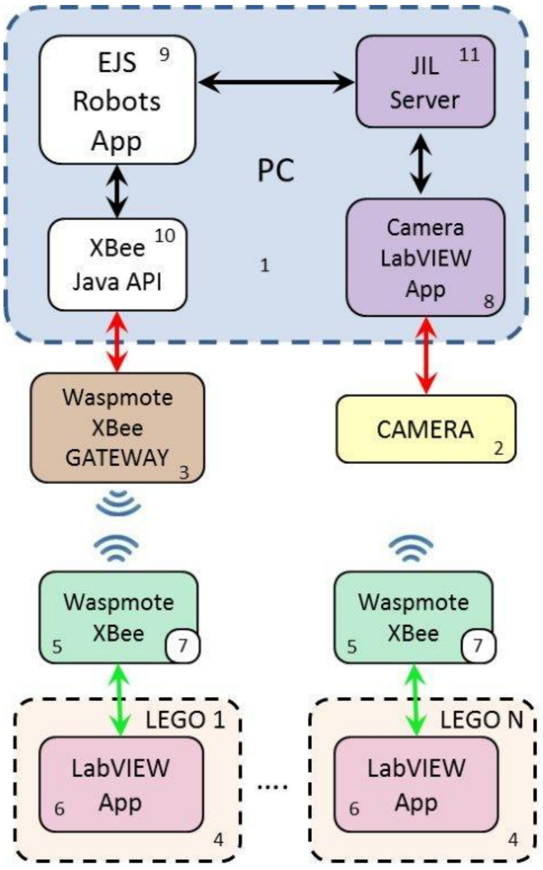
2.1. Platform Architecture and Components

The developed tool is a test platform for multi-mobile robot systems. It consists of two main components: the vision and the communication systems. Different software tools exchange information to transform the images captured by the camera into positioning data, which are sent to the robots via wireless communications. Figure 1 shows the block diagram of the architecture and the components of the platform, which are the following:

1. A personal computer (PC) to calculate and send the position and orientation to the robots.
2. A CCD camera (charge-coupled device) connected to the PC and equipped with wide angle optics. The camera is placed in the ceiling of the room, as described in [21].
3. A Waspmote Gateway card connected to the PC [22].
4. A group of LEGO Mindstorms NXT robots with the HiTechnic IR RC Car configuration [23,24].
5. A Waspmote card connected to each NXT LEGO Mindstorms robot by I^2C port [25].

6. A control and communication application, developed in LabVIEW, for each NXT LEGO robot.
7. An application, written in C++, for each Waspmote card.
8. An application (running on the PC), developed in LabVIEW, for the interaction with the camera [21].
9. A Java application, built with Easy Java Simulations (EJS) [26], for communication between the robots and the PC.
10. A Java XBee API to send and receive data through the Waspmote Gateway card [27].
11. The JIL server [28] to link LabVIEW with EJS.

Figure 1. Block diagram of the platform architecture.

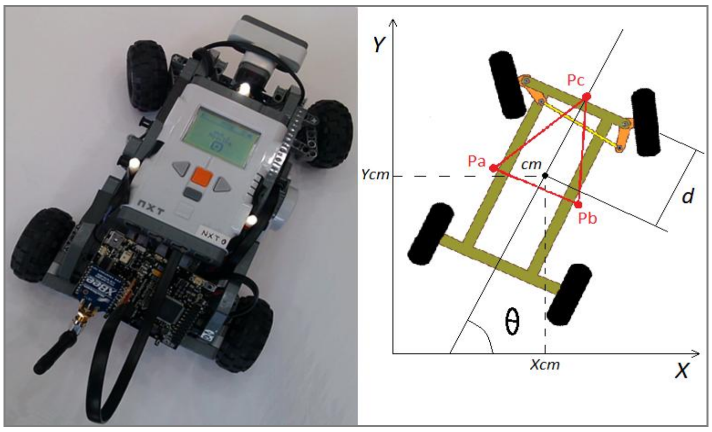


2.2. Vision System

Though LEGO platforms provide an interesting group of sensors such as ultrasonic, touch, or light sensors, there is no device for measuring absolute positions. This makes experimentation in robot formation control difficult. One option to solve the previous drawback is the use of a GPS as the main localization sensor as in [29]. However, when the experiments are performed indoors, GPS is impractical.

Another alternative is to incorporate a vision system to get the robots poses. The recent appearance of RGB-D cameras has made possible 3D mapping from a single image and state estimation. However, running the robot real-time state estimation algorithms on board would require a powerful computer, which would increase the cost of the platform considerably. Thus, a single camera on the ceiling observes the arena from an overhead position. Each robot can be distinguished thanks to three high brightness LEDs. These light spots form an isosceles triangle with different aspect ratio for each robot as shown in Figure 2a. In Figure 2b, the points *Pa*, *Pb*, and *Pc* represent the LEDs. The orientation is denoted by θ. The position of each robot is calculated using their center of mass (*Xcm; Ycm*).

Figure 2. (**a**) The NXT LEGO robot, (**b**) Isosceles triangle formed by the LEDs.



(**a**) (**b**)

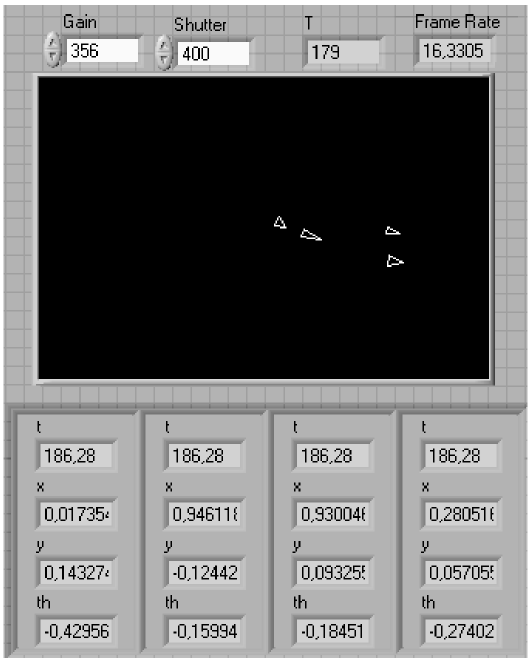
The Basler A631fc CCD camera (block 2 of Figure 1) is configured in monochrome mode for image acquisition in order to maximize the resolution and shutter speed. Under these conditions, we obtain images with a resolution of 1,388 × 1,038 pixels at 18.7 fps.

