کاهش زمان پاسخ در سیستم‌های جاسازی شده بلادرنگ

بر پایه Controller Area Network

رضا علایی [۱]، پیمان معلم [۲] و علی بهلولی[۳]

[۱]دانشجوی مقطع دکتری گروه مهندسی برق دانشکده فنی مهندسی دانشگاه اصفهان، re.alaei@eng.ui.ac.ir

[۲] استاد گروه مهندسی برق دانشکده فنی مهندسی دانشگاه اصفهان، p_moallem@eng.ui.ac.ir

[۳] استادیار گروه مهندسی معماری دانشکده کامپیوتر و فن‌آوری اطلاعات دانشگاه اصفهان ، bohlooli@eng.ui.ac.ir

چکیده - پروتکل *CAN* (*Controller Area Network*) یک استاندارد باس داده است که به صورت گسترده در صنایع استفاده می‌شود. یکی از ضعف‌های پروتکل *CAN* در سیستم‌های بلادرنگ تغییر در زمان پاسخ به دلیل اعمال مکانیزم *bit stuffing* است. به منظور کاهش *stuff bit* ها روش‌های مختلفی پیشنهاد شده است. مقاله حاضر به بررسی استفاده همزمان دو روش ماسک *XOR* و استفاده هوشمندانه شناساگر در ساختار پیام در لایه کاربرد پرداخته است. میزان تاثیر این روش با اعمال آن به دو گروه داده‌های تصادفی و داده‌های مربوط به سیستم واقعی مورد بررسی قرار گرفته است. در انتها نیز پتانسیل روش ترکیبی پیشنهاد شده بر کاهش بدترین زمان پاسخ یک سیستم واقعی مورد بررسی قرار گرفته است.

کلید واژه- الگوریتم CAN، Bit stuffing، بدترین زمان پاسخ سیستم، Real-time.

۱- مقدمه

در سال‌های اخیر به دلیل استفاده روزافزون از سیستم‌های بلادرنگ، راه‌های مخابره پیام که تامین کننده این نیاز باشند به سرعت در حال افزایش است. پروتکل CAN یکی از راه‌های ارتباط سریال است که اولین بار توسط Robert Bosch Gmbh و برای موتور خودرو پیشنهاد گردید[۱]. با وجود اینکه CAN در ابتدا به منظور استفاده در خودروها طراحی گردید ولی با توجه به قیمت ارزان، قابلیت اطمینان و امکان کار در محیط‌های با میزان نویز بالا به سرعت در سایر بخش‌های صنعت و بخش‌های کنترلی مبتنی بر سیستم‌های بلادرنگ مورد استفاده قرار گرفت[۲, ۳].

قابلیت پیش‌بینی رفتار زمانی، یکی از مهم‌ترین ویژگی‌های سیستم‌های بلادرنگ است. این در حالی است که برخی از فرایندهای مورد استفاده در پروتکل CAN برآورده شدن این خواسته را مختل می‌کند. یکی از این فرایندها bit stuffing است که به منظور سنکرون‌سازی پالس ساعت در سیستم و همچنین جلوگیری از وقوع الگوهای مشابه الگوی مربوط به وقوع خطا در سیستم به‌کار می‌رود. فرایند bit stuffing باعث عدم امکان پیش‌بینی دقیق بدترین زمان پاسخ به در پیام‌ها است. به منظور رهایی از وقوع bit stuffing روش‌های متعددی چون استفاده از ماسک XOR ، استفاده از بیت معکوس (inversion bit stuffing mechanism)، مدوله سازی ۸ به ۱۱ (eight to eleven modulation)، متمم بیت سوم (third bit complement)، Zero Stuff-bit CRC و انتخاب هوشمندانه شناساگر (carefully selecting priorities) ارائه شده است[۴-۱۰].

