Proceedings of the 2004 IEEE
International Conference on Robotics & Automation
New Orleans, LA • April 2004

Bandwidth Reduction for Controller Area Networks Using Adaptive Sampling

Ian A. Gravagne*, John M. Davis†, Jeffrey J. Dacunha†, Robert J. Marks*

Abstract

This paper presents a method by which controllers operating on real-time networks such as the Controller Area Network (CAN) can reduce their bandwidth requirements in response to periods of high network traffic from sporadic sources. The method derives from advances in the theory of dynamic systems on time scales.

1 Introduction

The rising complexity of many modern engineering systems such as robots, automation systems, and automobiles is gradually demanding that more and more actuators and sensors be available to interact with the system environment. In the past, it was common to give each actuator or sensor a dedicated channel to or from a main computational node, which we refer to as the controller or the control node. However, this situation is giving way to the notion of decentralized control, where controllers, actuators and sensors all communicate via one high-speed real time network backbone. One of the most reliable and well-understood such network protocols is the Controller Area Network (CAN). CAN is widely used by automobile manufacturers in Europe, and has seen application to various robotics and automation systems worldwide [6]-[11]. Though the topics in this paper are not limited to CAN systems, we use CAN as our motivating example without loss of generality.

A real-time network typically has the arrangement shown in figure 1. The principle components are the network itself, along with control nodes, actuator nodes and sensor nodes that send and receive messages over the network. One control node may request input from many sensor nodes (perhaps an entire sensor array) and then generate responses for one or more actuator nodes. Sensor nodes may be polled by the controller or may generate data on their own timeline. Thus, there are generally three types of messages that a CAN will need to handle: (1) high-speed periodic messages; (2) high-speed sporadic messages; and (3) low-speed messages. In the CAN architecture, each message consists of 0 to 8 bytes of data, along with an 11 or 29 bit identifier. All nodes "hear" every message on the bus (CAN is data-selective rather than address-selective). The identifier not only labels the type of data contained in the message, but also specifies the priority of the message. In a message collision, higher-priority messages will transmit first. *High-speed* messages imply that there are (possibly very tight) deadlines beyond which, if a message is not received, undesired effects result.

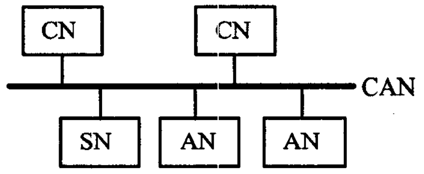

Figure 1: A typical CAN network contains control nodes (CN), sensor nodes (SN) and actuator nodes (AN).

A typical control loop is implemented on a CAN system quite naturally: the control node periodically places a no-data RTR (return transmit request) message on the bus. Sensors nodes responding to the RTR identifier then transmit their readings. The controller computes new actuator values on the basis of the sensor inputs and then places these new values on the bus. Efficiency can be increased in special cases by changing the order of operations (depending on the particulars of the sensor/actuator nodes) but this is the most basic method. A simple timeline appears in figure 2.

There are two potential pitfalls that figure 2 illustrates. First, there is always a delay between when the sensors report their readings to when the actuators respond. This delay (jitter) can cause instabilities but can be minimized by careful message scheduling [1][10][11]. Second, if many

*School of Engineering and Computer Science, Baylor University. Email: Ian_Gravagne@baylor.edu; Robert_Marks@baylor.edu

†Department of Mathematics, Baylor University. Email: John_M_Davis@baylor.edu; Jeffrey_Dacunha@baylor.edu

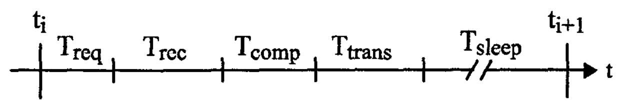

Figure 2: The CAN control loop timeline. During one control period, T_{req} is the RTR request time, T_{rec} the SN response time, T_{comp} the CN computation time, T_{trans} the time it takes the AN to receive its instructions and act, and T_{sleep} the inactive time until the beginning of the next period.

controllers with fixed control periods share the bus, little room for high-speed sporadic messages remains. These can be very important, signaling perhaps the depression of a brake pedal or that the fingers of a robot hand are making/breaking contact with an object. Thus, we next explore how to allow for limited periods of high-speed sporadic traffic by adapting the control period.

