Journal of Engineering Sciences and Information Technology
Volume (5), Issue (5) : 30 Dec 2021
P: 1 - 29

ISSN: 2522- 3321

مجلة العلوم الهندسية وتكنولوجيا المعلومات
المجلد (5)، العدد (5) : 30 ديسمبر 2021م
ص: 1 - 29

Designing a Virtual Platform for Modeling Nodes in Wireless Sensor Networks at the Central Processing Unit Level

Ghaidaa Mohammad Esber

Mothanna Alkubeily

Faculty of Mechanical and Electrical Engineering || Tishreen University || Syria

Samer Sulaiman*

Faculty of Mechanical and Electrical Engineering || Manara University || Syria

Abstract: Wireless sensor network simulation programs provide representation for an actual system, without needing to deploy real testbed which is highly constrained by the available budget, and the direct operations inside physical layer in most of these programs are hidden and work implicitly. This is what motivated us to build a kernel for a virtual simulation platform to be able to simulate protocol operations and algorithms at the node processing unit level, The proposed system aims to observe the execution of operations at the low level of the wireless sensor physical infrastructure with the ability to modify at this level. That give the improvers of wireless sensor nodes the ability to test their ideas without needing to use physical environment. We have built the functionality operations which are related to the platform kernel at several stages. We defined (as a first step) the essential operations inside a virtual microprocessor that uses a partial set pf MIPS instructions, and built the kernel of minimized virtual WSN simulator depending on the proposed microprocessor, that means we can add any number of nodes inside the GUI of the WSN simulator kernel, and these nodes use the proposed virtual microprocessor . Then we improved this platform by adding the instruction set of a real microprocessor that is used in wireless sensor network nodes. Finally, (and to ease and simplify the interaction operation between program GUI of the platform kernel and the user), we have built simplified compiler that allows user to deal with microprocessor GUI inside each node, and to clarify protocol and algorithm operations by a set of orders and functions without needing to deal with low level language (Assembly language) in a direct way. The simulation results have presented high flexibility and performance to this platform in observing the operation sequence inside wireless sensor nodes at assembly level, in addition to focus on some parameters that are related to microprocessor inside each node.

Keywords: Authentication, Compiler, Microprocessor, Sensor node, Simulation, Virtual Platform, Wireless Sensor Network (WSN).

تصميم منصة برمجية افتراضية لنمذجة العقد ضمن شبكات الحساسات اللاسلكية على مستوى وحدة المعالجة المركزية

غيداء محمد اسبر

*samer.sulaiman@manara.edu.sy

مثنى القبيلي
كلية الهندسة الميكانيكية والكهربائية || جامعة تشرين || سوريا
سامر سليمان
كلية الهندسة الميكانيكية والكهربائية || جامعة المنارة || سوريا

المستخلص: تقدم برامج المحاكاة المستخدمة ضمن مجال شبكات الحساسات اللاسلكية Wireless Sensor Networks (WSN) تمثيلاً عن النظام الحقيقي دون الحاجة للقيام بعملية نشر فعلية للعقد وما يترافق مع ذلك من تكاليفٍ باهظة، وتكون العمليات المباشرة المعرفة ضمن الطبقة الفيزيائية في معظم هذه البرامج ضمنية وغير مقدمة بشكلٍ واضح، وهذا ما دفعنا إلى بناء نواة لنظام منصة محاكاة افتراضية، لنكون بذلك قادرين على محاكاة البروتوكولات والخوارزميات المختلفة المطبقة ضمن شبكات الحساسات على مستوى وحدة المعالجة المركزية. تهدف منصة المحاكاة المقترحة إلى مراقبة تنفيذ العمليات على المستوى المنخفض للبنية الفيزيائية لعقد الحساسات مع القدرة على التعديل عند هذا المستوى، الأمر الذي يتيح لمطوري عقد الحساسات اللاسلكية اختبار أفكارهم دون الحاجة إلى بيئة عمل فيزيائية، وقد قمنا ببناء العمليات الوظيفية المتعلقة بنواة هذه المنصة على عدة مراحل، حيث قمنا بدايةً بتعريف وإضافة مجموعة العمليات المتعلقة بمعالج صغري افتراضي يستخدم مجموعة جزئية من تعليمات معالجMIPS، وبناء النواة المتعلقة بمحاكي WSN الافتراضية المصغرة اعتماداً على المعالج المقترح، وهذا يعني أنه بإمكاننا إضافة أي عدد من العقد ضمن واجهة المحاكي، والتي تستخدم المعالج الافتراضي المقترح. ثمَّ قمنا بتطوير هذه المنصة من خلال إضافة مجموعة العمليات المتعلقة بمعالج فعلي مستخدم ضمن عقد الحساسات، وأخيراً (وبهدف تسهيل وتبسيط عملية التفاعل بين واجهة البرنامج الرئيسية لنواة المنصة المقترحة والمستخدم) فقد بنينا مترجم Compiler مبسط يتيح للمستخدم التعامل مع واجهة المعالج المعرف ضمن كل عقدة من خلال مجموعة من الأوامر دون وجود الحاجة للتعامل بشكل مباشر مع لغة المستوى المنخفض (Assembly Language). وقد أظهرت النتائج مرونة وفعالية عالية لهذه المنصة المصممة في تتبع سير العمليات المنجزة ضمن عقد الحساسات اللاسلكية على مستوى لغة التجميع، إضافة إلى التركيز على بعض البارامترات المتعلقة بالمعالج الصغرى ضمن العقدة.

الكلمات المفتاحية: المصادقة (التوثيق)، المترجِم، المعالج الصغرى، عقدة الحساس، المحاكاة، المنصة الافتراضية، شبكة الحساسات اللاسلكية.

المقدمة.

تتألف شبكات الحساسات اللاسلكية Wireless Sensor Networks (WSN) من عدد من أجهزة الاستشعار صغيرة الحجم، ذاتية التغذية، تدعى عقد حساسة Sensor nodes، وتستخدم لمراقبة ظاهرة فيزيائية أو كيميائية محددة في الوسط المحيط، ويمكن أن تزوَّد بتجهيزات إضافية خاصة كالكاميرات والمايكروفونات لتصبح قادرة على التعامل مع الوسائط المتعددة. تقوم هذه الأجهزة بنقل المعلومات عن هذه الظاهرة لاسلكياً إلى المحطة القاعدية (المركز) Base station (Sink) للاستفادة منها، دون الحاجة لتواجد الإنسان في مكان الظاهرة المدروسة، وتستخدم هذه الشبكات ضمن مجال واسع من التطبيقات بدءاً من تطبيقات المراقبة والإشراف العسكرية وصولاً إلى مراقبة المرضى وتحسس الظواهر الطبيعية [1]، وتواجه عقد الحساسات المستخدمة ضمن هذه الشبكات الكثير من التحديات مثل محدودية الطاقة، محدودية عرض الحزمة، متطلبات جودة الخدمة، مقدرات المعالجة المحدودة، إضافة إلى تحديات الأمن [2].

تتم عملية نشر عقد الحساسات ضمن بيئة اختبار فعلية بهدف الحصول على نتائج دقيقة، لكنَّ ذلك يفرض وجود عدد من المعوقات التي تبدأ من الكلفة الباهظة، حيث أنهُّ من الصعب شراء عد كبير من عقد الحساسات عند الحاجة لإجراء اختبار في بيئة ممتدة على مساحة واسعة، ولاسيما عند إجراء الأبحاث العلمية الأكاديمية، إضافةً إلى عدم القدرة على تكرار الاختبار ضمن بعض البيئات ذات الظروف الخاصة كما هو الحال عند مراقبة البراكين، وهذا ما دعا لاستخدام نمذجة النظم ومحاكاتها والتي تعتمد على صياغة نموذج مبسَّط من النظام

الحقيقي، حيث أنَّها تسمح للمستخدمين بأن يراقبوا النظام عن قرب دون الحاجة لإجراء تنفيذ فعلي له، وخلال عملية الاختبار يتم تطبيق بارامترات مختلفة لدراسة سلوك النظام وبناء على ذلك يقرر المستثمرين فيما إذا كان النظام الحالي مناسباً أو بحاجةٍ للمزيد من التحسين، وقد ركزت برامج المحاكاة الخاصة بشبكات الحساسات اللاسلكية على طبقات الشبكة المختلفة[3]، لكنها لم تقدم معلومات تفصيلية عن عمليات المعالجة ضمن معالج العقدة، حيث أنها تعتمد على قيم بارامترات معرفة مسبقاً من معالج العقدة وفقاً للخصائص المتعلقة بنوع المعالج المعرَّف ضمنَ العقدة، وتقدم نتائج المحاكاة دون إظهار تفاصيل عمليات المعالجة أو على الأقل لا تسمح بمراقبة تسلسل عمليات التنفيذ ضمن معالج العقدة، وهذا ما دفعنا لبناء نواة لنظام منصة محاكاة برمجية افتراضية خطوة- خطوة وذلك بهدف محاكاة الخوارزميات والبروتوكولات على مستوى لغة التجميع لنتمكن بذلك من أن نصف ونراقب بوضوح مجموعة العمليات المستخدمة لتنفيذ البروتوكولات ضمن معالج العقدة، مع القدرة على التعديل والتصحيح عند هذا المستوى.

وسنستعرض في هذا البحث الخصائص المتعلقة بنواة منصة المحاكاة المقترحة والمواصفات الأساسية لعمليات المعالجات الصغرية التي يمكن استخدامها ضمن عقد الحساسات، إضافة إلى تقديم نتائج المحاكاة من خلال تطبيق بعض الأمثلة وذلك بهدف توضيح آلية العمل ضمن البيئة المصممة مثل إرسال بعض القيم بين العقد. وباعتبار أن أمن التوجيه يشكل تحدياً أساسيا ضمن هذا النوع من الشبكات حيث أن وسط الاتصال المستخدم هو وسط لاسلكي، مما يسمح لأي عقدة خارجية تقع ضمن مجال الإرسال بالوصول إلى المعطيات المتبادلة والحصول على تلك المعطيات، لذا سنقوم بتطبيق إحدى خوارزميات أمن التوجيه ضمن البيئة الافتراضية المصممة، وهي خوارزمية المصادقة بين عقدتين مستقلتين من عقد الحساسات بهدف مراقبة تنفيذ عملياتها على المستوى المنخفض، مع توضيح عمليات المترجم Compiler المخصص لإدخال الأوامر بلغة عالية المستوى وآلية تحويلها للغة التجميع ضمن عقد الشبكة.

مشكلة الدراسة:

تركز برامج المحاكاة المستخدمة ضمن مجال شبكات الحساسات اللاسلكية على طبقات الشبكة المختلفة، لكنها لا تقدم معلومات تفصيلية عن عمليات المعالجة ضمن معالج العقدة، حيث تكون أغلب العمليات المعرفة على المستوى المنخفض (مستوى لغة التجميع) مغلَّفة دون وجود إمكانية لتطبيق خاصية اكتشاف الأخطاء وتصحيحها (Debugging)، وهذا ما يقلل من إمكانية إجراء التحسينات بالشكل المرغوب نظراً لعدم القدرة على رؤية تفاصيل العمليات من جهة، وبقاء القيم المتعلقة بمواصفات وأداء المعالج قيماً تقريبية لا تعبر عن العمليات ضمن الشبكة بشكلٍ دقيق من جهةٍ أخرى.

ويمكن صياغة مشكلة الدراسة في التساؤلات التالية:

1- هل تتيح برامج المحاكاة المتوفرة مراقبة العمليات وتتبعها على مستوى المعالج الصغرى ضمن عقد الحساس؟
2- هل يوجد إمكانية ضمن هذه البرامج لإجراء تعديلات ضمن هذا المستوى مثل اختيار نوع المعالج الصغرى المستخدم ضمن العقدة، ودراسة أثر ذلك على الشبكة كلياً دون الاعتماد على قيم جاهزة ومعرفة مسبقاً؟
3- هل يعتبر تعديل البنى التحتية المتعلقة ببرامج المحاكاة الشهيرة لكي تظهر عمليات المستوى المنخفض وما يترافق مع ذلك من ضرورة الفهم العميق لعملية تصميم وبرمجة هذه المحاكيات، أمراً متاحاً بسهولة؟

وهذا ما دفعنا نحو التوجه لتصميم نواة متعلقة بمنصة محاكاة برمجية افتراضية لنتمكن بذلك من أن نصف ونراقب بوضوح تسلسل تنفيذ العمليات المتعلقة بالبروتوكولات والخوارزميات ضمن معالج العقدة مع القدرة على التعديل والتصحيح عند هذا المستوى.

فرضيات الدراسة: تفترض الدراسة مجموعة من النقاط نذكر منها:

1- بناء النواة المتعلقة بمنصة محاكاة برمجية افتراضية وذلك على مستوى تعليمات المعالج الصغرى، بما يساهم في التقليل من الاعتماد على البيئة المادية لعقد الحساسات اللاسلكية، ويعطي صورة أوضح عن آلية العمل ضمن عقد الحساسات.

2- يتم التركيز بشكلٍ أساسي على محاكاة تفاصيل العمليات المتعلقة بمجموعة تعليمات المعالج الصغرى (Microprocessor Instruction Set) ضمن العقدة دون الأخذ بالحسبان بنية وتركيب الدارات المستخدمة ببناء المعالج الصغرى كعتاد صلب (Hardware).