در مقاله حاضر از دو روش ماسک XOR و انتخاب هوشمندانه شناساگر، به طور همزمان استفاده شده است که در ادامه با عنوان روش ترکیبی از آن یاد می‌شود . روش ماسک XOR شامل انجام عملیات منطقی XOR بخش داده هر پیام با الگوی ...101010 به منظور از بین بردن بیت‌های یکسان پشت سر هم است. روش انتخاب هوشمندانه شناساگر شامل اجتناب از انتخاب شناساگرهایی است که به دلیل وجود بیت‌های یکسان پشت سرهم مستلزم عملیات bit stuffing هستند.

روش ترکیبی یاد شده به دو دسته داده شامل داده‌های تصادفی و داده‌های واقعی مربوط به یک سیستم پرنده اعمال خواهد شد. همچنین بدترین زمان پاسخ برای سیستم واقعی قبل و بعد از اعمال روش ترکیبی محاسبه و میزان بهبود ناشی از اعمال روش ترکیبی ارائه می‌شود. که شاهد ۵ تا ۱۰ درصدی عملکرد سیستم خواهیم بود.

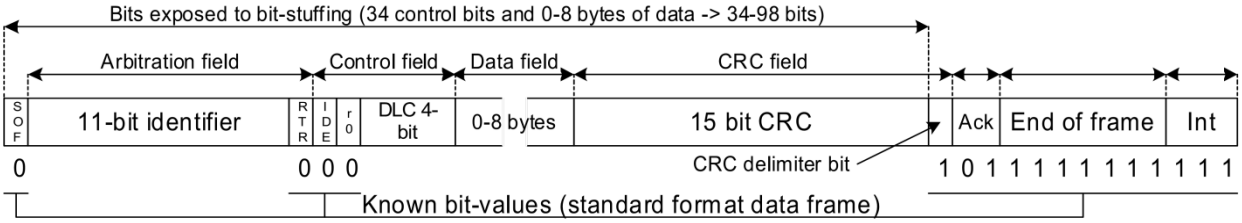
در ادامه و در بخش ۲ به معرفی اجمالی پروتکل CAN پرداخته شده است. بخش ۳ توضیحاتی پیرامون فرایند bit stuffing ارائه می‌دهد. بخش ۴ مشتمل بر روابط حاکم بر محاسبه بدترین زمان پاسخ است. بخش ۵ شامل نتایج به دست آمده و در بخش ۶ به نتیجه‌گیری می‌پردازیم.

۲- مروری بر پروتکل CAN

CAN یک باس داده چند اربابی است که به منظور کار در سرعت‌های تا ۱ Mbit/s طراحی شده است. ساختار کلی هر پیام در CAN به ترتیب شامل یک شناساگر منحصربه‌فرد (برای CAN 2.0 A شناساگر ۱۱ بیتی و برای CAN 2.0 B شناساگر ۲۹ بیتی است)، قسمت کنترل، داده‌ها، CRC و فیلد پایان فریم می‌باشد. شناساگر علاوه بر تعیین اولویت دستیابی به باس در هنگام درخواست همزمان گره‌ها، فیلتر شدن پیام‌های ناخواسته در گره‌ها را نیز ممکن می‌کند. قسمت کنترل وظیفه تعیین نوع پیام (Data یا Remote) و طول داده درون پیام که از صفر تا هشت قابل تنظیم است را به عهده دارد. فیلد داده حاوی اطلاعات درون پیام می‌باشد. قسمت CRC جهت صحت سنجی پیام استفاده می‌شود و فیلد پایان، اتمام پیام را به گیرنده‌ها اطلاع می‌دهد[۶].

ساختار یک پیام CAN در شکل ۱ نشان داده شده است.