2 System Model

The following arguments use results and notation common to a branch of mathematics termed *dynamic equations on time scales*. For readers not familiar with this topic, an extremely brief introduction appears in the Appendix. We start with the assumption that the plant to be controlled can be approximated by a linear system of the form

$$\begin{aligned} \dot{x} &= Ax + Bu, \quad A \in \mathbb{R}^{n\times n}; B \in \mathbb{R}^{n\times m} && (1)\\ u &= Kx, \quad K \in \mathbb{R}^{m\times n}, && (2) \end{aligned}$$

and, for simplicity, that full-state feedback control is available to make the system behave as desired. We next discretize to an isolated time scale $\mathbb{T}$, consisting of non-uniformly spaced points with unknown graininess. The discretization process is described in detail in [4], and yields

$$\begin{aligned} x^{\Delta}(t) &= \left[\frac{e^{A\mu(t)} - I}{\mu(t)} A^{-1}\right](A + BK)x(t) := \mathcal{A}(t)x(t),\\ t &\in \mathbb{T}. \end{aligned} \quad (3)$$

Note that, as discussed in [4], there is no ambiguity due to presence of A^{-1} because the quantity in square brackets above exists for all real A. Furthermore, in the special case that $\mu \equiv 0$, this quantity simply reduces to an identity matrix, leaving $\dot{x} = (A+BK)x$ as expected. The objective now is to generate the graininess $\mu(t)$ – the distance from the current sampling time to the next sampling time, or the control period – dynamically, depending on network traffic and other factors discussed next.

We now repeat some fundamental results regarding the stability of linear systems on arbitrary time scales. The authors of [5] prove that time-invariant linear systems, or a time-varying linear systems that are Jordan reducible, will be exponentially stable if and only if every system eigenvalue belongs to the *set of exponential stability* $\mathcal{S}(\mathbb{T})$. A subset of $\mathcal{S}(\mathbb{T})$, of interest here, is defined as

$$\begin{aligned} \mathcal{S}_{\mathbb{C}}(\mathbb{T}) &= \{\lambda \in \mathbb{C} : \alpha = \\ &\limsup_{t\to\infty} \frac{1}{t - t_0} \int_{t_0}^{t} \frac{\log|1 + \mu(\tau)\lambda(\tau)|}{\mu(\tau)} \Delta\tau < 0\}. \end{aligned} \quad (4)$$

In [3] the preceding result is extended by proving that a positive definite matrix $Q(t)$ exists, such that the Lyapunov function $V = x^T Q x$ leads to exponential stability, if all eigenvalues of $\mathcal{A}(t)$ belong to $\mathcal{S}_{\mathbb{C}}(\mathbb{T})$. Looking at (4), it becomes apparent that the integrand must be negative on average. Furthermore, under the condition that $\mu(t) > 0$ with no limit points, as in this paper, $\mathcal{S}_{\mathbb{C}}(\mathbb{T})$ reduces by a fundamental theorem of time scale calculus [2] to

$$\begin{aligned} \mathcal{S}_{\mathbb{C}}(\mathbb{T}) &= \{\lambda \in \mathbb{C} : \\ &\alpha = \lim_{t\to\infty} \frac{1}{t - t_0} \sum_{t_0}^{t} \log|1 + \mu(\tau)\lambda(\tau)| < 0\}. \end{aligned} \quad (5)$$

(Note that the summation limits are really the ordinals of the associated times. This is a convenient shorthand notation.) To make the problem more tractable, we next note that a sufficient condition for the infinite average in (5) to remain negative is for the moving average over every k points to remain negative. Setting $t_0 = 0$, we then have the desired stability criterion that

$$\frac{1}{t - \rho^k(t)} \sum_{\rho^k(t)}^{t} \log|1 + \mu(\tau)\lambda(\tau)| < 0 \quad \forall t \in \mathbb{T} \quad (6)$$

which further simplifies to

$$P(t) := \prod_{\rho^k(t)}^{t} |1 + \mu(\tau)\lambda(\tau)| < 1. \quad (7)$$

To complete the background, we lastly note that, as long as all eigenvalues of $(A + BK)$ have negative real parts, then there exists some non-zero positive constant $\mu_{\max}$ such that eigenvalues λ_i of $\mathcal{A}(t)$ will give $|1 + \mu(t)\lambda_i\{\mathcal{A}(t)\}| < 1$ if $\mu(t) < \mu_{\max}$ [4]. The region defined by $|1 + \mu\lambda| < 1$ is termed the *Hilger Circle*, a circle in the left-hand complex plain that passes through the origin, with center at $-\frac{1}{\mu}$. Note that, for systems of non-constant graininess, the Hilger Circle changes radius dynamically, and thus one interpretation of (4) is that system eigenvalues must reside in the circle "most" of the time, on average. Instances when $\mu(t) \geq \mu_{\max}$ are instantaneously unstable, but on average do not affect the overall stability if $\lambda_i \in \mathcal{S}_{\mathbb{C}}(\mathbb{T})$. We use this idea, in the simplified form of (7), to our advantage next.