3- سنأخذ بالحسبان إمكانية تعامل عقد الحساسات التي ستضاف إلى واجهة محاكي WSN مع نوعين من عمليات المعالجات المعرفة ضمن بيئتنا، وهي معالج صغري افتراضي، ومعالج من عائلة MSP430 series، وهذا يعني أنه بإمكاننا إضافة العدد المطلوب من العقد ضمن واجهة المحاكي، ويمكن لهذه العقد أن تستخدم تعليمات أحد المعالجين المذكورين.

أهداف الدراسة:

تهدف هذه الدراسة إلى استعراض العمليات المتعلقة بنواة المنصَّة الافتراضية، والتي تمَّ تصميمها لمحاكاة عقد الحساسات ضمن شبكات الحساسات اللاسلكية على المستوى المنخفض (مستوى لغة التجميع) مع القدرة على التعديل والتصحيح عند هذا المستوى. وقد قمنا باختبار البيئة المقترحة من خلال تطبيق إحدى الخوارزميات أمن التوجيه، وذلك لتوضيح سير العمليات ضمن البيئة المقترحة اعتماداً على المعالجات الصغرية المعرَّفة، والمترجم المبسَّط Simplified Compiler.

أهمية الدراسة:

تنبع الأهمية العلمية للدراسة من خلال التركيز على مراقبة تنفيذ الخوارزميات والبروتوكولات المستخدمة ضمن شبكات الحساسات اللاسلكية وتحسينها، وذلك عبر تصميم نواة لمنصة محاكاة برمجية افتراضية، والتي تأخذ بالحسبان عمليات المستوى المنخفض(عمليات المعالج الصغرى) مع وجود إمكانية المراقبة والتعديل على هذا المستوى، الأمر الذي لا يكون متاحاً لدى معظم برامج المحاكاة، والتي تهمل البارامترات المتعلقة بالمعالج وتعتمد على قيم معرفة مسبقاً وفقاً للخصائص المتعلقة بنوع المعالج المعرَّف ضمنَ العقدة، وتقدم النتائج دون إظهار تفاصيل عمليات المعالجة.

منهجية الدراسة.

سنعتمد ضمن طرائق البحث على الجانب التحليلي من خلال دراسة وتحليل مجموعة العمليات المتعلقة ببناء نواة منصة المحاكاة الافتراضية، وعمليات المعالجات الصغرية المعرفة ضمن عقد الحساسات (مثل مجموعة التعليمات المستخدمة، الذواكر، المسجلات، ...)، إضافةً إلى تحليل عمل إحدى الخوارزميات المراد اختبار عملياتها ضمن البيئة المصمَّمة، وهي خوارزمية المصادقة بين عقدتين مستقلتين من عقد الحساسات واستعراض الخطوات

اللازمة لتنفيذها، كما سنعتمد على الجانب التجريبي، وذلك من خلال تطبيق هذه الخطوات ضمن بيئة منصة المحاكاة الافتراضية المقترحة، ومراقبة تسلسل تنفيذ العمليات ضمنها.

وقد استخدمنا لبناء نواة البيئة المصمَّمة، وتعليمات الخوارزمية التي سيتم تطبيقها ضمن المعالج المقترح لغة (Microsoft Visual Studio C# 2015 language) #Cوالتي تعد لغة مرنة عند بناء تطبيقات تصميم الواجهات[4] (Graphical User Interface (GUI، وهذا ما سيعطينا سهولةً كبيرة ومجالاً واسعاً من خياراتِ التطوير لكامل منصة المحاكاة المقترحة في المستقبل.

هيكلة الدراسة:

تم تقسيم هذه الدراسة إلى ثلاثة مباحث، يتناول المبحث الأول هيكلية بعض برامج المحاكاة المستخدمة ضمن مجال شبكات الحساسات اللاسلكية، إضافة إلى بعض البرامج المستخدمة لمحاكاة المعالجات الصغرية، بينما يتطرق المبحث الثاني إلى نواة منصة المحاكاة البرمجية الافتراضية المصمَّمة، وخصائص المحاكي المقترح والمعالجات الصغرية المعرَّفة ضمنه، ويستعرض المبحث الأخير الخطوات المتعلقة بإحدى الخوارزميات المستخدمة لتحقيق أمن التوجيه ضمن شبكات الحساسات اللاسلكية، والمراد تطبيق عملياتها ضمن البيئة المقترَحة بهدف التركيز على آلية العمل وتتبع سير العمليات ضمنها.

المبحث الأول- الإطار النظري والدراسات السابقة

أولاً- الإطار النظري لبرامج المحاكاة في مجال شبكات الحساسات اللاسلكية والمعالجات الصغرية

ركزت برامج المحاكاة الخاصة بشبكات الحساسات اللاسلكية WSN على طبقات الشبكة المختلفة، لكنها لم تقدم معلومات تفصيلية عن عمليات المعالجة ضمن معالج العقدة، حيث أنها تعتمد على قيم بارامترات معرفة مسبقاً من معالج العقدة وفقاً للخصائص المتعلقة بنوع المعالج المعرَّف ضمنَ العقدة.

من جهةٍ أخرى، أظهرت بعض البرامج المستخدمة لمحاكاة المعالجات الصغرية المعلومات المتعلقة بعمليات المعالج الصغرى، لكنها لم تدعم البيئة المعبرة عن شبكات الحساسات اللاسلكية WSN، وسنستعرض فيما يلي البرامج الشائعة المستخدمة في مجال شبكات الحساسات اللاسلكية WSN وفي مجال المعالجات الصغرية مع إظهار الميزات الأساسية للمنصّة البرمجية المصمَّمة مقارنة معها.

ثانياً- الدراسات السابقة

1- محاكي NETTOPO [5]، يشكل هذا البرنامج إطاراً متكاملاً لعمليات النمذجة والمحاكاة، ويساعد على اختبار خوارزميات التّوجيه ضمن شبكات WSN، كما يسمح بنمذجة شبكات هجينة ذات حجم كبير، ومع الأخذ بالحسبان نماذج المحاكاة، فإنه يمكن للمستخدم أن يعرف بسهولة عدداً كبيراً من البارامترات الأساسية المطلوبة والتي تتعلق بعقد الحساسات مثل الطاقة الابتدائية، عرض حزمة، ونصف قطر الإرسال، ويمكن للمستخدم أن يعرف سلوك المعالجة الداخلي المتعلق بعقدة الحساس مثل استهلاك الطاقة وإدارة عرض الحزمة.

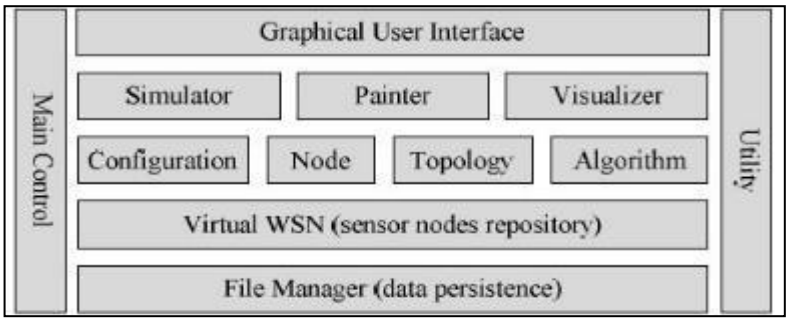
يعتمد لغة الجافا ويعمل على نظم التشغيل المختلفة، كما يتمتع هذا البرنامج بالعديد من الميزات كالمرونة من حيث إمكانية نشر العقد بشكل منفرد حيث يمكن للمستخدم أن يختار بشكل يدوي نشر العقدة ضمن موقع محدد وهذا يفيد بتحديد موقع العقدة الهدف أو العقدة المرسلة، إضافة لإمكانية تحريك العقد بشكلٍ منفرد حيث

يحرك أي عقدة لأي مكان ضمن حقل الـ WSN بعد نشر الشبكة وهذا ما يفيد بتحديث مواقع عقد معينة مثل تحريك العقدة الهدف من أجل خيارات متعددة، وإمكانية نشر العقد بشكل عشوائي حيث يمكن للمستخدم أن ينشر بشكل عشوائي عدداً محدداً من عقد الحساسات.

وتتوفر في هذا البرنامج خيارات المعالجة للعقدة مثل حذف العقدة، إظهار الخصائص الحالية للعقدة، تعديل القيم ضمن العقدة قبل إجراء المحاكاة، البحث عن العقدة من خلال الهوية المتعلقة بها، وعدم تنشيط بعض العقد ضمن مساحة محددة لتشكل بذلك فجوات ديناميكية Dynamic Holes.

إضافةً إلى ذلك، تتيح بيئة البرنامج تسجيل نتائج المحاكاة مما يسهل عملية التعامل معها. يبين الشكل (1) الهيكلية الأساسية المتعلقة ببرنامج NETTOPO.

يمكن تلخيص سيئة هذا البرنامج تجاه بيئتنا المقترحة بتغليف كامل العمليات، حيث ينخفض مستوى العمل ليصل لمستوى العقدة Node Level دون وجود إمكانية لمراقبة تفاصيل العمليات ضمن المسجلات والذواكر، ونفقد بذلك القدرة على اكتشاف الأخطاء وتصحيحها (Debugging)، وما يندرج تحت ذلك من كشف موضع الخلل وإجراء التحسين.

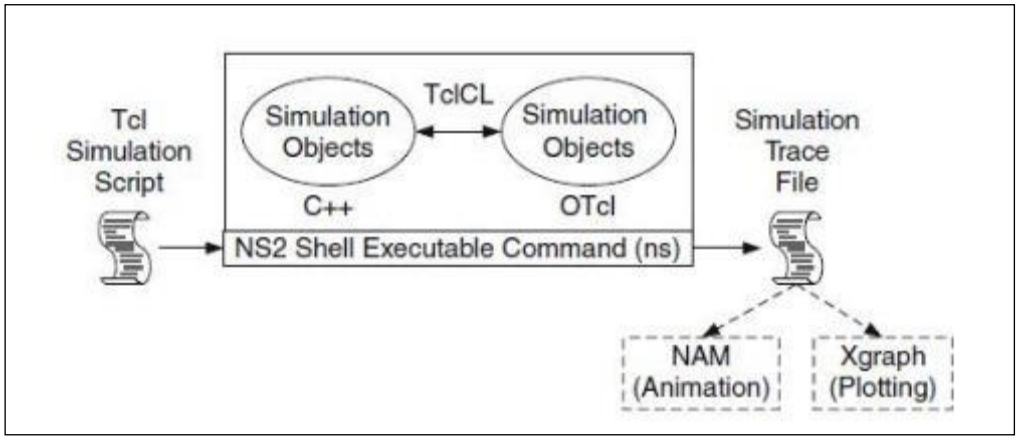


الشكل (1) الهيكلية الأساسية لبرنامج NETTOPO

2- محاكي NS2 [6]، تم إثبات فاعلية برنامج المحاكاة NS2 في دراسة الطبيعة الديناميكية لشبكات الاتصال حيث يتم من خلاله محاكاة البروتوكولات والتوابع ضمن الشبكات السلكية بالإضافة للشبكات اللاسلكية مثل (خوارزميات التوجيه، TCP، UDP)، ويمكن ببساطة تعريف هذا البرنامج كأداة محاكاة تعتمد مبدأ الأحداث المتقطعة Discrete Event Simulatorضمن أبحاث الشبكات. نعرف ضمن NS2 لغتين أساسيتين هما: C++ ولغة أوامر الأدوات غرضية التوجيه (OTCL) Object- Oriented Simulator، ويعتمد بشكلٍ أساسي على نظام Linux.

وفي عام 2006 قام فريق من الباحثين مدعوماً من قبل National Science Foundation(NSF) لبناء بديل عن NS2 يدعى NS3 والذي يقسم الشبكة إلى أربع أقسام وهي العقدة، أجهزة الشبكة، القنوات، والتطبيقات وميزته الأساسية هي إمكانية كتابة الـ Scriptعبر لغة C++ أو لغة Python، ويبين الشكل (2) الهيكلية الأساسية ضمن برنامج NS2.

تتلخص سيئة بيئة هذا المحاكي بأنها تعمل على مستوى الرزمة Packet Level كأدنى مستوى ممكن، وبالتالي فإنَّ العمليات تكون مغلفة مما يقلِّل من مجال الاختبار وإجراء التعديلات مقارنة مع بيئتنا المفترضة، والتي تنخفض بالعمليات إلى مستوى تعليمات المعالج الصغرى.