The images captured by the camera are processed by the LabVIEW Camera App (block 8 of Figure 1), which computes the robots poses. The execution of the application implies the following steps:

- *Barrel distortion correction:* The image magnification decreases with distance from the optical axis. The apparent effect means that an image has been mapped around a sphere (or barrel).
- *Perspective transformation:* Transforms the coordinates of the object to the camera coordinates, making rotation and translation transformations.
- *Calculation of positions and orientations:* From the reference points on the ground, we calculate the position and orientation taking into account the morphology of the obtained points.
- *Transformation of coordinates:* From pixels to meters.
- *Calibration:* Gets the perspective calibration parameters.

The front panel of the LabVIEW Camera App is shown in Figure 3. The values of the configuration of the camera (gain, shutter, and frame rate) and the period (T) are required to compute the bearings, and shown on the front panel. The application sends the absolute position and orientation to the robots every 200 milliseconds. Further details about this application can be found in [21].

Figure 3. User interface of the Camera LabVIEW App.



```
Gain      Shutter   T         Frame Rate
356       400       179       16,3305

t         t         t         t
186,28    186,28    186,28    186,28
x         x         x         x
0,01735   0,94611   0,93004   0,28051
y         y         y         y
0,14327   -0,12442  0,09325   0,05705
th        th        th        th
-0,42956  -0,15994  -0,18451  -0,27402
```

2.3. Communication System

After the processing of the robots poses, they are sent to the robots. Each one receives its position and orientation, and decides when to broadcast this information to its neighbors based on the local data, updating the control law afterwards. Thus, the control is distributed in the sense that each robot takes by itself the decision of when to transmit its state and the control law is computed locally, in contrast to [16], where the control law is computed on the central computer and sent to the robots.

Waspmote boards (block 5 of Figure 1) are used for the wireless communications. This card has a modular architecture and is based on the Atmel ATMEGA 1281 microcontroller. The Waspmote also integrates the Digi XBee RF (ZigBee standard) module for communication in the ISMB bands (Industrial Scientific Medical Band, RF-868 MHz) [30]. This module communicates with the microcontroller via interruptions using the UART at the speed of 38,400 bps. It allows the definition of different network configurations (e.g. p2p, tree, or mesh) and uses the protocol 802.15.4 with a frequency of 2.4 GHz [22]. For wireless network settings, each card uses the MAC address of its corresponding XBee. Each robot "talks" to its neighbors given by the communication graph (see Sections 3 and 4.2 for further details).

The power supply for the Waspmote is a lithium battery. The energy system has four operational modes (on, sleep, deep sleep, and hibernate). In normal operating mode (on), the consumption is 9 mA. Every 125 ns the microcontroller runs a routine of low level (machine language) instructions [31,32], which is fast enough to the time requirements of our system.

The I^2C bus interfaces the NXT LEGO with the Waspmote. The NXT LEGO runs as master and the Waspmote as slave. The Waspmote card runs an application (block 7 Figure 1) that allows the LEGO NXT to communicate with other robots and the PC. Every 100 ms the following tasks are executed:

1. It checks if a packet has been received by the XBee radio. Packets can be sent from the PC or from another robot. If the answer is positive, the received data is stored in a buffer until the NXT LEGO requires it.
2. It checks if a flag is true. If the answer is positive, it indicates that the NXT LEGO requires the sending of a packet to other robots (its neighbors) by Xbee radio. Once the packet is sent the flag is set to false. The NXT LEGO triggers an interruption routine (which sets the flag to true) when a packet must be sent. In event-based control, the flag is set to true at the event occurrence time.

Each NXT LEGO runs a LabVIEW App (block 6 of Figure 1) to interface with the Waspmote card by I^2C (see Section 2.4 for further details). This application requests data to the Waspmote every 200 ms. If the received packets are from the camera application (own position and orientation), the position is compared to the last broadcasted position, *i.e.*, to the state which generated the last event. If the difference is greater than a threshold, then the current position is sent to its neighbors by using the I^2C interface with the Waspmote. By contrast, if the source is any robot in the neighborhood, the received data is then used to update the control law as explained later in Section 3.

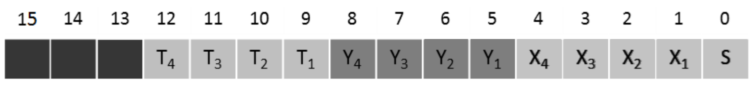
The communication between the PC and the Waspmote is carried out by the Waspmote's Gateway (block 3 of Figure 1) connected to the PC through the USB port. The Waspmote's Gateway has drivers for Java but not for LabVIEW. For this reason, it is necessary to use an application developed in Java to send the data to each corresponding robot using the gateway. In this case, the JIL Server application [28] (block 11 of Figure 1) is used as interface between the camera application and the EJS Robots App (block 9 of Figure 1). This latter application sends the position and orientation to the robots by using XBee-Java-API (block 10 of Figure 1).

The data packet sent from the PC and between the robots has a size of 16 bytes. This is the maximum size allowed by NXT LEGO Mindstorms for the communications by I^2C port [33].

Figure 4 shows the packet structure, which is as follows:

- Byte 0 (S): The source where the packet comes from. The value is 0 for the robot 0, 1 for robot 1, and so on.
- Bytes 1…4 ($X_1 \ldots X_4$): 4 bytes corresponding to the X coordinate of the robot position.
- Bytes 5…8 ($Y_1 \ldots Y_4$): 4 bytes corresponding to the Y coordinate of the robot position.
- Bytes 9…12 ($T_1 \ldots T_4$): 4 bytes corresponding to the robot orientation (θ).
- Bytes 13…15: Not used in this platform.