3 Adaptive Sampling

One consequence of the discussion above is that the control system will not lose stability if the eigenvalues fall outside of the Hilger Circle for limited periods of time. One way this might happen is to (intentionally) permit the sampling period to occasionally rise above $\mu_{\max}$, thus rendering $|1+\mu(t)\lambda(t)| \geq 1$ during that period. If the window for $P(t)$ is large enough, then a number of such infractions can be tolerated while still maintaining $P(t) < 1$. In fact, the number of tolerable unstable periods can be maximized if the controller is nominally operating with a period that minimizes $|1+\mu(t)\lambda(t)|$, which we term μ^*. This minimizes $P(t)$ and gives the system more head room to allow for unstable periods while still maintaining overall stability. If, however, $P(t)$ rises too much (i.e. approaches 1 too closely) it is conversely possible to monitor P and reduce the nominal sampling period below μ^* until P falls below some threshold.

Naturally, the presence of dense network traffic is one obvious condition sufficient to trigger a rise in the control period. There are conceivably many ways in which network traffic metrics could be used to raise $\mu(t)$, but if the network is a CAN, one particularly simple way is to set T_{sleep} to a constant (refer to figure 2), rather than setting μ (or $t_{i+1}-t_i$) constant as is usually done. T_{sleep} would be determined so that, in the absence of high-priority traffic, the nominal desired period is equal to the optimal period, μ^*. This value is called T^*_{sleep}. When network traffic becomes dense, the durations of T_{req}, T_{rec} and T_{trans} rise, and then the actual period rises above the desired period, $\mu(t) \geq \mu^*$. This is tolerable for a time – and may be tolerable indefinitely – unless $P(t)$ gets too close to unity, in which case the controller can switch to a lower desired period (by lowering T_{sleep} to some predetermined $\hat{T}_{sleep}$), below the optimal one. This yields a particularly simple update rule for $\mu_d(t)$ that can be implemented even on small embedded CAN controllers:

1. Measure the actual time duration $T_{req}+T_{rec}+T_{comp}+T_{trans}$.
2. If $P(t) < \bar{P}$ (some threshold < 1) then sleep for $T_{sleep} = T^*_{sleep}$. Otherwise $T_{sleep} = \hat{T}_{sleep}$.
3. Calculate $\mu = T_{req} + T_{rec} + T_{comp} + T_{trans} + T_{sleep}$.
4. Calculate $\max_i |1+\mu(t)\lambda_i\{\mathcal{A}(t)\}|$ and then update $P(t)$.
5. Sleep. Upon wake, repeat at step 1.

Note that step 4 appears computationally intensive, but in fact a range of values for $|1+\mu\lambda|$ versus $\mu(t)$ can be computed ahead of time and stored in a lookup table for fast reference. Step 1 is possible because CAN is a real-time network: once the controller knows that the actuator messages have left the transmit queue, it automatically follows that they have been received by the actuator nodes, after which the timing becomes strictly deterministic. The overall effect of the algorithm is to specify a desired period of

$$\mu_d(t) = \min(T_{req} + T_{rec} + T_{trans}) + T_{comp} + T_{sleep}, \quad (8)$$

which only occurs if there is no competing network traffic. Since T_{comp} is fixed and T_{sleep} takes on one of two values, $\mu_d(t)$ takes on one of two values as well.

4 Simulation

For an example, we choose the (inherently unstable) system

$$A = \begin{bmatrix} 0 & 1 \\ -4 & 1 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, K = \begin{bmatrix} 0 & -2 \end{bmatrix}. \quad (9)$$

The closed loop continuous system has poles at $\lambda_i = -0.5 \pm 1.94j$, which are stable although obviously not "optimal". (We want to preserve some interesting dynamics.) Off-line computation finds that the minimum of $\max_i |1+\mu(t)\lambda_i\{\mathcal{A}(t)\}|$ occurs at $\mu^* = 0.581$, and that $\mu_{\max} = 0.837$. Without loss of generality, we set $T_{comp} = 0$, and assume that, with no competing network traffic, $T_{req} + T_{rec} + T_{trans} = 0$. This implies that $T^*_{sleep} = 0.581$, and we choose $\hat{T}_{sleep} = 0.2$.