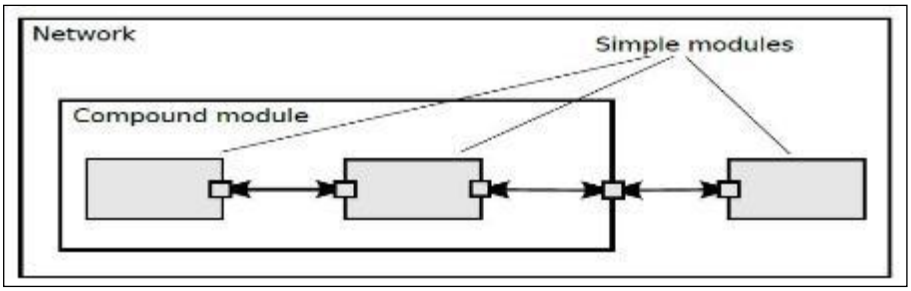


الشكل (2) الهيكلية الأساسية لبرنامج NS2

3- محاكي OMNET++ [7]، يشكل برنامج OMNET++ إطاراً لمحاكي شبكات غرضي التوجه، ويعتمد مبدأ الأحداث المتقطعة Discrete Event Principle، ويمتلك هذا الإطار هرمية شاملة ونوعية لذا فهو يستخدم في العديد من المجالات مثل نمذجة شبكات الاتصالات السلكية واللاسلكية، نمذجة البروتوكولات، جوانب تقييم الأداء المتعلقة بالنظم البرمجية المعقدة وغير ذلك من الجوانب، كما أنه يدعم شبكات الحساسات اللاسلكية باستخدام إطار INET الموجود ضمن النسخ الأخيرة منه.

تتميز الهيكلية المتعلقة ببرنامج OMNET++ بوجود نماذج هرمية حيث يتألف كل نموذج من عدة وحدات متداخلة هرمياً وتتصل مع بعضها البعض بتبادل الرسائل فيما بينها، حيث يعبر المستوى الأعلى ضمن هذه الهيكلية عن نموذج النظام System Module والذي يحوي مجموعة من الوحدات الجزئية Compound Modules، كما تحوي هذه الوحدات بدورها على وحدات جزئية أصغر Simple Modules (والتي تتعامل مع خوارزميات النموذج) كما هو مبين بالشكل (3). يتم وصف التركيب المتعلق بنموذج المحاكاة ضمن برنامج OMNET++ باستخدام لغة وصف الشبكة Network Description Language والتي يشار لها اختصاراً بـOMNET++ NED Language، حيث يتم عبر هذه اللغة التصريح عن النماذج البسيطة وتعريفها، إضافةً إلى كيفية اتصالها مع بعضها البعض وتجميعها ضمن النماذج المركبة، بينما يتم تنفيذ هذه النماذج البسيطة باستخدام لغة C++وذلك عبر مكتبة الأصناف الخاصة بالمحاكاة ضمن برنامج OMNET++.

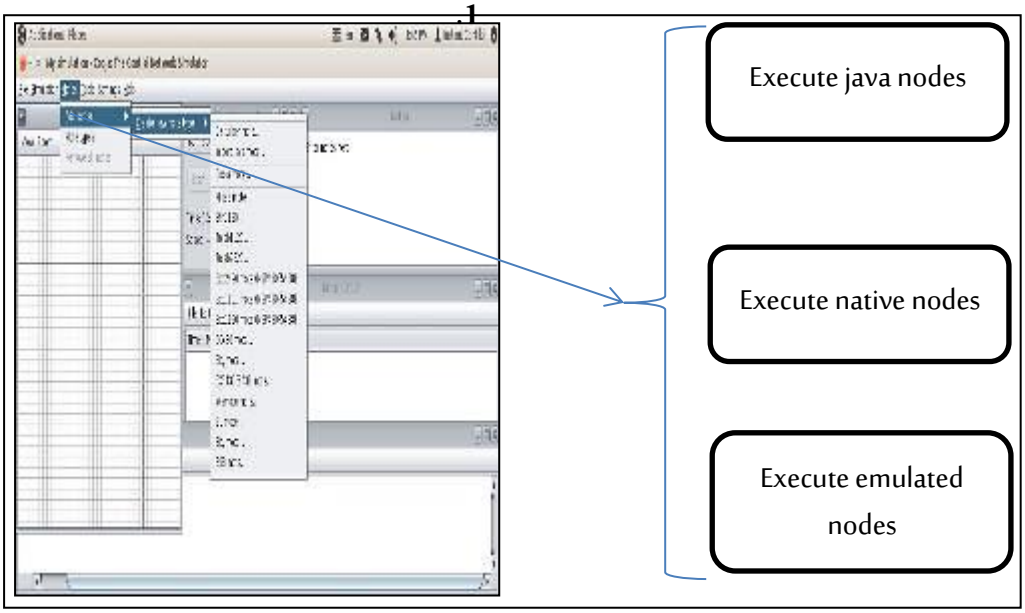
يتعامل المستخدم ضمن برنامج OMNET++ مع النماذج البسيطة والخوارزميات المعرفة ضمنها باستخدام لغة C++ دون وجود أي إظهار لتفاصيل عمليات المعالج الصغرى ضمن العقدة، وبالتالي عدم وجود إمكانية الاختبار والتحسين على المستوى المنخفض مقارنةً مع البيئة المصمَّمة.



الشكل (3) هيكلية برنامج OMNET++

4- محاكيCOOJA [8]، تم تصميم هذا المحاكي بشكلٍ أساسي لمحاكاة شبكات الحساسات التي تعمل على نظام التشغيل Contiki، ويعتمد لغة الجافا، ويمكن لكل عقدة من عقد الحساسات الموجودة في الشبكة المعرفة ضمن بيئة هذا البرنامج أن تكون من أنواعٍ مختلفة، حيث لا يكون الاختلاف بالبرمجيات المتعلقة بالعقدة فحسب، بل بالدارات المشكلة للعقدة المراد محاكاتها (الهاردوير ضمن العقدة) أيضاً. ويتميز برنامج COOJA بوجود مرونة كبيرة في عدة أجزاء من المحاكي، حيث يمكن بسهولة استبدال بعض الخصائص أو إضافة خصائص ووظائف أخرى. وتتميز العقدة المراد محاكاتها بوجود ثلاث خصائص أساسية وهي ذاكرة البيانات المتعلقة بها، نوع العقدة، والطرفيات المتعلقة بالدارات المشكلة لها (الهاردوير).

يمكن أن تتم مشاركة نوع العقدة بين عدة عقد، مما يحدِّد الخصائص العامة لهذه العقد، فعلى سبيل المثال، العقد من نفس النوع تشغل نفس الكود البرمجي المتعلق بالبرنامج Program Code على نفس الطرفيات المتعلقة بالدارات التي يتم محاكاتها، كما تمتلك العقد من نفس النوع نفس ذاكرة البيانات Data Memory باستثناء المحدد المتعلق بالعقدة Node ID، وخلال التنفيذ ستختلف ذواكر البيانات للعقد بعد التفاعل مع البيئة الخارجية. ويمكن ملاحظة العناصر الأساسية في برنامج COOJAمثل نوع العقدة والتي تظهر ضمن الشكل (4).



الشكل (4) أنواع العقد ضمن محاكي COOJA

يمكن ضمن برنامج COOJA أن تظهر تفاصيل العمليات أثناء تشغيل الشبكة بما في ذلك إظهار تنفيذ التعليمات على المستوى المنخفض وفقاً لنوع المعالج المعرف ضمن العقدة، لكن دون وجود أي إمكانية للقيام بتعديل هذه التعليمات أو إضافة أنواع أخرى من المعالجات واختبار أثر تغيير المعالج على الشبكة كلياً [9].

5- محاكي OPNET [10]، يشكل برنامج OPNET بيئة برمجية ضخمة وذات فاعلية كبيرة مع وجود مجال واسع من خيارات الاختبار والتطبيق ضمنها، حيث يمكن ضمن هذه البيئة محاكاة كامل الشبكات الهجينة وذلك مع العديد من البروتوكولات، وقد صُمم هذا البرنامج بشكلٍ أساسي لمحاكاة الشبكات الثابتة إذ تحوي مكتبة OPNET مجالاً واسعاً من النماذج الدقيقة للبروتوكولات والمعدات Hardware المتعلقة بالشبكات التجارية المتوفرة، إضافةً لمحاكاة الشبكات اللاسلكية.

تعمل المحاكاة على مستوى الرزم Packet Level، كما تقسَّم البنية الهرمية للنمذجة ضمن بيئة OPNET إلى ثلاث مستويات وهي مستوى الشبكة، مستوى العقدة، ومستوى المعالجة والذي يتم ضمنه محاكاة نموذج منفرد، وإجراء ترميز العقدة كنموذج مصدر لحركية البيانات[11] Data Traffic Source Model.

من أهم السلبيات ضمن بيئة OPNET مقارنة مع البيئة المقترحة بالإضافة إلى كونها تجارية، فإن هرمية النمذجة تنخفض كما ذكرنا مسبقاً، لكن دون الوصول لمستوى تعليمات المعالج الصغرى وبالتالي عدم وجود إمكانية التحسين الممكنة عند التعامل مع تفاصيل العمليات ضمن هذا المستوى.

6- محاكي MATLAB [12]، تمَّ تصميم برنامج MATLAB بهدف الوصول السهل لبرمجيات المصفوفات المطورة من قبل مشاريع EISPACK وLINPACK والتي تمثل معاً أساساً للبرمجيات المتعلقة بحسابات المصفوفات.

يشكل برنامج MATLAB بيئة مناسبة للحسابات العددية المتعددة النماذج، كما يمثل لغة برمجية عالية المستوى من أجل الحسابات والعمليات التقنية وقد تم تطويرها من قبل الـ Mathworks، فهو يكامل بين عمليات الحساب والنمذجة والمحاكاة إضافة للبرمجة بحيث يكون من السهل استخدام المحيط.

يعد هذا البرنامج بيئةً تفاعلية عناصر بياناتها الأساسية هي المصفوفات، وهذا ما يسمح بحل الكثير من مشاكل العمليات الحسابية والتقنية وخصوصاً تلك التي تتعلق بالمصفوفات والأشعة.

يسمح برنامج الماتلاب بكثيرٍ من التطبيقات نذكر منها:

- العمليات الرياضية والحسابية.
- معالجة المصفوفات وإظهار التوابع والبيانات وعمليات تحليل البيانات.
- تنفيذ الخوارزميات وتطويرها.
- تصميم وبناء الواجهات للمستخدم GUI.
- عمليات النمذجة والمحاكاة والمخططات الهندسية والعلمية
- التفاعل مع البرامج المكتوبة بلغات أخرى متضمنة C,C++,C#,Fortron وPython.

يتيح برنامج MATLAB التعامل مع الواجهات، لكنَّ الأدوات المتاحة ضمنه محدودة مقارنة مع #C (اللغة المستخدمة لتصميم بيئتنا) حيث تتيح لغة #C أدوات أكثر مناسبة لتصميم كل أنواع الواجهات، وتتطلب برمجة هذه الأدوات عدداً أقل من التعليمات مقارنة مع ماتلاب، كما يوجد مرونة أكبر عند تبادل المعلومات بين النوافذ المتعددة.

7- MPLAB [13]، يشكل برنامج MPLABبيئة برمجية تعمل على الحاسوب كي تؤمن محيطاً تطويرياً لتصميم المتحكم بشكل موسَّع، وتتيح بيئة هذا البرنامج اختيار أحد أنواع المتحكمات المدعومة من شركة Microchip بشكل أساسي بالإضافة لأنواع من المتحكمات التابعة لشركاتٍ عدة.

ونعرف ضمن هذا البرنامج واجهة محرر نصوص لكتابة التعليمات، حيث يمكن التعبير عن القيم وإظهارها باستخدام الترميز السِّتَّ عشري Hexadecimal Encodingوالذي يعبر عن ترميز العمليات على المستوى المنخفض، كما تتم الكتابة من خلال لغة التجميع Assemly Language الخاصة بالمتحكم الذي يتم اختياره، ويتيح هذا البرنامج تحديد موضع بداية كل قطاع تعليمات، ويمكن أن يقوم بإجراء محاكاة افتراضية بحيث يظهر كافة المسجلات ومواقع الذاكرة وأثر كل تعليمة خطوة بخطوة على هذه المواقع والمسجلات وعلى بوابات الدخل والخرج للمتحكم [14].

بالمقارنة مع الواجهة المصممة فإنَّ برنامج MPLAB لا يتيح إمكانية نمذجة عمليات معالجات عدة عقد تعمل معاً وبشكلٍ متزامن(كما هو الحال في شبكات الحساسات اللاسلكية) بل يتم التعامل مع معالج صغري واحد.

8- PROTEUS [15]، يعتبر برنامج PROTEUS شائع الاستخدام ويحوي مكتبة من العناصر الالكترونية والمتحكمات الصغرية، وتحوي هذه المكتبة نماذج تحاكي الخصائص الالكترونية لهذه العناصر بدقة عالية.

نعرف ضمن بيئة هذا البرنامج واجهة عمل تتيح إضافة أي عدد من العناصر الموجودة ضمن المكتبة الخاصة بالبرنامج ووضعها على الواجهة كرموز رسومية مع إمكانية الوصل بين نقاطها رسومياً لتشكيل أي دارة الكترونية كمخطط دارة، كما يتيح تغيير قيم العناصر على المخطط وفقاً لرغبتنا وإجراء المحاكاة بهدف استنتاج شكل إشارات الجهد والتيار عند كل نقطة من نقاط هذا المخطط. لكن وعند استخدام نموذج لأحد المتحكمات في المخطط فعلينا أن نستخدم MPLAB أو أحد البرامج المستخدمة لبرمجة المتحكمات، وذلك لكتابة البرنامج الذي سيشغل هذا المتحكم فنحفظه على شكل ملف بترميز الـ Hexadecimalثم نضعه في واجهة PROTEUS، حيث نقوم بدمجه مع المتحكم عن طريق أوامر خاصة.