Figure 4. Data packet structure.



2.4. Using I^2C in NXT LEGO

The LabVIEW toolkit for LEGO has buffers to write and read in each port, and three system call methods to access to these buffers. The write operation begins with the corresponding system call method, which constitutes the start of an asynchronous transaction between the NXT brick and the

Waspmote. If the return value is zero, the method starts a communication transaction. After a write transaction is started it is necessary to use a system call function to check the status of the port. To ensure the success of the transmission, there is a system call function to check the status port.

The read operation of a device through I^2C port is carried out in two stages. First, a write operation must be executed with the number of bytes that the slave has to send. If the count of bytes available in the read buffer is not zero, the read operation is executed and the data is obtained in the external buffer.

3. Event-Based Control and Communication in Multi-Agent Systems

3.1. Multi-Agent Systems and the Consensus Problem

The simplest model to represent the communication topology of a multi-agent system is a graph $G = \{V,E\}$, where the nodes V correspond to agents and the edges E represent communication links between nodes.

According to [3], a simple consensus algorithm to reach an agreement regarding the state of N single integrators of the form $\dot{x}_i(t) = u_i(t)$ can be expressed as an nth order linear system on a graph:

$$\dot{x}_i(t) = \sum_{j \in N_i} \left(x_j(t) - x_i(t) \right) \tag{1}$$

The dynamics of the group of agents can be written as:

$$\dot{x}(t) = -Lx(t) \tag{2}$$

where L is the laplacian of the graph, and N_i the set of neighbors of the node i [34].

Based on analytical tools from algebraic graph theory, it can be shown that if the graph is connected, then there is an unique equilibrium state for Equation (2) of the form $x_{eq} = \alpha \mathbf{1}$, where $\alpha = \frac{1}{N}\sum_{i=1}^{N} x_i(0)$ and $\mathbf{1} = (1 \ldots 1)^T$.

Consensus algorithms can be extended to formation control if the formation is represented by vectors of relative positions of neighboring agents. In particular, let us denote by $\mathbf{r}_{ij}$ the desired inter-vehicle relative position vector. The following control law:

$$\dot{x}_i(t) = \sum_{j \in N_i} \left(x_j(t) - x_i(t) - \mathbf{r}_{ij} \right) \tag{3}$$

yields the group to achieve the objective of the formation.

The consensus problem has been extending regarding the agents dynamics to higher order integrators [35,36], or linear systems [37].

3.2. Model of Non-Holonomic Mobile Robots

Single or double integrators do not describe properly the dynamics of most of commercial mobile robots, since these cannot move in any direction instantaneously. In robotics, holonomicity refers to the relationship between the controllable and total degrees of freedom of a given robot. If the controllable degrees of freedom are less than the total degrees of freedom the vehicle is non-holonomic. To avoid the non-holonomic constraint, Reference [38] defines the dynamics in terms of the front wheels coordinates $(\bar{x}_i, \bar{y}_i)$ as:

$$\begin{pmatrix} \dot{\bar{x}}_i \\ \dot{\bar{y}}_i \\ \dot{\theta}_i \end{pmatrix} = \begin{pmatrix} \cos\theta_i & -d\sin\theta_i \\ \sin\theta_i & d\cos\theta_i \\ 0 & 1 \end{pmatrix} \begin{pmatrix} v_i \\ \omega_i \end{pmatrix} \quad (4)$$

where $\bar{x}_i = x_i + d\cos\theta_i$, $\bar{y}_i = y_i + d\sin\theta_i$. The triple ($x_i$, y_i, θ_i) denotes the position and the orientation of the vehicle (see Figure 2b, $x_i = Xcm$, $y_i = Ycm$ for each vehicle), and v_i and ω_i are the longitudinal and angular velocities, respectively. Let us consider as control inputs for each robot $u_i = (v_i, \omega_i)$.

3.3. Time Schedule Control

In a distributed control approach, each agent collects information from its neighboring nodes and updates the control law according to some rules. In a networked system, collecting information means to transmit it through the communication channel. The information can be transmitted through the network in a periodic or an event-based fashion. The first case involves the traditional approach of sampling at pre-specified time instances, usually separated by a specific sampling period. However, the selected period must assure adequate system performance over a wide range of operating conditions. This can yield to a conservative choice and result in significant over-provisioning of the communication-network. Event-triggering represents one way of generating sporadic transmissions across network channels [39]. The communication is invoked only when something significant has occurred in the system.

The problem of multi-agent systems with event-based communications has been recently addressed [40,41]. Whereas the controller design of [41] is centralized, [40] presents a distributed approach, in which the agents are modeled by single and double integrators. The control law is based on the states that the agents broadcast to their neighbors through the network at event times. The average consensus is reached preserving convergence properties and the experimental results show a significant reduction in the number of transmissions respect to a periodic approach. Distributed event-based control has been also addressed in [42], where physical coupling may exist between the subsystems.

The control signals in each robot are computed based on the broadcasted information, which means that are not updated continuously, not even at each measurement of its own state received by the camera (see Section 2.2). The rule to determine when to broadcast is given by the *trigger function*. A trigger function determines the instants of time $\{t_k^i\}$ at which each agent broadcasts the state to its neighbors. In general, these broadcasting times are not equidistant as in periodic communications. Moreover, the proposed trigger functions depend on local information only.

If we denote the state of an agent i as $x_i(t)$ and $\hat{x}_i(t)$ as the broadcasted state of agent i to its neighbors, which is a piecewise constant function, an event is triggered for an agent i when:

$$f_i\left(x_i(t), \hat{x}_i(t), \bigcup_{j \in N_i} \hat{x}_j(t) \right) > 0 \quad (5)$$

is fulfilled, where N_i is the neighborhood of agent i. The trigger functions are said to depend on local information because the decision of when to re-compute the control law depends on the last update values and the neighbors' states, and each agent does not have to monitor the states of its neighbors continuously in order to evaluate the triggering condition, in contrast to the trigger rules proposed in [10].