Figures 3 and 4 illustrate scenarios where a burst of network traffic occurs between $15 < t < 45$. In both figures, the first 3 seconds show $\mu = \mu_d = 0.2$ because, while there is theoretically no traffic in this interval, P is initialized to 1 which exceeds the programmed threshold $\bar{P} = 0.25$. The averging window for $P(t)$ is 25 samples. During the high-traffic interval, figure 3 shows that $P < \bar{P}$ at all times, and therefore the controller tolerates the additional delays even though 14 periods exceed the instantaneous stability limit, $\mu > \mu_{\max}$. Thus, in this interval, only 42 control periods are necessary versus 52 if the controller operated on

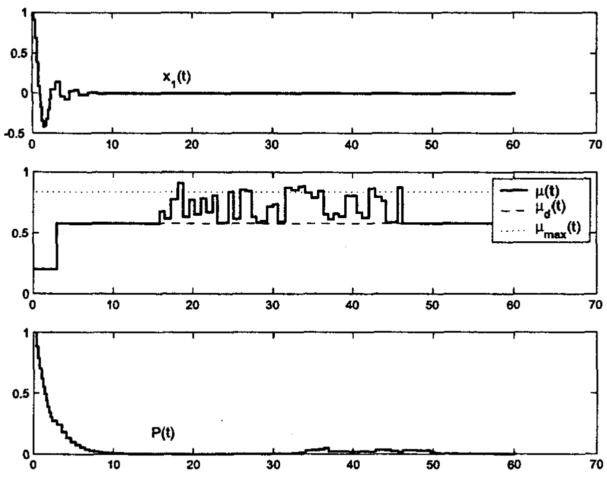

Figure 3: The system response to a burst of traffic.

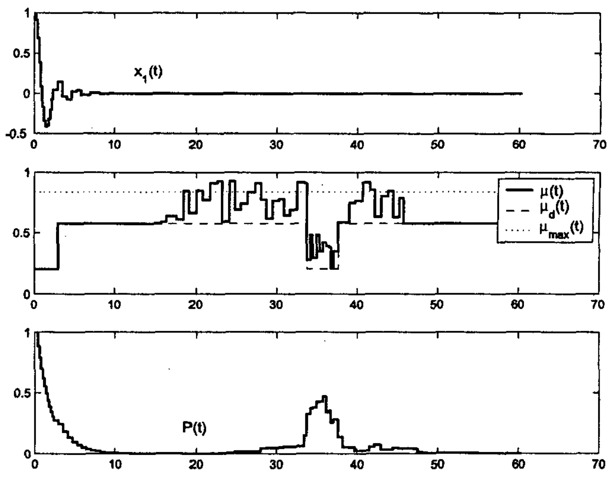

Figure 4: System response to another burst of traffic. Note that $P(t) > 0.25$ for a brief time.

a fixed period basis at the optimal period, a bandwidth reduction of 19.2%, leaving additional bandwidth available to the sporadic messages on the bus. In figure 4, traffic becomes dense enough that $P > \bar{P}$ briefly, and the controller switches to $\mu_d = 0.2$. This has the effect of quickly reducing the magnitude of P, and during the high-traffic interval only 45 control periods are necessary, a 13.5% reduction in bandwidth. Again, the control period exceeds the instantaneous stability limit of $\mu_{\max}$ for 14 periods. Significantly higher bandwidth reductions would be possible under this algorithm compared against a system using the "rule of 10" (i.e. that the controller should sample at least 10 times faster than the smallest time constant in the system).

5 Conclusion

The paper presents a simple algorithm to permit controllers on real-time distributed control networks such as CAN to adapt their sample periods in response to bursts of high-priority sporadic traffic. Using recent results from the mathematical field of dynamical equations on time scales, the analysis shows that the algorithm can maintain stability on average even when the delays are so large that the system is instantaneously unstable. As robotics and automation systems grow ever more complex and require more sensors and actuators, algorithms such as this may help to stave off the inevitable bandwidth crunch.