يمكننا في هذا البرنامج إضافة مجموعة متحكمات بنفس الوقت على مخطط الدارة وتشكيل وصل داراتي بينها، كما يمكن تصميم وحدة اتصال وربطها مع كل متحكم وربط وحدات الاتصال بالشبكة، ولكن يعتبر هذا الأمر معقد جداً لأنه يتطلب تصميم الكتروني لكل أنواع القطع في عقدة الحساس بالإضافة إلى تشابك الخطوط الهائل في المخطط [16].

تختلف بيئة برنامج PROTEUS عن البيئة المقترحة بأمرين أساسيين، الأول هو أنَّ عملية إضافة عدد كبير من العقد تعتبر معقدة جداً، حيث يستحيل عملياً الوصل بين العقد إلا إذا تم تصميم دارات وحدات الاتصال بين العقد إلكترونياً بشكل كامل لتكون لاسلكية، والثاني هو أنه لا يمكن رؤية تفاصيل ما يحصل ضمن المتحكمات وبالتالي نفقد خاصية إمكانية رؤية تفاصيل العمليات واكتشاف الأخطاء (Debugging)، والتي تعتبر ميزة رئيسية ضمن برنامجنا المقترح، حيث لا يمكن إظهار الإشارات المستقبلة إلا كواحدات وأصفار بعد فك التعديل وكتيارات وجهود بشكل عام، كما لا يمكن معرفة تفاصيل معالجتها داخل المعالج الصغرى لعقدة الحساس.

9- محاكي MARS [17]، يمتلك محاكي MARS واجهة مصممة باستخدام لغة الجافا، ويستخدم لمحاكاة لغة التجميع MIPS وهو يركز على العمليات ضمن معالجٍ واحد دون وجود أي إمكانية للتفاعل بين عدة معالجات (أي عندما تعمل عدة عقد معاً وبشكلٍ متزامن ضمن نفس البيئة)، بخلاف بيئتنا المصممة والتي تدعم تفاعل معالجات عدة عقد ضمن النافذة GUI الخاصة بمحاكي WSN الافتراضي المصغر.

نلاحظ مما سبق أنَ معظم المحاكيات المستخدمة في مجال شبكات الحساسات اللاسلكية لا تقدم تفاصيل دقيقة عن عمليات المعالجة ضمن العقدة من جهة، ومن جهة أخرى فإن أغلب المحاكيات التي تُظهر عمليات المعالج الصغرى وتتعامل معها لا تأخذ بعين الاعتبار عمل عدة معالجات في آن واحد (كما هو الحال في شبكات WSN)، ويبين الجدول (1) نقاط المقارنة بين هذه المحاكيات المذكورة.

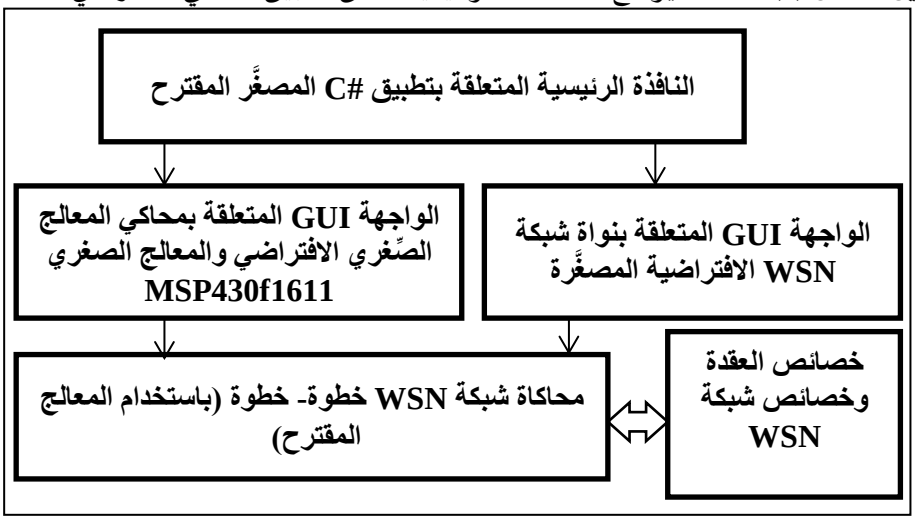
جدول (1) المقارنة بين المحاكيات المذكورة

اسم المحاكي	القدرة على تعديل العمليات على مستوى معالج العقدة من وجهة نظر المستخدم	القدرة على مراقبة العمليات على مستوى معالج العقدة من وجهة نظر المستخدم	دعم التفاعل بين عدة معالجات صغرية تعمل معاً (شبكة WSN)
NS2	-	-	√
OMNeT++	-	-	√
NETTOPO	-	-	√
COOJA	-	√	√
OPNET	-	-	√

اسم المحاكي	القدرة على تعديل العمليات على مستوى معالج العقدة من وجهة نظر المستخدم	القدرة على مراقبة العمليات على مستوى معالج العقدة من وجهة نظر المستخدم	دعم التفاعل بين عدة معالجات صغرية تعمل معاً (شبكة WSN)
MATLAB	-	-	√
MPLAB	-	√	-
PROTEUS	-	√	-
MARS	√	√	-

المبحث الثاني- نواة منصة المحاكاة البرمجية الافتراضية المصغرة

تم في البحث [18] اقتراح طريقة جديدة لبناء نموذج نواة لنظام منصة محاكاة (خطوة – خطوة) لمحاكاة تفاصيل العمليات المتعلقة بالبروتوكولات وذلك على مستوى منخفض (مستوى لغة الAssembly)، حيث يصف هذا النموذج عن قرب مجموعة عمليات المعالج، والخوارزميات المستخدمة لتنفيذ البروتوكولات ضمن معالج عقدة الحساس، ويبين الشكل (5) مخططاً يوضح العلاقات الوظيفية ضمن تطبيق المحاكي الافتراضي.



الشكل (5) العلاقات الوظيفية ضمن تطبيق المحاكي الافتراضي المصغر المقترح

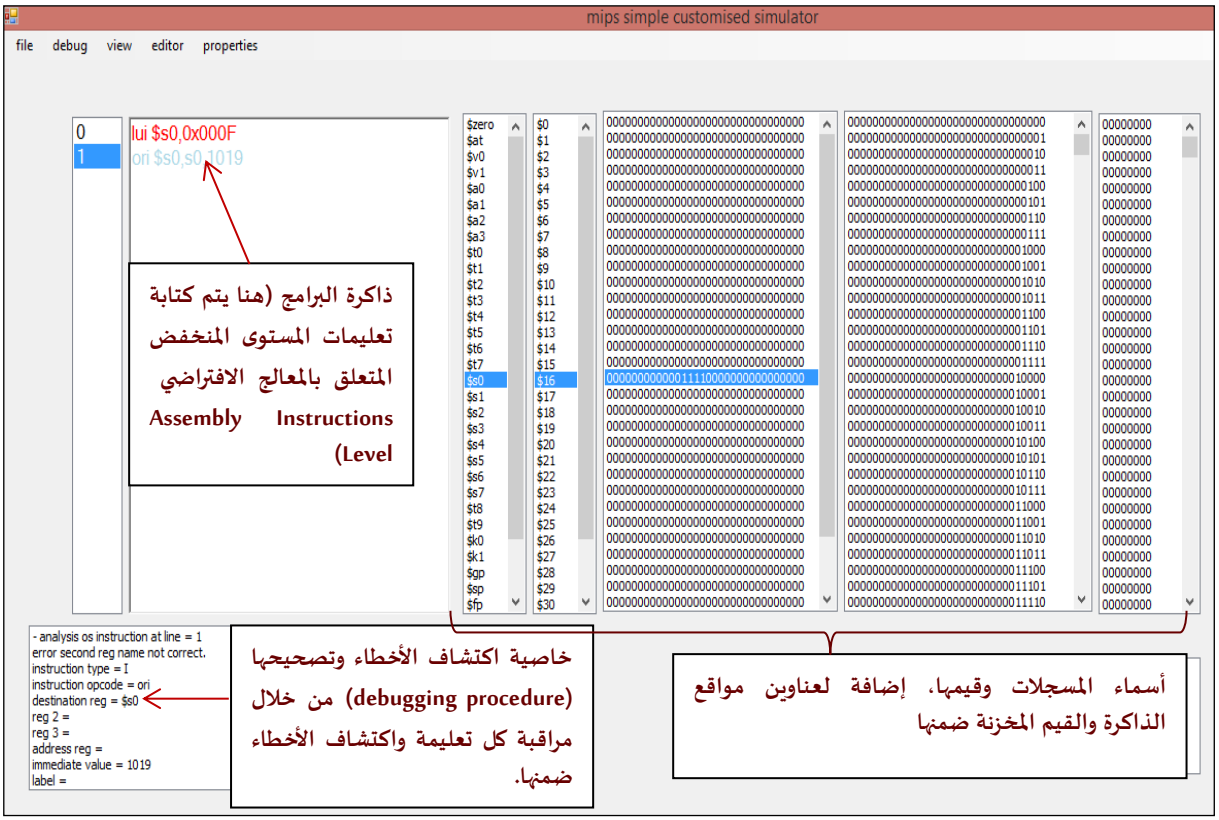
اولاً- الخصائص المتعلقة بالمعالجات الصغرية المستخدمة ضمن نواة المنصة المقترحة

تمَّ اعتماد نوعين من المعالجات ضمن بيئة المحاكي المقترحة، حيث أضفنا بدايةً عمليات معالج صغري افتراضي يستخدم مجموعة جزئية من تعليمات المعالج MIPS(Million Instructions Per Second)والتي تستخدم في نمط المستخدم User Mode [19]، وذلك بهدف تبسيط الخوارزميات المستخدمة ضمن التطبيق من جهة، ولأن هذه المجموعة الجزئية من تعليمات MIPS تكون فعالة بشكلٍ كافٍ لبرمجة أي خوارزمية باستخدام لغة التجميع من جهةٍ أخرى، (ولتحقيق ذلك تم بناء مترجم Compiler مبسط، مرن وقابل لإضافة تعليمات جديدة).

يمكن تعريف مجموعة الخصائص والمتطلبات المتعلقة بهذا المعالج وفق الآتي:

- الهدف من المعالج المقترح هو تحقيق المتطلبات الأساسية اللازمة لإنجاز عمليات الخوارزميات ضمن تطبيق WSN المصمم وذلك من خلال استخدام مجموعة جزئية من التعليمات حيث تكون هذه المجموعة الجزئية فعالة بشكل كاف للسماح ببرمجة أي خوارزمية.
- يستخدم هذا المعالج طريقة مبسطة لمعالجة المقاطعات والاستثناءات حيث لا يوجد مُكدِّس لمعالجة المقاطعات والاستثناءات المتعددة، وذلك لتبسيط تنفيذ الخوارزميات ضمن واجهتنا ولإنقاص تعقيد الدارات المستخدمة لبناء المعالج المقترح (إذا أردنا ان نفعل ذلك باستخدام لوحات FPGA على سبيل المثال).
- تعريف خطوط نقل منفصلة لكل من ذاكرة البرامج وذاكرة المعطيات.
- كما هو الحال في المحاكيات MARS، يركز المعالج المقترح على عمليات المسجلات والذواكر، مع وجود إمكانية ضبط عدد الـClockالمتعلق بكل تعليمة.
- يستخدم نفس مسجلات الأغراض العامة المستخدمة في معالجات MIPS (سعة كل مسجل32bits)، إضافة لاستخدام 32 خط عنونة.
- يتعامل مع القيم الرقمية الصحيحة(أي أننا لا نستخدم تعليمات الفاصلة العائمة).
- حجم ذاكرة البيانات هو 1kByte من مساحة التخزين، وتخصص العناوين من 1019 لـ 1023 لوحدة الاتصال.

يبين الشكل (6) الواجهة الرئيسية المعبرة عن عمليات المعالج الصغرى الافتراضي ضمن إحدى العقد.