Specific trigger functions based on the error $e_i(t) = \hat{x}_i(t) - x_i(t)$ are proposed. For example, static trigger functions have the form $f_i(e_i(t)) = |e_i(t)| - c_0$, where c_0 is a constant value.

The control law $u_i = (v_i, \omega_i)$ to reach the desired formation is computed as:

$$\begin{pmatrix} v_i \\ \omega_i \end{pmatrix} = -\begin{pmatrix} \cos\theta & -d\sin\theta_i \\ \sin\theta_i & d\cos\theta_i \end{pmatrix}^{-1} \begin{pmatrix} L & \mathbf{0} \\ \mathbf{0} & L \end{pmatrix} \begin{pmatrix} \hat{\hat{x}} - \bar{x}_r \\ \hat{\hat{y}} - \bar{y}_r \end{pmatrix} \tag{6}$$

where:

- $\hat{\hat{x}}$ and $\hat{\hat{y}}$ are the stack vectors of the broadcasted positions, *i.e.* $\hat{\hat{x}}^T = \left(\hat{\hat{x}}_1, ..., \hat{\hat{x}}_N\right)$ and $\hat{\hat{y}}^T = \left(\hat{\hat{y}}_1, ..., \hat{\hat{y}}_N\right)$.
- $\bar{x}_r$ and $\bar{y}_r$ are the stack vectors of the desired positions of the agents with reference to the center of the group.

Combining Equations (4) and (6) it yields that:

$$\begin{aligned} \dot{\bar{x}} &= -L\left(\hat{\hat{x}} - \bar{x}_r\right) \\ \dot{\bar{y}} &= -L\left(\hat{\hat{y}} - \bar{y}_r\right) \end{aligned} \tag{7}$$

3.4. Stability Anaalysis

Note that defining the control law given in Equation (6), two decoupled equations are obtained and they correspond to the equation of the event-triggered consensus problem of single integrators studied in [40], but extended to reach a formation. There it was proved that the consensus was reached, so in this case the agents achieve the desired formation translated into the XY plane by a vector which is the average of the initial positions (average consensus).

Hence, the stability results sum up in the following corollary to the Theorem 4 in [40], in which time-dependent trigger functions of the form:

$$f_i(e_i(t)) = |e_i(t)| - (c_0 + c_1 e^{-\alpha t}) \tag{8}$$

are considered.

Remark 1. In trigger functions given in Equation (8) it is assumed that the initial conditions are t_0 = 0. Otherwise, Equation (8) can be rewritten as:

$$f_i(e_i(t)) = |e_i(t)| - (c_0 + c_1 e^{-\alpha(t-t_0)}) \tag{9}$$

where t_0 refers to the initial time of the experiment. Thus, the threshold decreases exponentially and takes the value c_0 in the limit when time goes to infinity.

Corollary 2. If trigger functions given in Equation (8) are defined for the errors in the x- and y-coordinates, such that:

$$\begin{aligned} e_{x,i} &= \hat{\hat{x}}_i - \bar{x}_i \\ e_{y,i} &= \hat{\hat{y}}_i - \bar{y}_i, \end{aligned} \tag{10}$$

the formation given by $\left(\bar{x}_r, \bar{y}_r\right)$ is reached with an error that depends on the value of parameter c_0 of the trigger function, and the center of the formation is the average of the initial positions:

$$a_x = \frac{1}{N}\sum_{i=1}^{N}\bar{x}_i(0)$$
$$a_y = \frac{1}{N}\sum_{i=1}^{N}\bar{y}_i(0). \tag{11}$$

Proof. Let us denote:

$$\delta_x = \bar{x} - a_x\mathbf{1} - \bar{x}_r$$
$$\delta_y = \bar{y} - a_y\mathbf{1} - \bar{y}_r \tag{12}$$

the disagreement vectors in the *x*- and *y*-coordinates, respectively, where 1 = (1 … 1)^T is an eigenvector of the laplacian *L* [3]. Thus, it holds that $\dot{\delta}_x = \dot{\bar{x}}, \dot{\delta}_y = \dot{\bar{y}}$, and it yields:

$$\dot{\delta}_x = -L(\delta_x + e_x)$$
$$\dot{\delta}_y = -L(\delta_y + e_y), \tag{13}$$

where $e_x = (e_{x,1} \ldots e_{x,N})^T$ and $e_y = (e_{y,1} \ldots e_{y,N})^T$. Hence, applying the results of Theorem 4 in [40], it yields the disagreement vectors δ_x, δ_y converge to the following regions, respectively:

$$\|\delta_x\| \le \frac{L}{\lambda_2(G)}\sqrt{N}c_0$$
$$\|\delta_y\| \le \frac{L}{\lambda_2(G)}\sqrt{N}c_0, \tag{14}$$

where $\lambda_2(G)$ is the algebraic connectivity of the graph *G* (second smallest eigenvalue of *L*).

Remark 2. Note that the convergence of the disagreement vectors δ_x, δ_y proves that the group of robots reaches the formation at $(a_x\mathbf{1} + \bar{x}_r, a_y\mathbf{1} + \bar{y}_r)$ with an error given by Equation (14).

Despite the fact that the existing theory that proves the robustness of the proposed control strategy and the simulation results that show a significant improvement as far as the number of transmissions between agents is concerned, there are neither experimental results on a real platform nor a simulation tool to perform a wide set of experiments of formation control of mobile robots. The following section describes the developed software and all its capabilities.