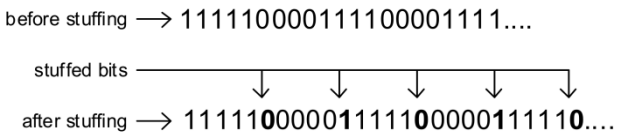
ارسال داده توسط هر گره زمانی میسر است که باس بیکار باشد. در صورتی که باس بیکار باشد گره‌هایی که قصد ارسال داده دارند، شناساگر خود را به صورت تک بیتی و پشت سر هم ارسال می‌کنند. در پروتکل CAN، بیت ۰ را بیت غالب و بیت ۱ را بیت مغلوب می‌نامند. باس طوری طراحی شده است که در صورتی که چند گره به طور همزمان در حال ارسال شناساگر خود بر روی باس باشند و تنها یکی از آن‌ها ۰ باشد، مقدار باس ۰ خواهد شد. در واقع باس مانند یک گیت AND بزرگ عمل می‌کند[۱۱].



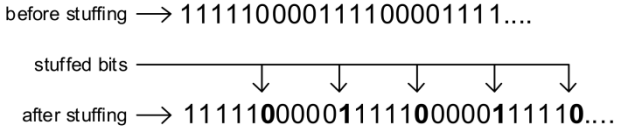
شکل ۱- فرمت استاندارد پیام CAN.

۳- معرفی فرایند Bit stuffing

در پروتکل CAN شش بیت پشت سر هم یکسان (۰۰۰۰۰۰ یا ۱۱۱۱۱۱) به منظور مخابره خطا به کار می‌رود. جهت جلوگیری از ایجاد این الگو در ساختار پیام و ایجاد تغییرات بیتی لازم جهت همگام سازی گذرگاه داده، پس از هر پنج بیت یکسان، یک بیت مخالف (stuff bit) اضافه می‌شود. در واقع گره ارسال کننده پیام پس از هر پنج بیت یکسان، یک بیت مخالف قرار می‌دهد و در محل گره گیرنده این stuff bit حذف می‌شود تا پیام اصلی استخراج شود.



شکل ۲ فرایند bit stuffing را نشان می‌دهد.



شکل ۲- فرایند bit stuffing.

بنابراین مکانیزم bit stuffing موجب می‌شود تا طول پیام بزرگتر از طول مورد انتظار باشد. طول پیام برای ساختار استاندارد CAN (شناساگر ۱۱ بیتی) بدون در نظر گرفتن bit stuffing با استفاده از رابطه (۱) قابل استخراج است:

$$8s + 47 \qquad (۱)$$

که در آن s تعداد بایت داده و ۴۷ تعداد بیت‌های کنترلی است. با توجه به اینکه تنها ۳۴ بیت (از SOF تا CRC) از ۴۷ بیت تحت فرایند bit stuffing قرار می‌گیرند بیشترین طول پیام در صورت وقوع bit stuffing از رابطه (۲) قابل محاسبه است.

$$8s + 47 + \left\lfloor \frac{34+8s}{5} \right\rfloor \qquad (۲)$$

که $\lfloor a/b \rfloor$ تابع floor است و مقدار صحیح a/b را نتیجه می‌دهد [۷, ۱۳].

۴- آنالیز بدترین زمان پاسخ

بدترین زمان پاسخ، R_m،که به صورت بیشترین زمان مورد نیاز برای رسیدن پیام به مقصد تعریف می‌شود توسط رابطه (۳) قابل محاسبه است.

$$R_m = J_m + w_m + C_m \qquad (۳)$$

که در آن J_m بیشترین زمان مورد نیاز برای در صف قرار دادن پیام توسط یک task است. w_m بیشترین زمان در صف بودن ناشی از دو عامل تداخل (درخواست باس توسط پیام‌های با اولویت بالاتر) و انسداد B_m (ارسال پیام با اولویت پایین‌تر که اندکی قبل از پیام مورد نظر باس را در اختیار گرفته است) می‌باشد.