More analysis is necessary to characterize the behavior of this algorithm under various conditions. The analysis here does not take into account the problem of jitter, and it is not clear that any of the commonly used message priority scheduling schemes will work well in an adaptive sampling situation: such algorithms have always been designed to maximize synchronicity rather than allowing for the purposeful introduction of asynchronicity, as was done here. Further subjects of study are improved adaptation laws that can vary $\mu_d(t)$ continuously, and examination of the effects of the present technique on nonlinear plants.

6 Appendix

A thorough introduction to dynamic equations on time scales is beyond the scope of this appendix; however, Bohner and Peterson [2] have written an excellent overview of the subject. In short, the theory springs from the doctoral dissertation of S. Hilger in 1988. Starting the early 1990's, the theory began to grow until in 2001 a complete treatise on the subject appeared [2].

A *time scale* $\mathbb{T}$ in this context is defined as an arbitrary non-empty closed subset of the real numbers. Thus time scales can be any of the usual integer subsets (e.g. $\mathbb{Z}$ or $\mathbb{N}$), the entire real line ($\mathbb{R}$) or any combination of discrete points unioned with continuous intervals. The bulk of engineering systems theory to date rests on two time scales, $\mathbb{R}$ and $\mathbb{Z}$ (or more generally $h\mathbb{Z}$, meaning discrete points separated by distance h). However, as this paper illustrates, there are occasional instances when necessity or convenience dictates the use of an alternate time scale.

The question of how to how to approach the study of *dynamic systems on time scales* then becomes relevant, and in fact the majority of research on time scales so far has focused on expanding and generalizing the vast suite of tools available to the differential and difference equation theorist.

The paper makes use of a few essential definitions and theorems from this body of work, which we discuss now.

Definition 1 *The* ***forward jump operator*** $\sigma(t) : \mathbb{T} \to \mathbb{T}$ *and the* ***backward jump operator*** $\rho(t) : \mathbb{T} \to \mathbb{T}$ *are given by*

$$\sigma(t) = \inf_{s\in\mathbb{T}}\{s > t\}, \quad \rho(t) = \sup_{s\in\mathbb{T}}\{s < t\}. \tag{10}$$

The ***graininess function*** $\mu(t) : \mathbb{T} \to [0, \infty)$ *is given by*

$$\mu(t) = \sigma(t) - t. \tag{11}$$

Evidently, since the forward jump operator returns the next point in the time scale, the graininess can be visualized as the step size. Note that, for a closed interval of $\mathbb{R}$, $\sigma(t) = 0$ except at the rightmost point, and therefore $\mu(t) = 0$ except at the rightmost point. Thus it becomes clear that time scales consist of collections of two types of elements, *scattered points* (i.e. points where $\rho(t) \neq t$ and $\sigma(t) \neq t$) and *dense points* (i.e. points where $\rho(t) = t$ or $\sigma(t) = t$.) At the endpoints of continuous intervals, we have left- and right-dense points as well. Points that are right- and left-scattered are termed *isolated.*

Definition 2 *For some function* $f : \mathbb{T} \to \mathbb{R}$, *the* ***delta derivative*** *of* $f(t)$, *designated* $f^{\Delta}(t)$, *is the number (when it exists) with the property that there is some neighborhood* U *of* t *where*

$$\left|[f(\sigma(t)) - f(s)] - f^{\Delta}(t)[\sigma(t) - s]\right| \leq \epsilon\,|\sigma(t) - s| \quad \forall s \in U,$$

and $\epsilon > 0$.

We say that f is delta differentiable provided the delta derivative f^{Δ} exists for all $t \in \mathbb{T} - \{\max(\mathbb{T})\} := \mathbb{T}^{\kappa}$, i.e. the timescale minus its right-most point if that point exists. For simplicity in this paper we omit the $\mathbb{T}^{\kappa}$ notation because of the convention that, if the time scale does not have a maximum, $\mathbb{T}^{\kappa} = \mathbb{T}$. Not surprisingly, the condition for existence of the delta derivative is simply that f be continuous over all closed, continuous intervals, if there are any (i.e. all subsets of $\mathbb{R}$). If this is the case, the delta derivative is well defined by the equality

$$f^{\Delta} = \frac{f(\sigma(t)) - f(t)}{\mu(t)}. \tag{12}$$

Needless to say, the generalizations of the usual rules of differentiation are not always as simple as their continuous cousins (e.g. $[x^3(t)]^{\Delta} \neq 3x^2(t)$).