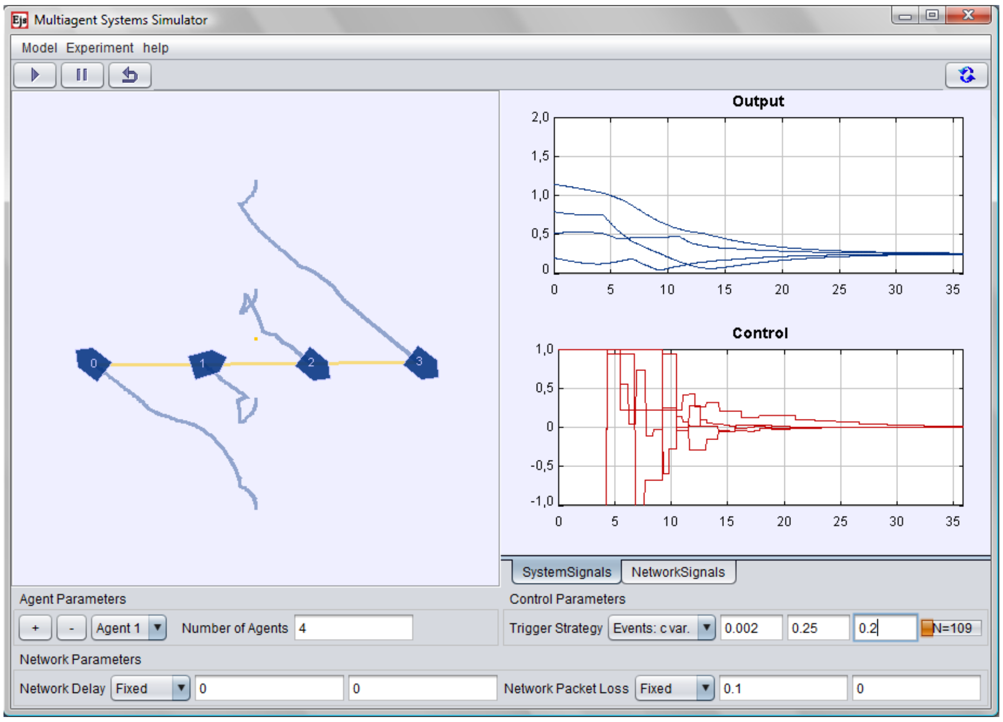
4. Simulation Tool

The many control and system configuration options needed to simulate a multi-agent system demand a graphical user interface (GUI) with a high degree of flexibility. The GUI is intended to make rapid prototyping and simulation of wireless autonomous agents which execute distributed control algorithms and perform event-based communications. Nevertheless, the GUI has been made keeping the interaction with the user relatively simple and intuitive in order to be used as a pedagogical tool for advanced engineering control courses. The simulation tool is available on line at http://lab.dia.uned.es/mass. See [43] for further details.

4.1. Description of the Simulator

The interface of the application, shown in Figure 5, has five main panels, a menu bar, and a small task bar. The two upper panels of the interface provide a quick view of the multi-agent system and a time plot of the outputs and control signals. The top left panel shows an animation of the complete multi-agent system. Each agent is numbered and shows a trace of its former positions. The lower left panel, named *Agent Parameters*, allows users to set the number of agents in the system, as well as to add and remove a particular agent.

Figure 5. Illustrative example of the simulation tool. A multi-agent system with four agents is simulated. Note that the control actions are constant between two communication events.



Agent Parameters
+ - Agent 1 Number of Agents 4
Control Parameters
Trigger Strategy Events: c var. 0.002 0.25 0.2 N=109
Network Parameters
Network Delay Fixed 0 0
Network Packet Loss Fixed 0.1 0

The time plots on the top right panel, which are grouped in the *System Signals* tab, display the relative distance to the desired formation as well as the control actions of each agent. There are also plots grouped in the *Network Signals* tab, which provide information about the measured delay of the sent packets.

The lower panel, named *Network Parameters*, is devoted to setting up the network delay and also the network packet loss. Users can choose this panel to set fixed or random delay and loss probability functions. The lower right panel *Control Parameters* is used to specify the communication strategy which triggers the sending of packets from the agents in order to update the control actions. The interface is completed with a top task bar that provides buttons to start, pause, and reset the simulation.

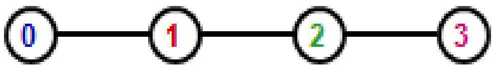
The components of the interface described earlier provide the basic functionality required to operate the application. A menu bar provides some additional features such as the possibility to specify the dynamic model of the agents and to select a predefined multi-agent system configuration and the experiment to be performed. The user can also save a configuration, mark it as default or load a previous configuration.

4.2. A Practical Example

This section presents a usage example of the simulator described above. Different experiments are prefixed. We select one of them in the top menu, for example, *Experiment 2*, in which the communication topology for four vehicles is given by the graph in Figure 6, and the desired formation respect to the center of the group is:

$$x_r = [-0.7\ \ -0.25\ \ 0.2\ \ 0.65], \quad y_r = [0\ \ 0\ \ 0\ \ 0] \tag{11}$$

Figure 6. Communication graph describing one-dimensional network topology.



The user can find in the help menu all the information about these experiments. Random initial conditions are generated, but we may be interested in comparing the performance of different control strategies (periodic and event-based, for example) for a prefixed initial coordinates. We can do that by right-clicking in the drawing panel and selecting the option "*Setting current state as default*".