C_m بیشترین زمان پاسخ برای ارسال پیام به دلیل محدودیت‌های فیزیکی باس است که برای باس با سرعت ۱Mbit/s که دارای بیشترین طول داده (۸ بایت) است برابر با ۱۳۰ میکروثانیه است. با استفاده از رابطه (۲)، C_m به صورت رابطه (۴) قابل محاسبه است که در آن τ_{bit} زمان ارسال یک بیت است و برای سرعت باس برابر با ۱Mbit/s برابر ۱ میکروثانیه است.

$$C_m = \left(8s + 47 + \left\lfloor \frac{34+8s}{5} \right\rfloor\right)\tau_{bit} \qquad (۴)$$

w_m را می‌توان توسط رابطه (۵) محاسبه نمود.

$$w_m = B_m + \sum_{\forall j \epsilon hp(m)} \left\lceil \frac{w_m + j_j + \tau_{bit}}{T_j} \right\rceil C_j \qquad (۵)$$

که $\lceil a/b \rceil$ تابع ceiling است و کوچکترین عدد صحیح بزرگتر از a/b را نتیجه می‌دهد. $hp(m)$ مجموعه‌ی پیام‌های با اولویت بالاتر از پیام m، T_j دوره تناوب پیام j، J زمان $jitter$ و B_m زمان انسداد بوده و برابر با بیشترین زمانی است که ارسال پیام به دلیل اشغال بودن باس توسط پیام با اولویت پایین‌تر به تعویق می‌افتد. B_m توسط رابطه (۶) قابل محاسبه است که $lp(m)$ مجموعه‌ی پیام‌های با اولویت پایین‌تر از پیام m است.

$$B_m = \max_{\forall k \in lp(m)} (c_k) \qquad (۶)$$

همانطور که ملاحظه می‌شود w_m در رابطه (۵) در هر دو طرف تساوی وجود دارد. به منظور حل این معادله می‌توان از رابطه بازگشتی به صورت زیر استفاده نمود.

$$w_m^{n+1} = B_m + \sum_{\forall j \epsilon hp(m)} \left\lceil \frac{w_m^n + j_j + \tau_{bit}}{T_j} \right\rceil C_j \qquad (۷)$$

به عنوان مقدار اولیه می‌توان $w_m^0 = 0$ را برابر با صفر در نظر گرفت. رابطه بازگشتی فوق تا جایی که $w_m^{n+1} = w_m^n$ ادامه می‌یابد[۷-۱۵].

۵- روش پیشنهادی

در این بخش ابتدا نتایج مربوط به اعمال هر یک از روش‌های ماسک XOR و انتخاب هوشمندانه شناساگر به طور مجزا ارائه شده است و سپس نتیجه استفاده از روش ترکیبی پیشنهادی بر روی بدترین زمان پاسخ یک سیستم واقعی ارائه شده است.

۵-۱- انتخاب هوشمندانه شناساگر

با توجه به اینکه سرایند پیام (شش قسمت ابتدایی ساختار پیام) تحت bit stuffing قرار می‌گیرد با انتخاب هوشمندانه شناساگر می‌توان از وقوع bit stuffing در سربرگ پیام اجتناب نمود. هر چند استفاده از این روش مانع از استفاده از مجموعه کامل شناساگرها ($۲^{۱۱}$ شناساگر) می‌شود لیکن در عمل معمولا تعداد زیر سیستم‌ها و پیام‌های متنوع مورد نیاز در یک سیستم کمتر از این تعداد بوده و با کمبود تعداد شناساگر روبرو نمی‌شویم. نتایج حاصل از استفاده از این روش در جدول ۱ ارائه شده است. با توجه به نتایج جدول ملاحظه می‌شود که با اعمال این روش می‌توان stuff bitها را به کلی حذف نمود یا به مقدار حداقل ۱ کاهش داد.