الشكل (6) الواجهة GUI المعبرة عن بيئة محاكي المعالج الصغرى الافتراضي

كما تم اعتماد المعالج المذكور لبناء نواة محاكي WSN، وهذا يعني أنّهُ بالإمكان إضافة أي عدد من العقد والتي تستخدم المعالج المقترح كخطوة أولى، مع الأخذ بالحسبان الخصائص الزمنية المتعلقة بالإرسال والاستقبال

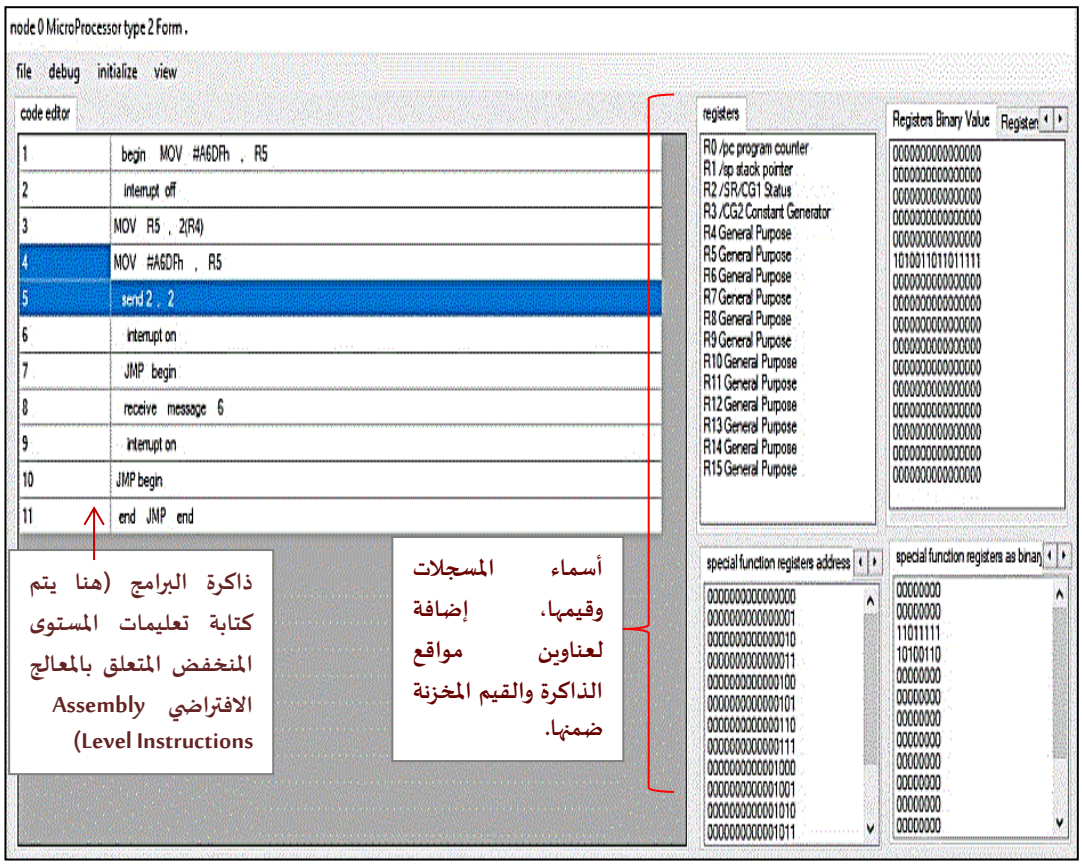
اعتماداً على نوع التعديل وبعض الافتراضات المتعلقة بالمكونات الداخلية لوحدة الإرسال والاستقبال Transceiver Unit وذلك بهدف التبسيط والتسهيل.

إضافةً إلى ما سبق فقد قمنا بتطوير العمليات المتعلقة بنواة البيئة المصممة من خلال إعطائها إمكانية التعامل مع تعليمات معالج حقيقي مستخدم ضمن عقد الحساسات، حيث قمنا بإضافة مجموعة تعليمات المعالج الصغرى [20] MSP430f1611Processorوالذي ينتمي إلى سلسلة معالجات Texas Instruments MSP430X1XX Microcontroller، ويُستخدم هذا المعالج ضمن بعض أنواع عقد الحساسات مثل Tmote Sky [21] والتي تعد نموذجاً لعقد الحساسات ذات الاستهلاك المنخفض جداً للطاقة، وتتبع هذه العقدة المعايير الصناعية مثل USB وIEEE 802.15.4 وتتحسس معلومات الحراة والرطوبة والضوء، ويمتلك هذا المعالج عدداً من الخصائص نذكر منها:

- تأخذ الهيكلية الأساسية ضمن هذا المعالج مبدأ (Reduced Instruction Set Compiler) RISC وذلك باستخدام 27 تعليمة و7 أنماط للعنونة، إضافة إلى إمكانية استخدام أي تعلمية ضمن أي نمط من أنماط العنونة.
- إمكانية الوصول للمسجلات بشكل كامل بما في ذلك مسجلات الحالة Status Registers، عداد البرنامج Program Counter، ومؤشر الكدسةStack. Pointer
- يحوي المعالج الصغرى 16 مسجلاً وكل مسجل ذو حجم 16 bits، وتقلل المسجلات (ذات الحجم 16 bits) من الحاجة لجلب القيم من الذاكرة، وتتم عمليات المسجلات خلال دورة مفردة Single- Cycle.
- تولد مسجلات (توليد القيم الثابتة) Constant Generator Registersستة قيم تُستخدَم كقيم مباشرة وتنقص من حجم الكود.
- تسمح خطوط عناوين 16 bits بالوصول المباشر والقفز عبر كامل مجال الذاكرة.
- تسمح خطوط البيانات (16 bits) بالمعالجة المباشرة للمعاملات على امتداد الكلمة Word- Wide Arguments.
- إمكانية نقل البيانات بشكلٍ مباشر من ذاكرة لذاكرة دون الحاجة لاستخدام مسجل وسيطي Intermediate Register.
- إمكانية تعريف صيغ التعليمات والعنونة ككلمة Word أو كبايت Byte.

كما نعرف ضمن هذا المعالج 7 أنماط عنونة للمعامل المصدر Source Operand و4 أنماط عنونة للمعامل الهدفDestination Operand، مما يعطي إمكانية عنونة كامل مجال العنونة دون أي استثناء.

يبين الشكل (7) الواجهة GUI المعبرة عن بيئة محاكي المعالج الصغرى MSP430f1611Processor.



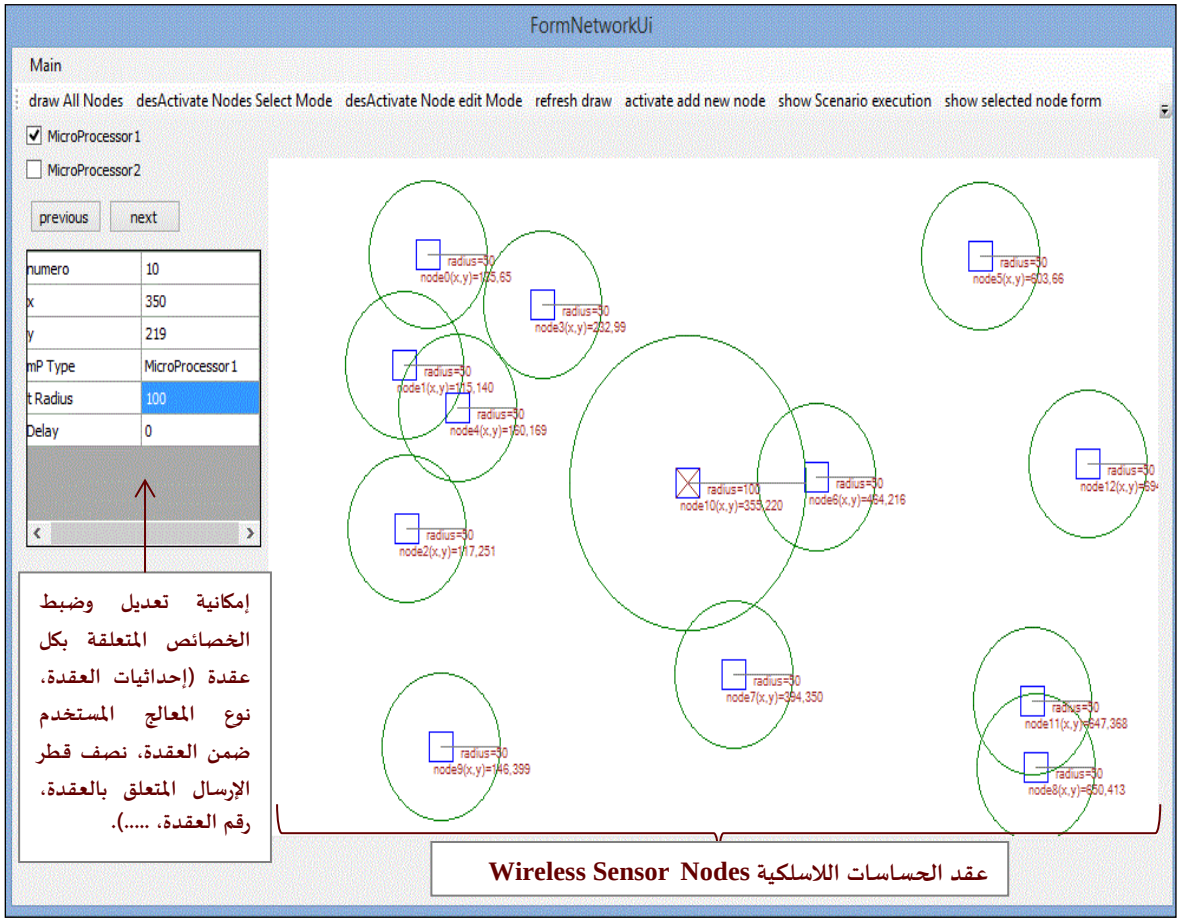
الشكل (7) الواجهة GUI المعبرة عن بيئة محاكي المعالج الصغرى MSP430f1611Processor

ثانياً- الخصائص والميزات المتعلقة بنواة محاكي WSN الافتراضي المقترح

تمتلك بيئة المحاكي عدداً من الميزات نذكر منها:

- سهولة برمجة هذه البيئة الافتراضية، حيث أننا نستخدم التعليمات والأدوات الأساسية المتوفرة في Microsoft Visual Studio 2015 (Windows Forms Applications)، إضافةً إلى أنَّVisual Studio يعطينا سهولة بعملية اختبار أي كود برمجي، وذلك باستخدام خاصية الـ Debug والتي تؤمن مرونة بمقدرات التعديل لنتمكن عبر ذلك من تضمين أي خاصية ضمن واجهة المنصة.
- إمكانية تطبيق خاصية اكتشاف الأخطاء وتصحيحها Debugging Procedure خلال تنفيذ التعليمات ضمن تطبيقنا، الأمر الذي يمكننا من مراقبة نتائج كل عملية واكتشاف الأخطاء التي يمكن أن تكون موجودة، إضافةً إلى إمكانية ضبط الـ Clock المتعلق بكل تعليمة وفقاً لنوع المعالج المقترح.
- إظهار قيم مواقع الذاكرة والمسجلات وفق الترميز الثنائي (ذو الأساس 2)Binary Encoding، والعشري (ذو الأساس 10) Decimal Encoding، إضافةً إلى الترميز السِّتَّ عشري (ذو الأساسHexadecimal Encoding 16) ، والقدرة على تعديل هذه القيم عند الضرورة، إضافة لإمكانية إظهار Source Code Format لكل تعليمة.
- إمكانية إضافة وحذف وتحريك العقد ضمن الواجهة المقترحة وتغيير خصائص هذه العقد مثل نصف قطر الإرسال المتعلق بالعقدة.

يبين الشكل (8) الواجهة الرئيسية المعبرة عن نواة محاكي WSN الافتراضية المصممة.



الشكل (8) الواجهة GUI المعبرة عن نواة محاكي WSN الافتراضية المصممة

المبحث الثالث- خوارزمية المصادقة بين عقدتين مستقلتين من عقد الحساسات عبر الواجهة المصممة [22]:

يشكل التوجيه جزءاً مهماً من العمليات الأساسية ضمن شبكات الحساسات اللاسلكية WSN والتي تحافظ على تشغيل الشبكة واستمرار عملها بالشكل الصحيح، لاسيما عندما يتم نشرها ضمن بيئة معادية Hostile Environment حيث أنها تكون معرضة لأنواع مختلفة من الهجمات، وتعود هذه الهجمات للعديد من العوامل نذكر منها الطبيعة غير الآمنة لقنوات الاتصال اللاسلكية من جهة، ونتيجة للضعف الموجود ضمن البنى التحتية المقاومة لعمليات التلاعب بعمليات التوجيه من جهة أخرى [23]. لذا كان اختيارنا إحدى خوارزميات أمن التوجيه المستخدمة في شبكات WSN وهي خوارزمية المصادقة بين عقدتين مستقلتين من عقد الحساسات Authentication Scheme between Two Individual Sensor Nodes، وذلك لاختبارها ضمن بيئتنا الافتراضية المصممة من أجل تقييم أدائها، وتوضيح سير العمليات وآلية تحقيق المصادقة بين العقد، بالإضافة إلى استعراض مبدأ العمل ضمن الواجهة المتعلقة بنواة البيئة المصممة، وسنعتمد في هذا الاختبار على استخدام عمليات المعالج الصغرى الافتراضي كمعالج مستخدم ضمن عقد الحساسات، والتي سيتم إدخال عملياتها اعتماداً على مترجم مبسط Simplified Compiler.

أولاً- الخطوات المتعلقة بتنفيذ عمليات الخوارزمية المذكورة

عندما تريد عقدة الحساس أن تجري اتصالاً آمناً مع عقدة حساس أخرى تحتاج أولاً كل عقدة أن تتحقق من هوية العقدة الأخرى (إجراء المصادقة)، وفي هذه الهيكلية سيتم افتراض أنه لن يتم تخزين معلومات المفاتيح في عقد حساسات فردية، وبدلاً من ذلك كل معلومات المفاتيح سيتم تخزينها ضمن عقد حساسات ذات مستوى عالي(High End Sensor Node (Master Node، ومن خلال حماية الـ Master Nodeيمكننا جعل كامل شبكة الحساسات آمنة، ويتم ذلك وفق الآتي:

لنفرض وجود عقدتين A، B ضمن الشبكة وتريدان أن تتصلا مع بعضهما البعض بشكلٍ آمن، حيث تحتوي العقدة ذات المستوى العالي Master Node على المفاتيح المتعلقة بكل عقد الحساسات K1,K2,K3,...,Kn وهوية هذه العقد IDs والصيغة المشفرة من هذه الهويات مع المفاتيح المرتبطة بها أي .....(IDA)K1,(IDB)K2,(IDC)K3 وذلك قبل نشر العقد. وتكون المفاتيح متوفرة فقط عند الـ Master Node، حيث تحوي العقدة A على الهوية IDA و(IDA)K1 وتحوي العقدة B على الهوية IDB و K2 (IDB).