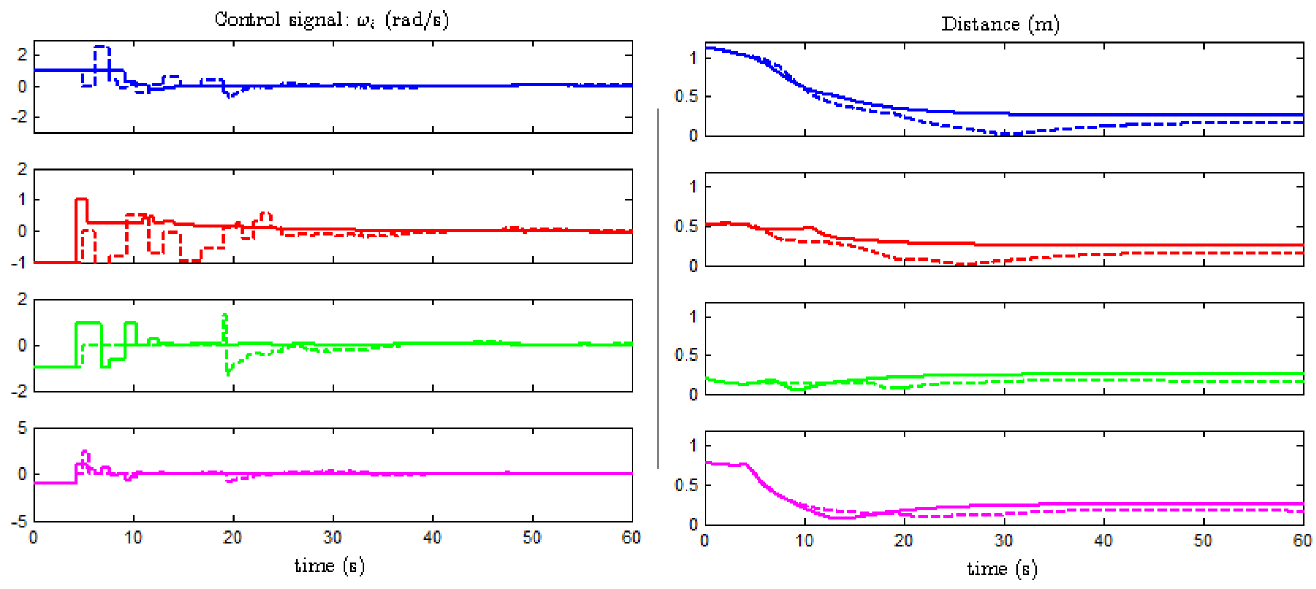
Figure 5 shows the simulation output when the trigger function is of the form $f_i(e_i(t)) = |e_i(t)| - 0.002 + 0.25e^{-0.2t}$. The user can change the parameters of f_i while the simulation is running or switch to a periodic communication and check online the effect of these changes in the system.

5. Experimental Results

5.1. Comparison of Simulation and Experimental Results

Figure 7a,b shows the second component of the control signals (ω_i) and the distances to the formation $d_i = \sqrt{(\bar{x}_i - \bar{x}_{i,r})^2 + (\bar{y}_i - \bar{y}_{i,r})^2}$, respectively, for the four agents when the experiment described in Section 4.2 is performed over the model (solid lines) and the real system (dashed lines).

Figure 7. Comparative of simulation and experimental results: (**a**) Control signal, (**b**) Distance to formation.

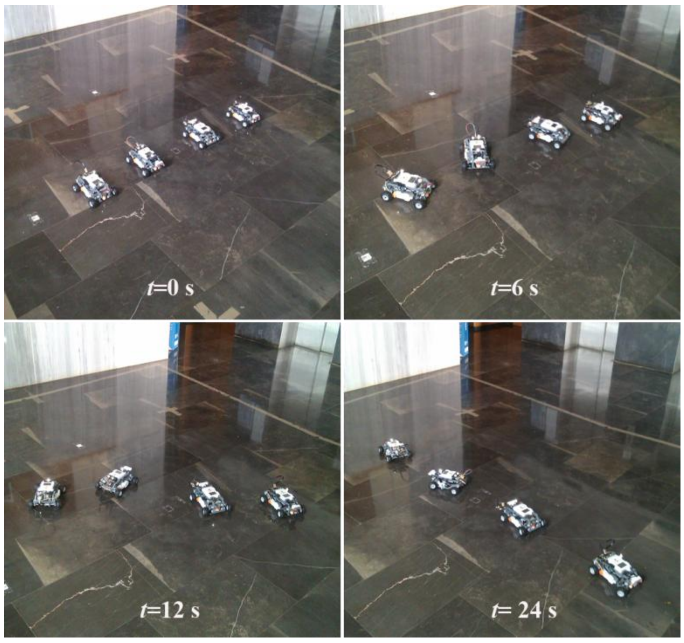


(**a**) (**b**)

We observe that the major discrepancies are at the beginning of the experiment. Note that the control signals are constant piecewise functions and change their value only at event times, while Equation (6) reads that they change continuously with the orientation θ_i even though $\hat{\bar{x}} - \bar{x}_r$ and $\hat{\bar{y}} - \bar{y}_r$ remain constant in the inter-event times. The reason for this is the constraint imposed by the hardware in the real system. Because one agent cannot monitor its state continuously (the camera sends measurements every 200 ms, which is the minimum period allowed by the Waspmote cards), and the slow variation of the angle, the computation required to implement the continuous control law given in Equation (6) is not worthy. Thus, the angle is considered to remain constant between events.

Moreover, the selected trigger functions decrease with time. This means that the threshold to detect an event is greater for small values of time, and fewer events are generated consequently. When the system approaches to the desired formation, the threshold decreases and makes the error arbitrary small. Note that d_i takes the same value for all i = 1,…,4 and the final state only depends on the initial conditions, as predicted by the consensus problem theory (see Section 3). However, the final state for the real experiment differs slightly from the result given by the model. The reason for that might be sensing errors, dead-zone in the actuator and/or transmission delays. Finally, Figure 8 shows the view of the experiment at t = 0, 6, 12, and 24 s.

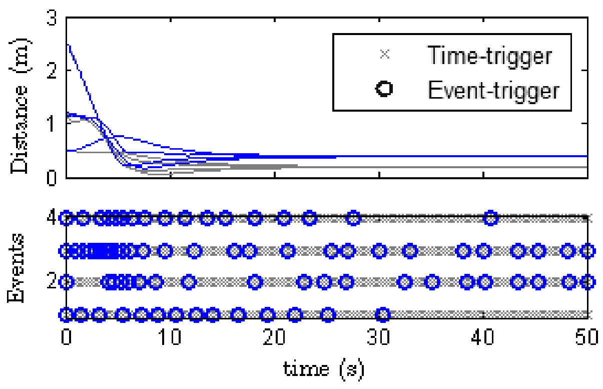
Figure 8. View of the real experiment at t = 0, 6, 12, and 24 s.