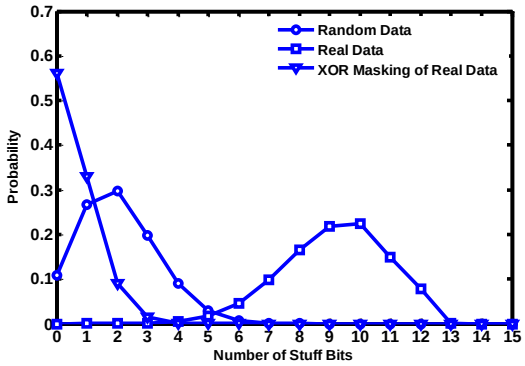
جدول ۱- تعداد شناساگرهای از صفر تا سه stuff bit برای پیام‌های با طول داده متفاوت

Number of Stuff-bits	CAN message data length								
	0	1	2	3	4	5	6	7	8
0	0	0	0	0	745	745	745	745	1131
1	1332	1436	1490	1490	1005	1005	1005	1005	765
2	634	560	520	520	279	279	279	279	145
3	81	51	38	38	19	19	19	19	7

بر طبق جدول، شناساگرها را می‌توان به دو گروه اصلی تقسیم نمود: گروه اول شناساگرهای مربوط به پیام‌های حاوی ۰ تا ۳ بایت داده (DLC = 0-3) است. این گروه حداقل دارای یک stuff bit هستند. گروه دوم شامل بیش از ۳ بایت داده می‌باشد. در این گروه حذف کامل stuff bit امکان‌پذیر است که در این صورت در پیام‌های شامل ۴ تا ۷ بایت داده و پیام‌های حاوی ۸ بایت داده، تعداد شناساگرهای مجاز به ترتیب به ۷۴۵ و ۱۱۳۱ محدود می‌شود.

۵-۲- اعمال ماسک XOR

روش اعمال ماسک XOR به ۸۰۰۰۰۰ داده تصادفی و ۸۰۰۰۰۰ داده واقعی اعمال گردید. هر دو گروه داده شامل ۸ بایت داده در ساختار پیام‌های خود هستند. نتایج مربوط به احتمال وقوع bit stuffing با تعداد مشخص برای هر دو گروه داده قبل و پس از اعمال این روش در شکل ۳ نشان داده شده است.



شکل ۳- احتمال وقوع تعداد stuff bit مشخص برای داده‌های تصادفی و واقعی با طول داده ۸ بایت قبل و بعد از اعمال ماسک XOR.

همانطور که در شکل مشاهده می‌شود تعداد stuff bitها در داده‌های واقعی پیش از اعمال ماسک بسیار بیشتر از داده‌های تصادفی است. علت این امر احتمال بیشتر شکل‌گیری الگوهای بیتی مشابه (صفرهای پشت سر هم یا یک‌ها پشت سرهم) در داده‌های واقعی است. در واقع در سیستم‌های واقعی داده‌ها معمولا اعداد صحیح کوچک هستند که همین امر موجب ایجاد الگوهای بیتی مشابه پشت سر هم می‌شود. در حالی که در ارتباط با داده‌های تصادفی امکان ۰ یا ۱ بودن یک بیت، یکسان و برابر با ۵۰٪ است و مقدار هر بیت مستقل از مقدار بیت‌های مجاور است که نتیجه آن احتمال کمتر بروز داده‌های دارای الگوی بیتی مشابه پشت سر هم خواهد بود. با اعمال روش ماسک XOR تغییر چندانی بر نتایج داده‌های تصادفی رخ نمی‌دهد لیکن در ارتباط با داده‌های واقعی تاثیر چشمگیری بر احتمال وقوع stuff bitها رخ می‌دهد. در این حالت بیشترین تعداد stuff bitها از مقدار ۱۳ (۱۵ بر اساس رابطه (۲)) به مقدار ۳ کاهش می‌یابد.