- تولد العقدة A رقماً عشوائياً Nonce A وتحسب رمز مصادقة الرسائل MAC (Message Authentication Code) باستخدام الـ Nonce A والـ (IDA)K1، وتقوم بإرسال الرسالة التالية للعقدة B:

IDA||Nonce A||MAC{(IDA)K1||NonceA}

- عند استقبال الرسالة، تحاول العقدة B حساب رمز المصادقة MAC من أجل المصادقة، ولكنّ المفتاح (IDA)K1 متوفر فقط عند الـ Master Node، لذا تولد العقدة B رقماً عشوائياً بعد استقبال الرسالة من العقدة A وهو Nonce B، ثمَّ ترسل لـ Master node الرسالة التالية بهدف التحقق من العقدة.

Nonce A||Nonce B|| IDB||H{(IDB)K2}||IDA

- تقوم العقدة (Master node) بإجراء مصادقة من خلال حساب تابع البعثرة Hash Function عبر (IDB)K2 والذي يكون مع الـ Master node، وبعد مصادقة العقدة B من خلال الرسالة المستقبلة من العقدة B والتأكد من صحة الرسالة تقوم بإرسال (IDA)K1 للعقدة B.
- تقوم العقدة B بحساب MAC{(IDA)K1||Nonce A} ومقارنته مع الـ MAC المقدم ضمن الرسالة القادمة من العقدة A، وإذا تساوت قيم الـ MAC مع بعضها البعض تتأكد عندئذ العقدة B من أن العقدة A هي عقدة موثوقة ضمن الشبكة وتتصل معها بشكلٍ آمن.

ثانياً- بعض النقاط المتعلقة بآلية اختبار وتنفيذ الخوارزمية المذكورة ضمن البيئة المقترحة

- سنقوم بتطبيق الخوارزمية المذكورة ضمن نواة البيئة المصممة باستخدام مترجم وسيطي مبسط Compiler Simplified يتيح للمستخدمين التعامل مع تعليمات المستوى المنخفض(مستوى لغة التجميع) Assembly Language Level دون وجود حاجة لإدخال هذه التعليمات بشكل مباشر، بل يتم عوضاً عن ذلك إدخال مجموعة من الأوامر المعبرة عن عمليات معينة ضمن العقد، وإدخال تعاريف المتحولات المستخدمة ضمن معالج كل عقدة واللازمة لإنجاز هذه الأوامر ليتم تحويلها بشكلٍ تلقائي إلى تعليمات Assembly Instructions، وهذا ما يعطي الواجهة المصممة مرونة سهولة كبيرة في التعامل مع القيم وإدخالها، وهذا ما سنستعرضه لاحقاً أثناء تطبيق الخوارزمية المذكورة ضمن البيئة المقترحة.
- سنقوم أثناء تطبيق الخوارزمية بتهيئة مواقع الذاكرة المتعلقة بالعقد A, B, and Master node.

- من أجل التحقق من سلامة ودقة نتائج الخوارزمية المطبقة على المحاكي المقترح، سنستخدم صيغاً مبسطة عند إجراء الحسابات المتعلقة بتابع مصادقة الرسالة MAC{(IDA)K1||NonceA} وبتابع البعثرة Hash Function المتعلق بـ (IDB)K2أي (H{(IDB)K2})، حيث سنعتمد لحساب MAC{(IDA)K1||NonceA} التابع mod{(NonceA+IDA(KA))/5}، كما سنستخدم لحساب H{(IDB)K2} التابع mod {IDB(K2)/7.
- سنعتمد بروتوكولاً مبسطاً ضمن ترويسة الرسالة بهدف تمييز وجهة الرسالة، ويأخذ هذا البروتوكول القيم المدرجة ضمن الجدول (2) بالحسبان.

جدول (2) معلومات ترويسة الرسالة لتحديد وجهة الرسالة ضمن الخوارزمية المدروسة

Message Type	Message Header (1 Byte)
رسالة طلب إجراء اتصال آمن.	01010010
رسالة استجابة (رد على رسالة طلب اتصال آمن). (العقدة B - - - - - - - - - ->العقدة A)	01010011
رسالة طلب معلومات من العقدة ذات المستوى العالي Master. (العقدة B - - - - - - - - - ->العقدة Master)	10010001(part1)
	00100001(part2)
رسالة استجابة من العقدة ذات المستوى العالي Master. (العقدة Master - - - - - - - - - ->العقدة B)	01010000

النتائج والمناقشة

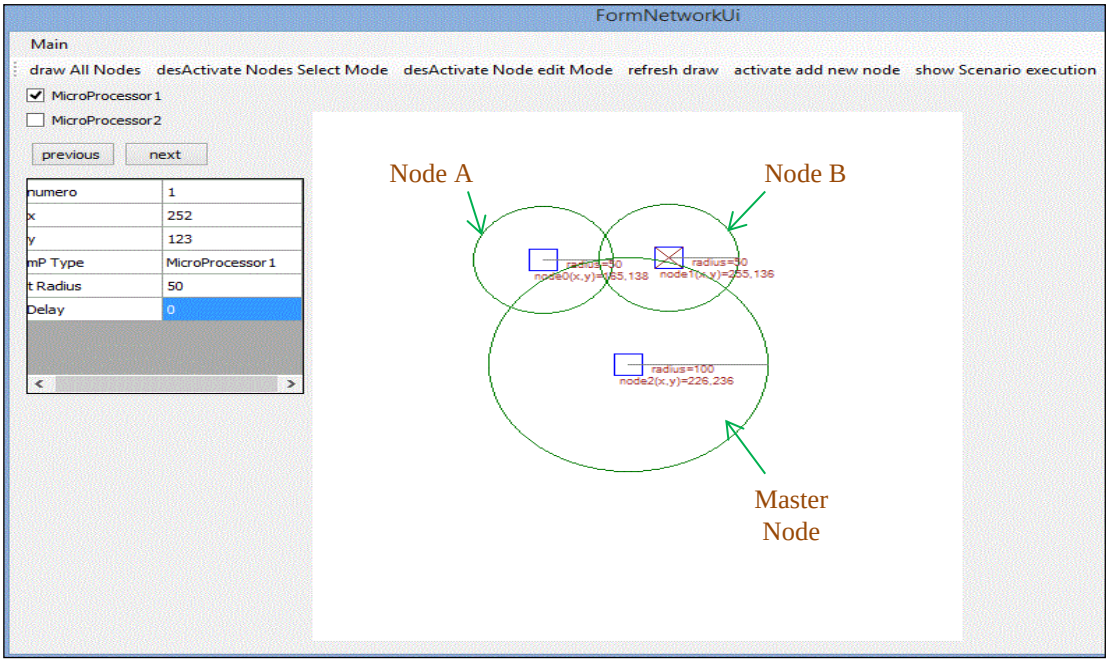
سنأخذ بالحسبان ضمن الاختبار البارامترات التالية:

- عدد العقد: 3 (بما يوضح سير العمليات ضمن الخوارزمية المذكورة).
- عدد مرات التكرار: 10 مرات، بما يكفي لتأكيد النتائج ومقاربتها بشكلٍ دقيق.
- نوع المعالج المعرف ضمن العقد: سنستخدم مجموعة عمليات المعالج الصغرى الافتراضي.

سيناريوهات العمل:

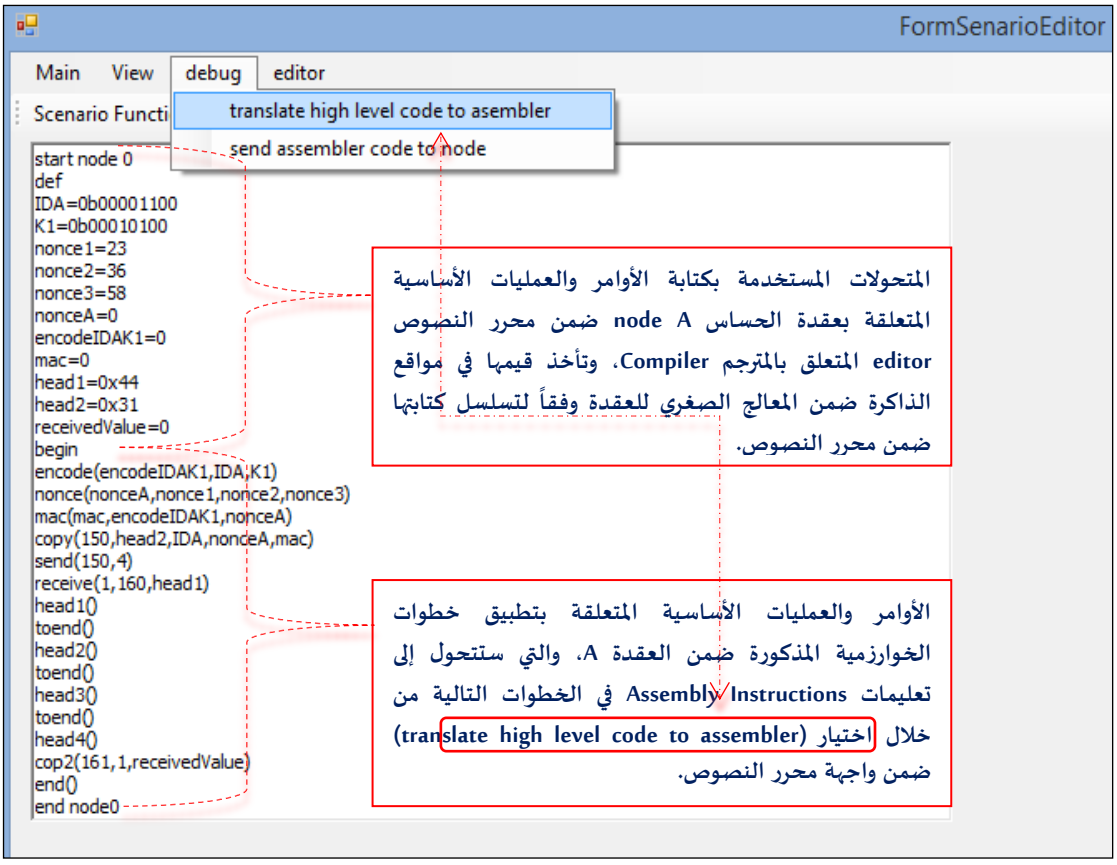
سنقوم بتنفيذ الاختبار وفقاً للخطوات الآتية:

✓ إضافة ثلاث عقد: العقدة A (node0)، العقدة B (node1)، والعقدة ذات المستوى العالي Master (node2) Node كما هو مبين بالشكل (9).



الشكل (9) إضافة العقد الثلاث إلى الواجهة GUI المتعلقة بنواة محاكي WSN المصمَّمة

✓ إدخال مجموعة من الأوامر المعبرة عن عمليات الخوارزمية ضمن العقد باستخدام محرر التعليمات المصمَّم لكل عقدة، حيث يبين الشكل (10) محرِّر التعليمات الخاص بالعقدة A.