5.2. *Event-Trigger* Versus *Time-Trigger*

Figure 9 compares the results obtained from event-trigger and time-trigger experiments. In the last case, each agent receives data from the camera every 200 ms, it processes the information, broadcasts its state to its neighbors, and computes the control law every two samples (400 ms). In the lower graph of Figure 9 the broadcasts in both cases are depicted for each agent (labeled from 1 to 4). For the periodic case, the number of packets sent through the network during the experiment should be $50s \times 4 agents / 0.4 = 500$. However, for the event-based control this number is by far lower: 83 events in total.

Figure 9. Comparison of the results obtained for event-trigger and time-trigger experiments.



In the upper graph of Figure 9 the distance to the formation is shown. Note that even though the desired formation and the initial conditions are the same, the final state (which depends on the average of the initial positions) is different. As remarked before, these differences come out because it is difficult in experimentation to exactly reproduce the same conditions.

For instance, network delays cannot be controlled, or sensing errors may occur more frequently in one case than in the other. However, observe that the time at which the formation is achieved (the stationary) is very similar for both time-trigger and event-trigger.

6. Conclusions

We have described an experimental environment for mobile robots. The hardware platform was developed using different components and software to overcome the limitations of the LEGO Mindstorms NXT robots. A distributed control approach has been used in formation control. Each robot communicates to its neighborhood in an event-triggering fashion for network saving purposes. A simulation tool with high degree of flexibility has been presented. An example has demonstrated its usability, and simulated and experimental results have been presented showing how the multi-agent system reaches the desired formation with few communications meanwhile the system performance is preserved. Future works will include the implementation of collision and obstacle avoidance control and formation control with specified paths. Exploring self-triggering techniques [44] to predict the next event occurrence in order to optimize the communication between the camera and the robots is also a future line of research.

Acknowledgments

This work was supported in part by the Spanish Ministry of Economy and Competitiveness under Projects DPI2012-31303 and DPI2011-27818-C02-02.

Conflict of Interest

The authors declare no conflict of interest.

References

1. Franchi, L.; Freda, G.O.; Vendittelli, M. The sensor-based random graph method for cooperative robot exploration. *IEEE ASME Trans. Mechatron.* **2009**, *14*, 163–175.
2. Magnenat, S.; Rétornaz, P.; Bonani, M.; Longchamp, V.; Mondada. F. ASEBA: A modular architecture for event-based control of complex robots. *IEEE ASME Trans. Mechatron.* **2011**, *6*, 321–329.
3. Olfati-Saber, R.; Fax, J.; Murray, R. Consensus and cooperation in networked multi-agent systems. *IEEE Proc.* **2007**, *95*, 215–233.
4. Lian, F.-L.; Moyne, J.; Tilbury, D. Network design consideration for distributed control systems. *IEEE Trans. Control Syst. Technol.* **2002**, *10*, 297–307.
5. Aström, K.; Bernhardsson, B. Comparison of Riemann and Lebesgue Sampling for First Order Stochastic Systems. In Proceedings of IEEE Conference on Decision and Control, Las Vegas, NA, USA, 10–13 December 2002; pp. 2011–2016.
6. Tabuada, P. Event-triggered real-time scheduling of stabilizing control tasks. *IEEE Trans. Autom. Control* **2007**, *52*, 1680–1685.
7. Rabi, M.; Johansson, K.H.; Johansson, M. Optimal Stopping for Event-Triggered Sensing and Actuation. In Proceedings of 47th IEEE Conference on Decision and Control, Cancun, Mexico, 9–11 December 2008; pp. 3607–3612.
8. Dimarogonas, D.V.; Frazzoli, E.; Johansson, K.H. Distributed event-triggered control for multi-agent systems. *IEEE Trans. Autom. Control* **2012**, *57*, 1291–1297.
9. Mazo, M.; Tabuada, P. Decentralized event-triggered control over wireless sensor/actuator networks. *IEEE Trans. Autom. Control* **2011**, *56*, 2456–2461.
10. Wang, X.; Lemmon, M.D. Event-triggering in distributed networked control systems. *IEEE Trans. Autom. Control* **2011**, *56*, 586–601.
11. Marchand, N.; Durand, S.; Guerrero-Castellanos, J.F. A general formula for event-based stabilization of nonlinear systems. *IEEE Trans. Autom. Control* **2013**, *58*, 1332–1337.
12. Macdonald, E.A. Multi-Robot Assignment and Formation Control. M.Sc. Thesis, Georgia Institute of Technology, Atlanta, GA, USA, 7 June 2011.
13. Huang, J.; Farritor, S.M.; Qadi, A.; Goddard, S. Localization and follow-the-leader control of a heterogeneous group of mobile robots. *IEEE ASME Trans. Mechatron.* **2006**, *11*, 205–215.
14. Mehrjerdi, H.; Saad, M.; Ghommam, M.J. Hierarchical fuzzy cooperative control and path following for a team of mobile robots. *IEEE ASME Trans. Mechatron.* **2011**, *16*, 907–917.
15. Cruz, D.; Mcclintock, J.; Perteet, B.; Orqueda, O.; Cao, Y.; Fierro, R. A multivehicle platform for research in networked embedded systems. *IEEE Control Syst.* **2007**, *27*, 58–78.
16. Durand, S.; Minet, J.; Guerrero, J.F.; Marchand, N. ASYNCAR, a Radio-Controlled Vehicle for Asynchronous Experiments Implementation of an Event-Based Cruise Control. In Proceedings of 8th International Conference on Electrical Engineering, Computing Science and Automatic Control, Merida City, Mexico, 26–28 October 2011; pp. 1–6.
17. Casani, M.; Garulli, A.; Giannitrapani, A.; Vicino, A. A LEGO Mindstorms Multi-Robot Setup in the Automatic Control Telelab. In Proceedings of 18th International Federation of Automatic Control (IFAC) Word Congress, Milano, Italy, 28 August–2 September 2011; pp. 9812–9817.