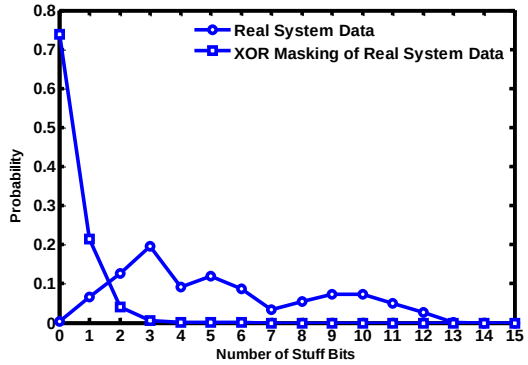
به منظور بررسی کارایی روش ماسک XOR، این روش بر روی یک سیستم واقعی با سرعت ۱۷۰ کیلو بیت بر ثانیه حاوی پیام‌های با طول داده متفاوت نیز اعمال گردید (به دلیل عدم دسترسی به یک بانک داده واقعی استاندارد از داده‌های یک سیستم واقعی خاص استفاده شده است). سیگنال‌های مربوط به این سیستم در جدول ۲ نشان داده شده است. نتایج قبل و بعد از اعمال ماسک در شکل ۴ نشان داده شده است. در اینجا نیز اعمال ماسک تاثیر چشمگیری در تعداد stuff bitها داشته و موجب تنزل آن از مقدار ۱۳ به ۳ می‌شود.

جدول ۲- سیگنال‌های سیستم واقعی

Subsystem name	Data frame size (bytes)	Period (ms)
Gyroscope	6	10.0
Accelerometer	6	10.0
Magnetometer	6	10.0
Barometer	4	10.0
Act	8	10.0
Act feedback	8	10.0
Battery	4	1000.0

۵-۳- ارزیابی روش پیشنهادی

روش ترکیبی پیشنهادی بر روی سیستم معرفی شده در جدول ۲ اعمال گردید. نتایج مربوط به بدترین زمان پاسخ قبل و پس از اعمال روش در جدول ۳ نشان داده شده است. مقایسه نتایج حاکی از کاهش بدترین زمان پاسخ پس از اعمال روش است. در واقع کاهش تعداد stuff bit ها به طور مستقیم بر روی بدترین زمان پاسخ موثر است (طبق رابطه (۳)). محاسبات بدترین زمان پاسخ با استفاده از رابطه (۳) و شکل ۴ انجام شده است.



شکل ۴- احتمال وقوع تعداد stuff bit مشخص برای داده‌های واقعی با طول داده متفاوت (سیگنال‌های جدول ۲) قبل و بعد از اعمال ماسک XOR.

جدول ۳- نتایج مربوط به آنالیز بدترین زمان پاسخ مربوط به سیستم واقعی (سیگنال‌های جدول ۲)

Subsystem name	Data frame size (bytes)	Period (ms)	R_m(ms) before applying the method	R_m(ms) after applying the method
Gyroscope	6	10.0	1.44118	1.35882
Accelerometer	6	10.0	2.11765	1.95294
Magnetometer	6	10.0	2.79412	2.54706
Barometer	4	10.0	3.35294	3.04706
Act	8	10.0	4.14706	3.73529
Act feedback	8	10.0	4.94118	4.42353
Battery	4	1000.0	5.5	4.92353

کلیه زمان‌های ارائه شده بر حسب میلی‌ثانیه هستند. طبق معادله $\lfloor (Control\ field = 19)/5 \rfloor$ و شکل ۳ قبل از اعمال روش ترکیبی تعداد stuff bitها در سرآیند پیام و قسمت داده به ترتیب برابر ۳ و ۱۳ است در حالی که پس از اعمال روش ترکیبی این مقادیر به ۰ (کلیه پیام‌ها دارای طول داده بزرگ‌تر از سه بایت هستند) و ۳ کاهش می‌یابد. بهبود ۵ تا ۱۰ درصدی بدترین زمان پاسخ در جدول ۳ مشهود است.