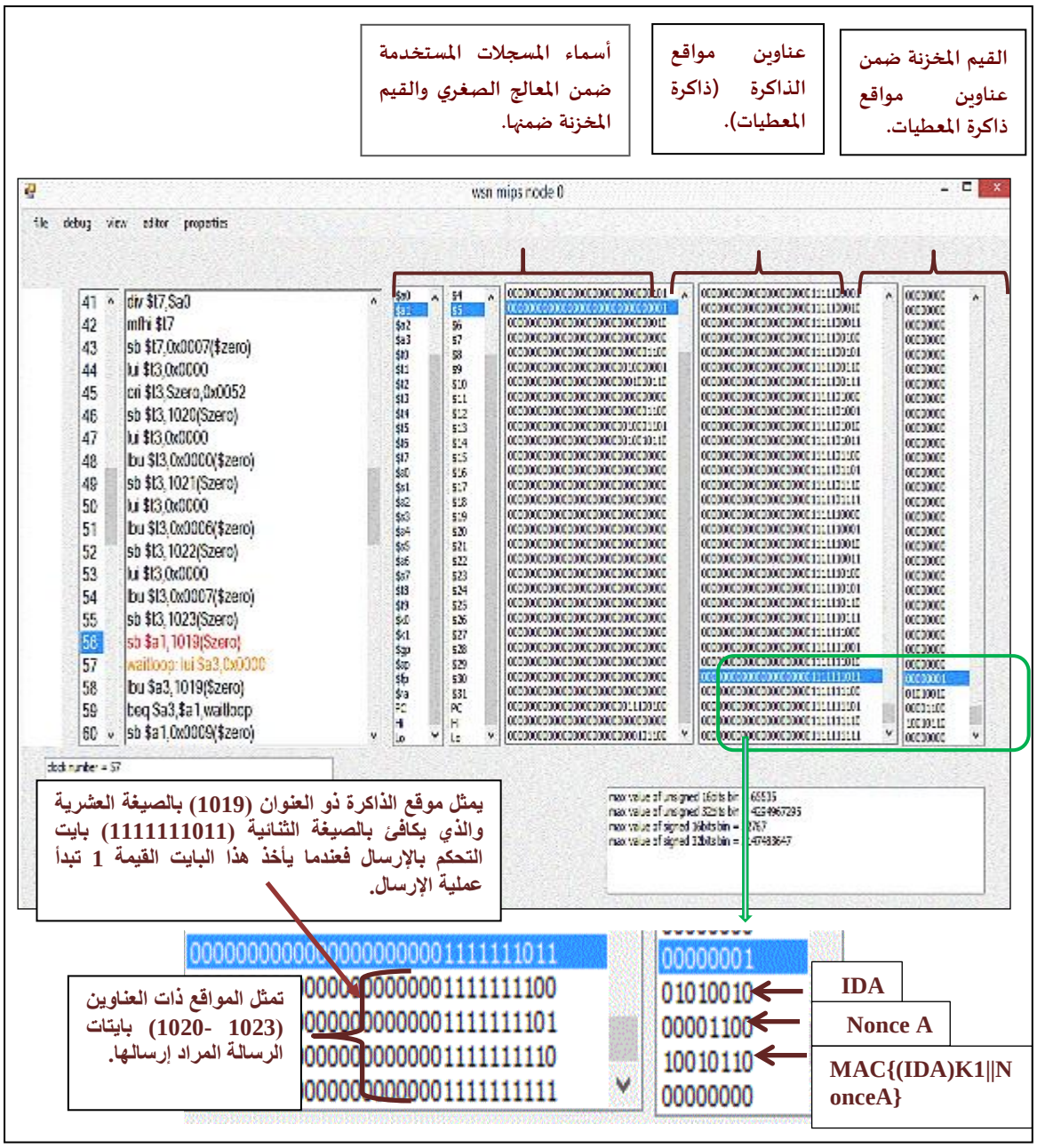
الشكل (10) واجهة المحرر editor الخاص بالمترجم والذي يتم ضمنه إدخال الأوامر المعبرة عن تطبيق الخوارزمية ضمن العقدة A

- ✓ تحويل الأوامر إلى تعليمات المستوى المنخفض (Assembly Instructions) الملائمة؛ باستخدام مترجم مبسط Simplified Compiler، ويتم ضمن التعليمات تهيئة العقد بالقيم الابتدائية، بالإضافة لإجراء الحسابات المتعلقة بخطوات تنفيذ الخوارزمية المذكورة (مثل حساب قيمة الـ MAC والرقم العشوائي Nonce A, Nonce B) ضمن كل عقدة.
- ✓ البدء بالتنفيذ (خطوة_ خطوة) ومراقبة تنفيذ العمليات ضمن المعالج المقترح المتعلق بكل عقدة.
- ✓ تكرار التنفيذ ضمن الواجهة GUI المتعلقة بالمنصّة المقترحة وفق السيناريوهات، وبمعدل (10) مرات للسيناريو.

السيناريو الأول: تريد العقدة A إجراء اتصال آمن مع العقدة B، ولكنها تقع على مسافة بعيدة منها، وذلك بافتراض أن العقد نُشِرت بشكلٍ عشوائي:

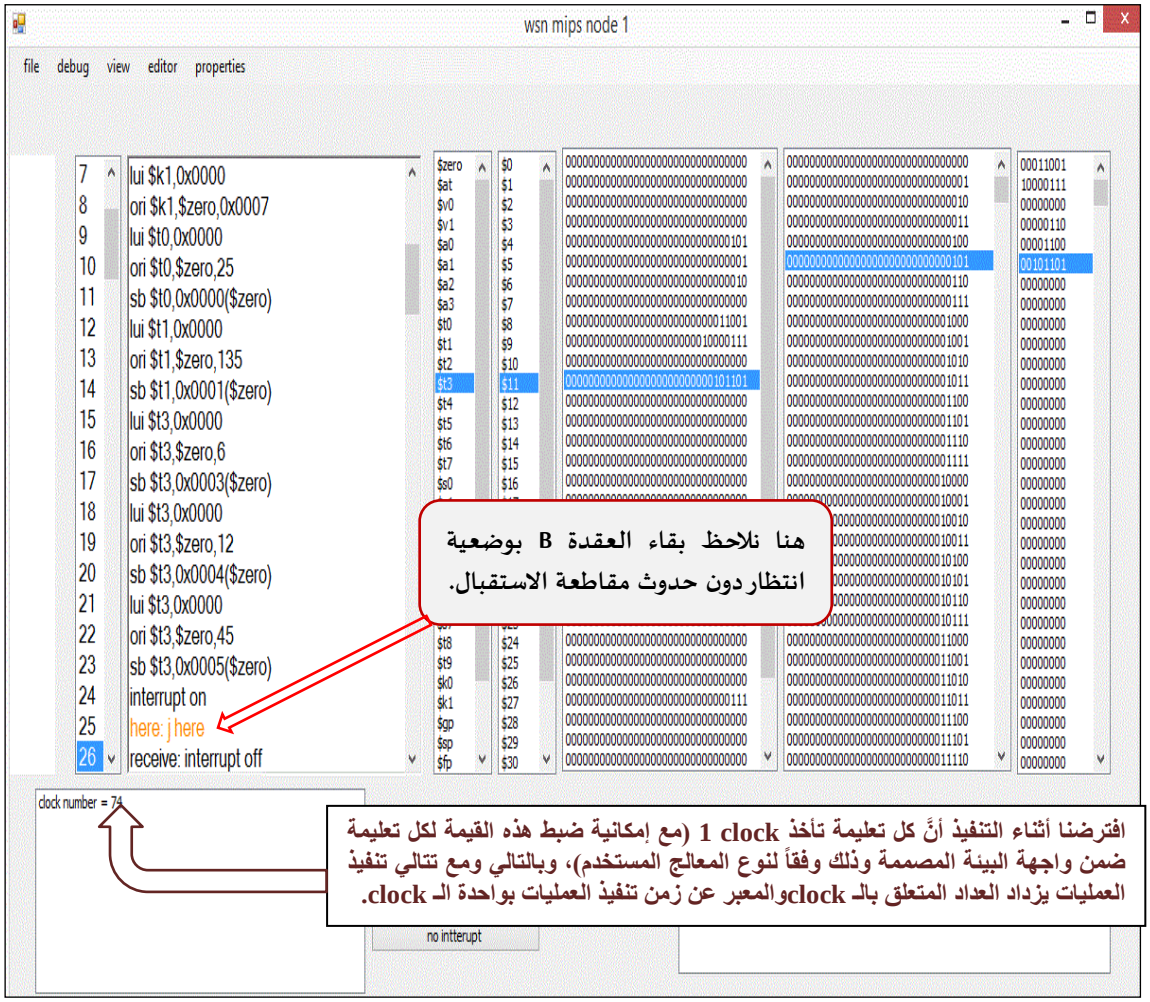
نتائج السيناريو الأول:

بدايةً، وبعد إجراء العقدة A لعمليات التهيئة المتعلقة بها، فإنَّها ستحسب رمز مصادقة الرسائل MAC باستخدام الـ Nonce A والـ K1(IDA)، ثم تبدأ بإرسال الرسالة {IDA||Nonce A||MAC{(IDA)K1||NonceA} للعقدة B كما هو مبين بالشكل (11) وذلك بهدف إجراء اتصال آمن معها، بينما تكون بقية العقد بوضع الانتظار.



الشكل (11) إرسال العقدة A للرسالة {IDA||Nonce A||MAC{(IDA)K1||NonceA} إلى العقدة B

لن تظهر أية استجابة من العقدة B ولن تستقبل الرسالة، وذلك لأن العقدة A تقع خارج مجال الاستقبال المتعلق بالعقدة B كما هو مبين بالشكل(12).



الشكل(12) عدم استقبال العقدة B للرسالة المرسلة من العقدة A

السيناريو الثاني: تريد العقدة Aإجراء اتصال آمن مع العقدة B، ولكنها مهيأة بقيم أولية غير دقيقة لحساب رمز مصادقة الرسائل (عقدة خبيثة تحاول الاتصال بالشبكة أو خطأ أثناء إرسال البيانات).

نتائج السيناريو الثاني:

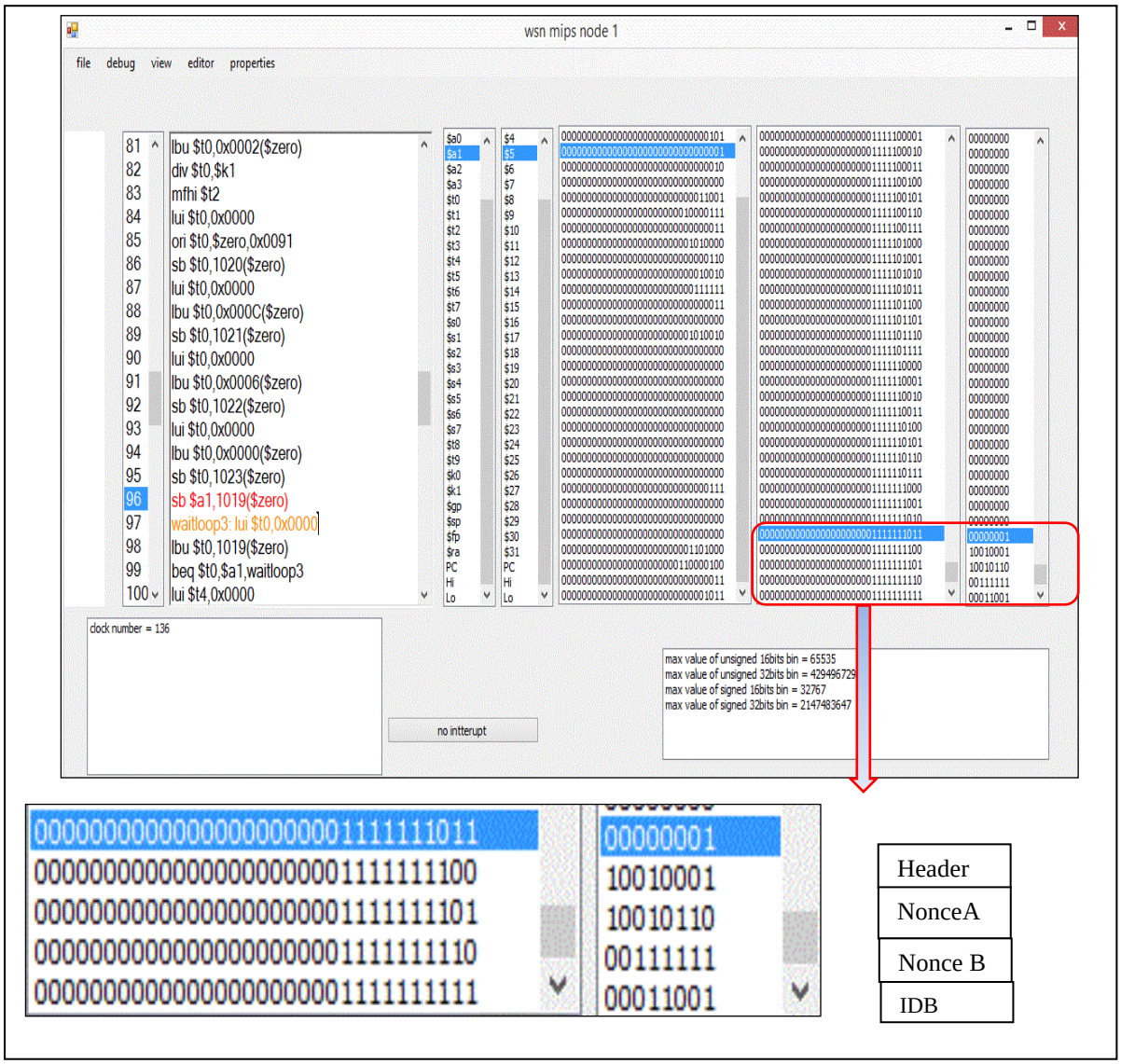
تستقبل العقدة B الرسالة المرسلة لها من قبل العقدة A بعد إجرائها لعمليات التهيئة المتعلقة بها، ولكي تحسب MAC{(IDA)K1||NonceA} بهدف التحقق، فإنها تحتاج لـ (IDA)K1 لذا تراسل العقدة Master للحصول عليه، حيث تحسب H{(IDB)K2 ثم ترسل الرسالة على جزأين كما هو مبين بالشكل (13) والشكل(14) وذلك باعتبار أن الـ Buffer المتعلق بالـ Transceiver وفقاً لافتراضنا هو4Byte، حيث يبين الجدول (3) الجزء الأول الذي يُرسَل ويبين الجدول (4) الجزء الثاني الذي يُرسَل.