18. Maze, M.; Wan, Y.; Namuduri, K.; Varanasi, M. A LEGO Mindstorms NXT-Based Test Bench for Cohesive Distributed Multi-Agent Exploratory Systems; Mobility and Coordination Proc.: Garden Grove, CA, USA, 19–21 June 2012; pp. 2000–2013.
19. Fu, G.; Corradi, P.; Menciassi, A.; Dario, P. An integrated triangulation laser scanner for obstacle detection of miniature mobile robots in indoor environment. *IEEE ASME Trans. Mechatron.* **2011**, *16*, 778–783.
20. Espinosa, F.; Santos, C.; Marrón-Romera, M.; Pizarro, D.; Valdés, F.; Dongil, J. Odometry and laser scanner fusion based on a discrete extended kalman filter for robotic platooning guidance. *Sensors* **2011**, *11*, 8339–8357.
21. Chaos, D. Control no lineal de vehículos subactuados marinos no-holonómicos. Ph.D. Thesis, Universidad Nacional de Educación a Distancia (UNED), Madrid, Spain, 10 December 2011.
22. Libelium Comunicaciones Distribuidas S.L., Waspmote Datasheet. Available online: http://www. libelium.com/downloads/documentation/waspmote_datasheet.pdf (accessed on 18 June 2013).
23. LEGO Group. LEGO Mindstorms NXT User Guide. Available online: http://mindstorms.lego.com/ en-us/support/buildinginstructions/8547/8547%20User%20Guide%20English.aspx (accessed on 18 June 2013).
24. HiTechnic IR RC Kart Building Instructions. Available online: http://www.hitechnic.com/ file.php?f=503-HTRCKart.pdf (accessed on 18 June 2013).
25. Robot Electronics, Using the I2C Bus. Available online: http://www.robot-electronics.co.uk/ acatalog/I2C_Tutorial.html (accessed on 4 April 2013).
26. Esquembre, F. Easy Java Simulations: A software tool to create scientific simulations in Java. *Comput. Phys. Commun.* **2004**, *156*, 199–204.
27. Raap, XBee-API A Java API for Digi XBee/XBee-Pro OEM RF Modules. Available online: http://code.google.com/p/xbee-api/ (accessed on 6 April 2013).
28. Vargas, H.; Sánchez, J.; Salzmann, C.; Esquembre, F.; Gillet, D.; Dormido, S. Web-enabled remote scientific environments. *Comput. Sci. Eng.* **2009**, *11*, 34–46.
29. Viguria, A.; Howard, M. An Integrated approach for achieving multirobot task formations. *IEEE ASME Trans. Mechatron.* **2009**, *14*, 176–186.
30. Digi International, Inc. Product Manual v1.xEx—802.15.4 Protocol. Available online: http://ftp1.digi.com/support/documentation/90000982_K.pdf (accessed on 7 April 2013).
31. Libelium Comunicaciones Distribuidas S.L. Waspmote Technical Guide. Available online: http://www.libelium.com/documentation/waspmote/waspmote-technical_guide_eng.pdf (accessed on 7 April 2013).
32. Guascon, D. Long Range Multiprotocol Wireless Sensor Networks. In Proceedings of Best Demo Award at the Sensing Technology 2010, Birmingham, UK, 2010.
33. National Instruments, LabVIEW Toolkits for LEGO Mindstorms NXT Programming Guide. Available online: http://download.ni.com/evaluation/mindstorms/LabVIEW_for_NXT_Advanced_ Programming_Guide.pdf (accessed on 18 June 2013).
34. Godsil, G.R. *Algebraic Graph Theory*; Springer: New York, NY, USA, 2001.
35. Ren, W.; Atkins, E. Distributed multi-vehicle coordinated control via local information exchange. *Int. J. Robust Nonlinear Control* **2007**, *17*, 1002–1033.

36. Ren, W.; Moore, K.; Chen, Y. High-Order Consensus Algorithms in Cooperative Vehicle Systems. In Proceedings of International Conference on Networking, Sensing and Control, Ft. Lauderdale, FL, USA, 23–25 April 2006; pp. 457–462.
37. Seo, J.H.; Shim, H.; Back, J. Consensus of high-order linear systems using dynamic output feedback compensator: Low gain approach. *Automatica* **2009**, *45*, 2659–2664.
38. Lawton, J.R.T.; Beard, R.W.; Young, B.J. A decentralized approach to formation maneuvers. *IEEE Trans. Robot. Autom.* **2003**, *19*, 933–941.
39. Lemmon, M. Event-Triggered Feedback in Control, Estimation, and Optimization. In *Networked Control Systems*; Springer Verlag: Heidelberg, Germany, 2010.
40. Seyboth, G.; Dimarogonas, D.; Johansson, K. Control of Multi-Agent Systems via Event-Based Communication. In Proceedings of 18th International Federation of Automatic Control (IFAC) World Congress, Milano, Italy, 28 August–2 September 2011; pp. 10086–10091.
41. Demir, O.; Lunze, J. Cooperative Control of Multi-Agent Systems with Event-Based Communication. In Proceedings of American Control Conference, Montreal, Canada, 27–29 June 2012; pp. 4504–4509.
42. Guinaldo, M.; Dimarogonas, D.; Johansson, K.; Sanchez, J.; Dormido, S. Distributed Event-Based Control for Interconnected Linear Systems. In Proceedings of 50th IEEE Conference on Decision and Control, Orlando, FL, USA, 12–15 December 2011; pp. 2553–2558.
43. Guinaldo, M.; Farias, G.; Fabregas, E.; Sánchez, J.; Dormido-Canto, S.; Dormido, S. An interactive simulator for networked mobile robots. *IEEE Netw. Mag.* **2012**, *26*, 14–20.
44. Mazo, M.; Anta, A.; Tabuada, P. An ISS self-triggered implementation for linear controllers. *Automatica* **2010**, *46*, 1310–1314.