۶- نتیجه‌گیری

در این مقاله روشی جدید بر اساس انتخاب هوشمندانه شناساگر و اعمال ماسک XOR به منظور کاهش تعداد stuff bit ها و درنتیجه بهبود بدترین زمان پاسخ ارائه شده است . کارایی روش ارائه شده برای دو مجموعه داده تصادفی و واقعی ارزیابی گردید. سپس با استفاده داده‌های یک سیستم واقعی پتانسیل روش پیشنهاد شده بر کاهش بدترین زمان پاسخ بررسی و بهبود ۵ تا ۱۰ درصدی پاسخ سیستم مشاهده شد.

مراجع

[۱] I. ISO, "Road Vehic les—Interchange 0f Digital In formation Controller Area Networ k (CAN) for High Speed Communication (First Edition)," ed: 1993, 1898.

[۲] S. Misbahuddin and N. Al-Holou, "Efficient data communication techniques for controller area network (CAN) protocol," in *Computer Systems and Applications, 2003. Book of Abstracts. ACS/IEEE International Conference on*, 2003, p. 22.

[۳] K. M. Zuberi and K. G. Shin, "Non-preemptive scheduling of messages on controller area network for real-time control applications," in *Real-Time Technology and Applications Symposium, 1995. Proceedings*, 1995, pp. 240-249.

[۴] K. Tindell, H. Hansson, and A. J. Wellings, "Analysing real-time communications: controller area network (CAN)," in *Real-Time Systems Symposium, 1994., Proceedings.*, 1994, pp. 259-263.

[۵] M. Nahas and M. J. Pont, "Using XOR operations to reduce variations in the transmission time of CAN messages: A pilot study," in *Proceedings of the Second UK Embedded Forum* ,2005 ,pp. 4-17.

[۶] T. Nolte, H. Hansson, and C. Norström, "Minimizing CAN response-time jitter by message manipulation," in *Real-Time and Embedded Technology and Applications Symposium, 2002. Proceedings. Eighth IEEE*, 2002, pp. 197-206.

[۷] G. Cena, I. C. Bertolotti, T. Hu and A. Valenzano "A mechanism to prevent stuff bits in CAN for achieving jitterless communication", *IEEE Trans. Ind. Informat.*, vol. 11, no. 1, pp.83 -93 2015.

[۸] Cena, G.; Cibrario Bertolotti, I.; Tingting Hu; Valenzano, A. "Fixed-Length Payload Encoding for Low-Jitter Controller Area Network Communication", *Industrial Informatics, IEEE Transactions on*, On page(s): 2155 - 2164 Volume: 9, Issue: 4, Nov. 2013.

[۹] Cena, G.; Bertolotti, I.C.; Tingting Hu; Valenzano, A. "Performance comparison of mechanisms to reduce bit stuffing jitters in controller area networks", *Emerging Technologies & Factory Automation (ETFA), 2012 IEEE 17th Conference on*, On page(s): 1 - 8.

[۱۰] T. Nolte ,H. Hansson, C. Norström, and S. Punnekkat, "Using bit-stuffing distributions in CAN analysis," in *IEEE Real-Time Embedded Systems Workshop at the Real-Time Systems Symposium*, 2001.

[۱۱] R. Bosch, "CAN specification version 2.0," *Rober Bousch GmbH, Postfach*, vol. 300240, 1991.

[۱۲] N. Çenesİz and M. Esin, "Controller area network (CAN) for computer integrated manufacturing systems," *Journal of Intelligent Manufacturing*, vol. 15, pp. 481-489, 2004.

[۱۳] R. I. Davis, A. Burns, R. J. Bril, and J. J. Lukkien, "Controller Area Network (CAN) schedulability analysis: Refuted ,revisited and revised," *Real-Time Systems,* vol. 35, pp. 239-272, 2007.

[۱۴] K. Tindell and A. Burns, "Guaranteeing message latencies on control area network (CAN)," in *Proceedings of the 1st International CAN Conference*, 1994.

[۱۵] K. Tindell, A. Burns, and A. J. Wellings, "Calculating controller area network (CAN) message response times," *Control Engineering Practice,* vol. 3, pp. 1163-1169, 1995.