Of course, along with differentiation one would like to have integration. For this, we require some mild technical conditions [2], including that f be *regulated,* meaning that its right- and left-sided limits exist at any right- and left-dense points in $\mathbb{T}$.

Theorem 3 *Let* f *be regulated. Then there exists a function* F *and region of differentiation* D *such that*

$$F^{\Delta}(t) = f(t), \quad t \in D \tag{13}$$

Definition 4 *A function* $F : \mathbb{T} \to \mathbb{R}$ *is called an* ***antiderivative*** *of* f *provided*

$$F^{\Delta}(t) = f(t) \quad \forall t \in \mathbb{T}^{\kappa}. \tag{14}$$

Theorem 5 *Every right-dense continuous function has an antiderivative. If* $t_0 \in \mathbb{T}$, *then*

$$F(t) = \int_{t_0}^{t} f(\tau)\Delta\tau \quad t \in \mathbb{T}. \tag{15}$$

As one would hope, the theorems above reveal that, in the continuous case $\mathbb{T} = \mathbb{R}$, delta antiderivatives and integrals are the usual antiderivatives and definite integrals from standard calculus. When $\mathbb{T} = \mathbb{R}$, these quantities correspond to indefinite and definite sums often seen in the study of difference equations. Without further exposition, the usual properties of integrals hold as well, including linearity and homogeneity. However, as with derivatives, the usual integration "rules of thumb" do not hold. These definitions lead to a foundational theorem of time scale calculus, which states:

Theorem 6 *If* f *is right-dense continuous, then*

$$\int_{t}^{\sigma(t)} f(\tau) = \mu(t)f(t). \tag{16}$$

This theorem, along with linearity of the time scale integral, is what equates the integral to a sum in the case that $\mathbb{T}$ consists only of isolated points.

From the definitions above, the next obvious step is to investigate linear and the non-linear time scale differential equations, e.g. systems of the form $x^{\Delta\Delta} + ax^{\Delta} + bx = f$ and beyond. Since the time scale itself is often allowed to be arbitrary (or occasionally must adhere to mild assumptions), the theoretical foundations that underpin the study of dynamic equations on time scales are extremely broad. The types of time scales in this paper are relatively "tame" in comparison to the variety that are possible, but nevertheless, time scale theory provides a rigorous and holistic technique by which we can study non-uniform sampling problems with relative ease.

References

[1] Audsley, N.; et. al; "Applying new scheduling theory to static priority pre-emptive scheduling," J. Software Engineering, Sept. 1993, pp. 284-292.

[2] Bohner, Martin; Peterson, Allan; Dynamic Equations on Time Scales, Birkhauser Boston, 2001.

[3] Dacunha, Jeffrey; "Stability for time-varying linear dynamical systems on time scales," internal report, department of Mathematics, Baylor University, October 2003.

[4] Gravagne, Ian; Davis, John; Dacunha, Jeffrey; "A unified approach to discrete and continuous high-gain adaptive controllers using time scales," submitted, SIAM J. Control and Optimization.

[5] Potzsche, C.; Siegmund, S.; Wirth, F.; "A spectral characterization of exponential stability for linear time-invariant systems on time scales," J. Discrete and Continuous Dynamical Systems, v.9(5), Sept. 2003, pp. 1223-1241.

[6] Tindell, Ken; Burns, Alan; "Guaranteeing message latencies on control area network (CAN)," Proc. 1st International CAN Conference, 1994.

[7] Wargui, M.; Tadjine, M.; Rachid, A.; "A scheduling approach for decentralized mobile robot control system," IEEE Int'l Conf. Intelligent Robots and Systems, v.2, 1997, pp. 1138-1143.

[8] Wargui, M.; Rachid, A.; "Application of controller area network to mobile robots," Proc. Mediterranean Electromechanical Conf., v.1, 1996, pp. 205-207.

[9] Wiesspeiner, W.; Windischbacher, E.; "Distributed intelligence to control a stair-climbing wheelchair," Proc. IEEE Int. Conf. Engineering in Medicine and Biology, v.17(2), 1995, pp. 1173-1174.

[10] Zuberi, Khawar; Shin, Kang; "Non-preemtive scheduling of messages on controller area network for real-time control applications," Proc. Real-Time Technology and Applications, 1995, pp. 240-249.

[11] Zuberi, Khawar; Shin, Kang; "Scheduling messages on controller area network for real-time CIM applications," IEEE Trans. Robotics and Automation, v.13(2), April 1997, pp. 310-314.