جدول (3) الجزء الأول من الرسالة المرسلة من العقدة B إلى العقدة Master

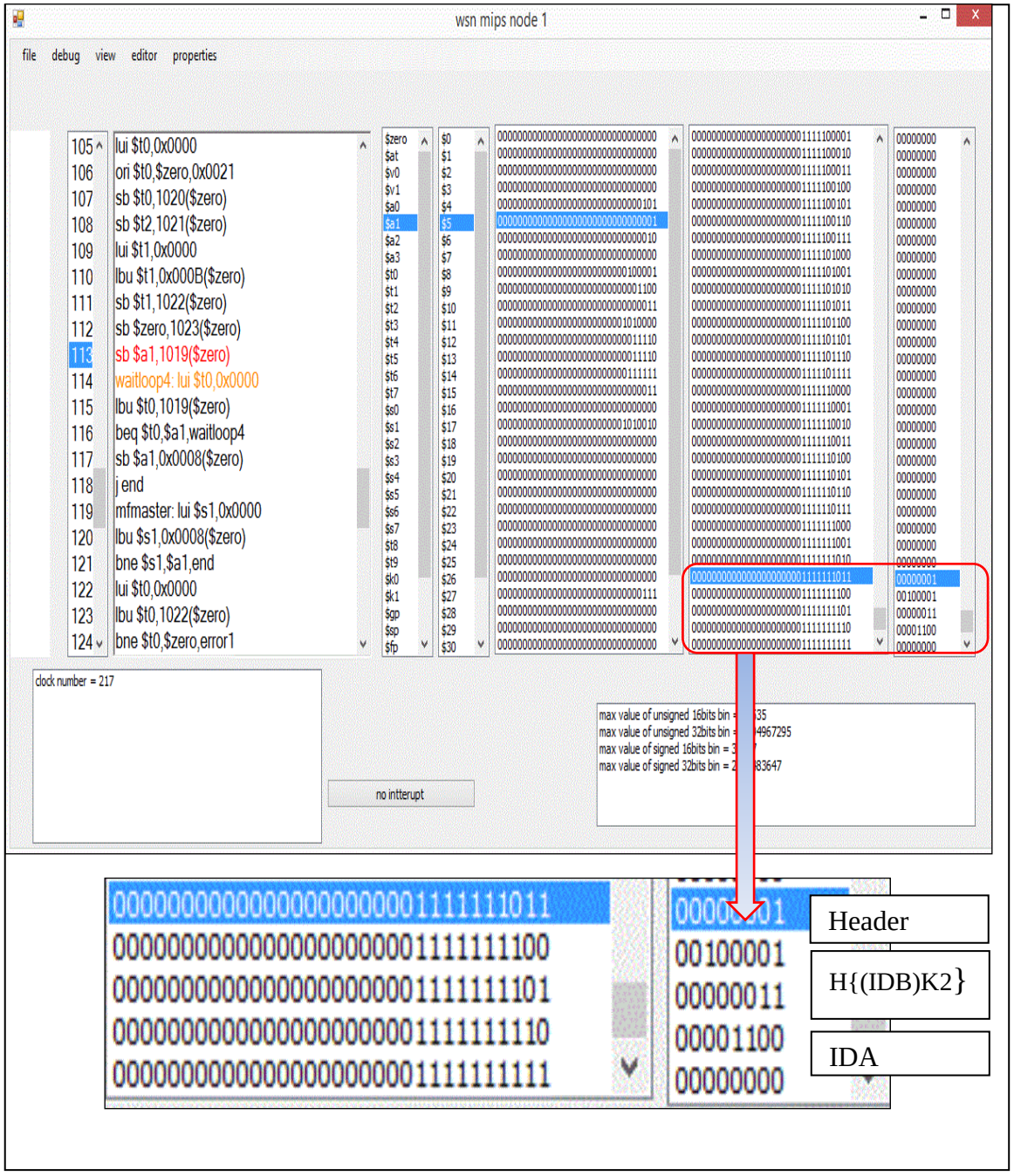
Message Header	Nonce A	Nonce B	IDB
10010001	10010110	00111111	**00011001**

جدول (4) الجزء الثاني من الرسالة المرسلة من العقدة B إلى العقدة Master

Message Header	H{(IDB)K2	IDA	Zero
00100001	00000011	00001100	**00000000**

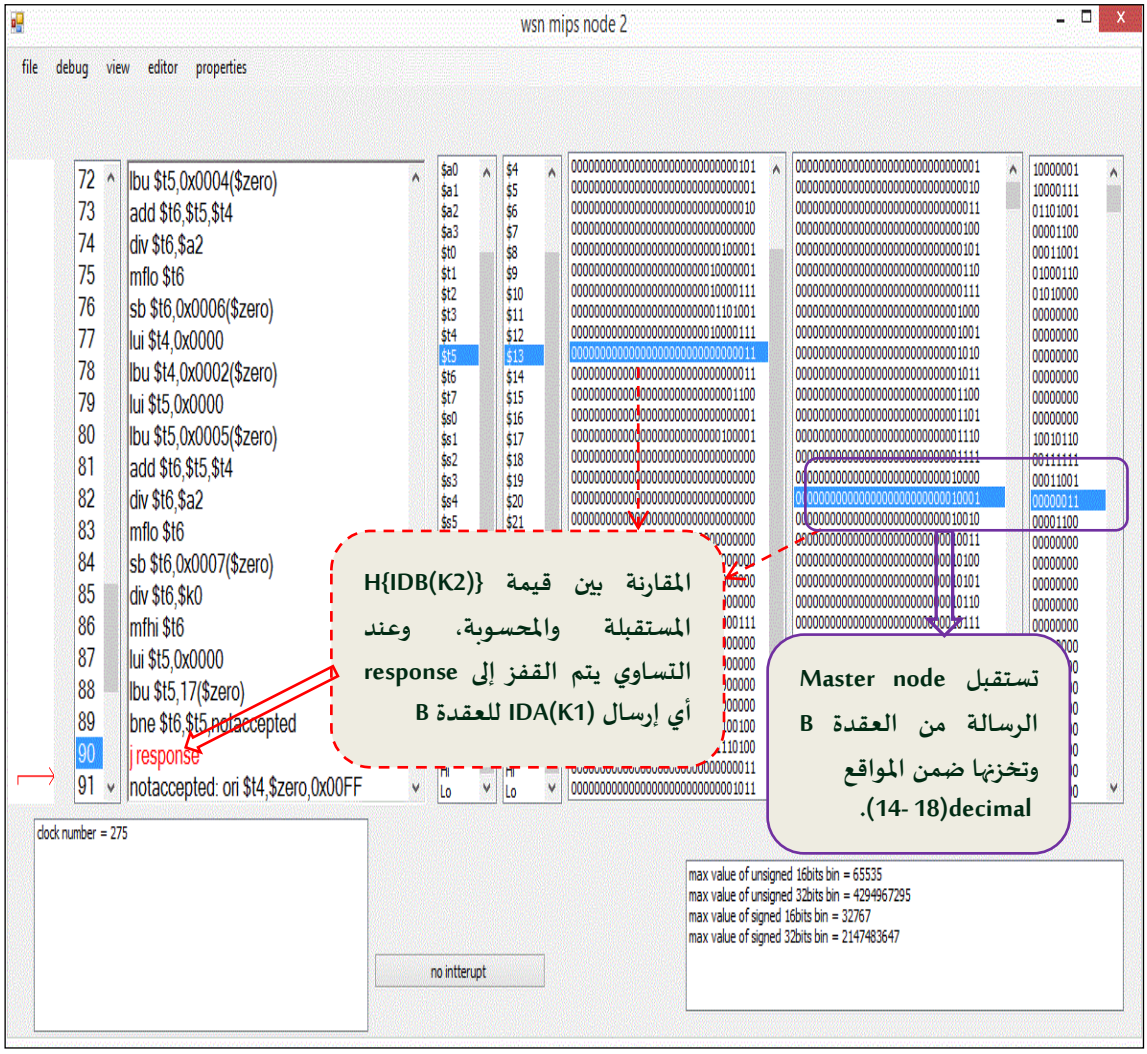


الشكل (13) إرسال العقدة B للجزء الأول من الرسالة (Nonce A||Nonce B|| IDB) إلى العقدة ذات المستوى العالي Master Node



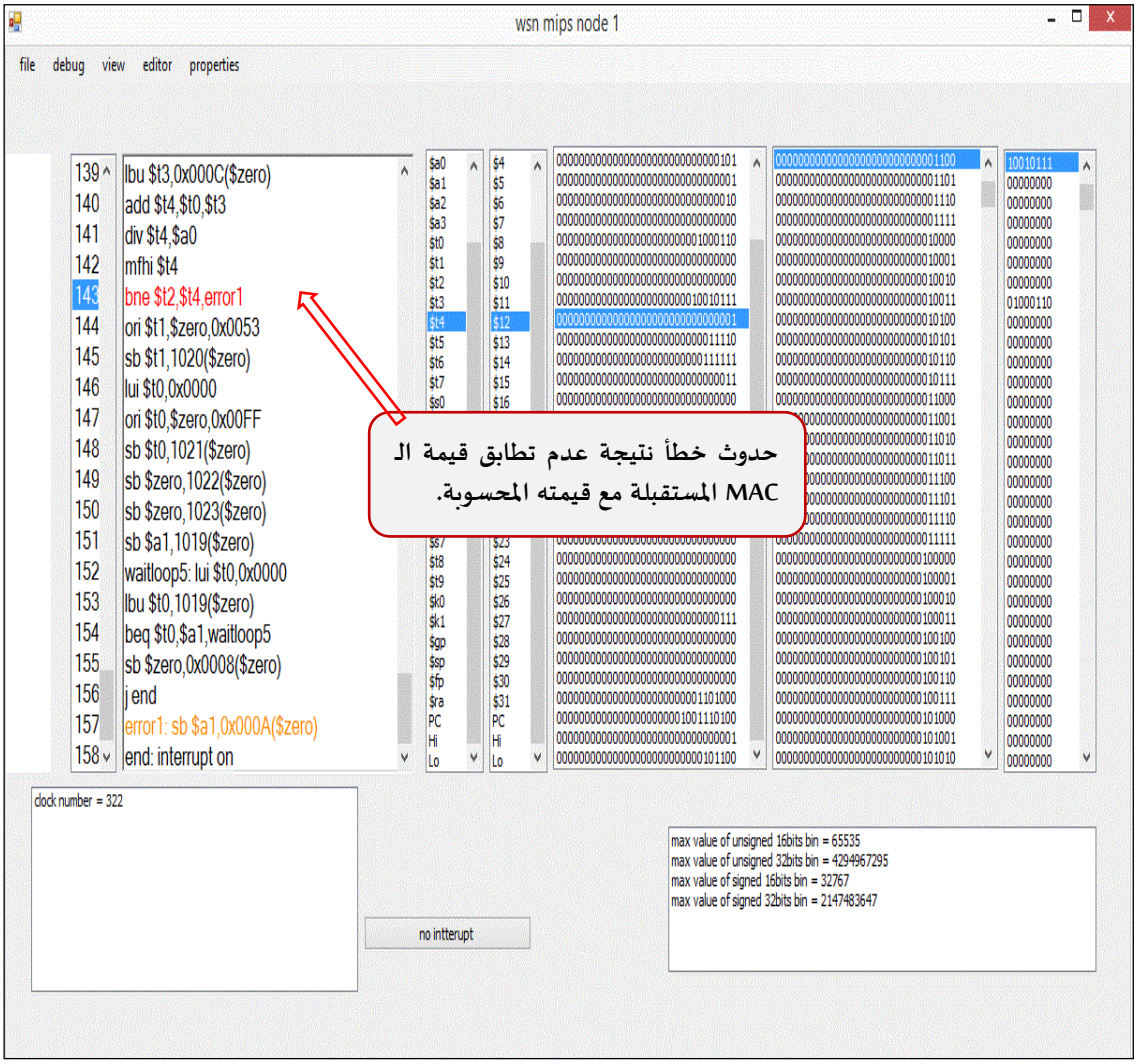
الشكل (14) إرسال العقدة B للجزء الثاني من الرسالة (H{(IDB)K2}||IDA) إلى العقدة ذات المستوى العالي Master Node

بعد ذلك تحسب العقدة Master H{(IDB)K2} وفقاً للتابع{IDB(K2)/7} mod (قيمة المسجل $t6) ثم تقارنه مع القيمة التي استقبلتها من العقدة B وهي القيمة المخزنة ضمن الموقع (17 in Decimal) (قيمة المسجل $t5) وبحال تساوي القيمتين تقوم العقدة Master بالقفز نحو اللافتة label: response والتي تُرسَل ضمنها (IDA)K1 للعقدة B حيث يبين الشكل (15) هذه المقارنة.



الشكل (15) تحقق القعدة ذات المستوى العالي من موثوقية العقدة B قبل أن ترسل لها IDA(K1)

تحسب العقدة B بعد حصولها على IDA(K1) MAC{(IDA)K1||NonceA} وتقارنها مع القيمة المستقبلة من العقدة A كما هو مبين بالشكل (16) ونتيجةً لعدم تساوي القيمتين تعتبر العقدة B أن العقدة A هي عقدة غير موثوقة ضمن الشبكة، وترفض إجراء اتصال آمن معها.



الشكل (16) مقارنة العقدة B بين قيمة الMAC المستقبلة وقيمته المحسوبة

السيناريو الثالث: تريد العقدة A إجراء اتصال آمن مع العقدة B وهي مهيأة بقيم أولية صحيحة لحساب رمز مصادقة الرسائل.

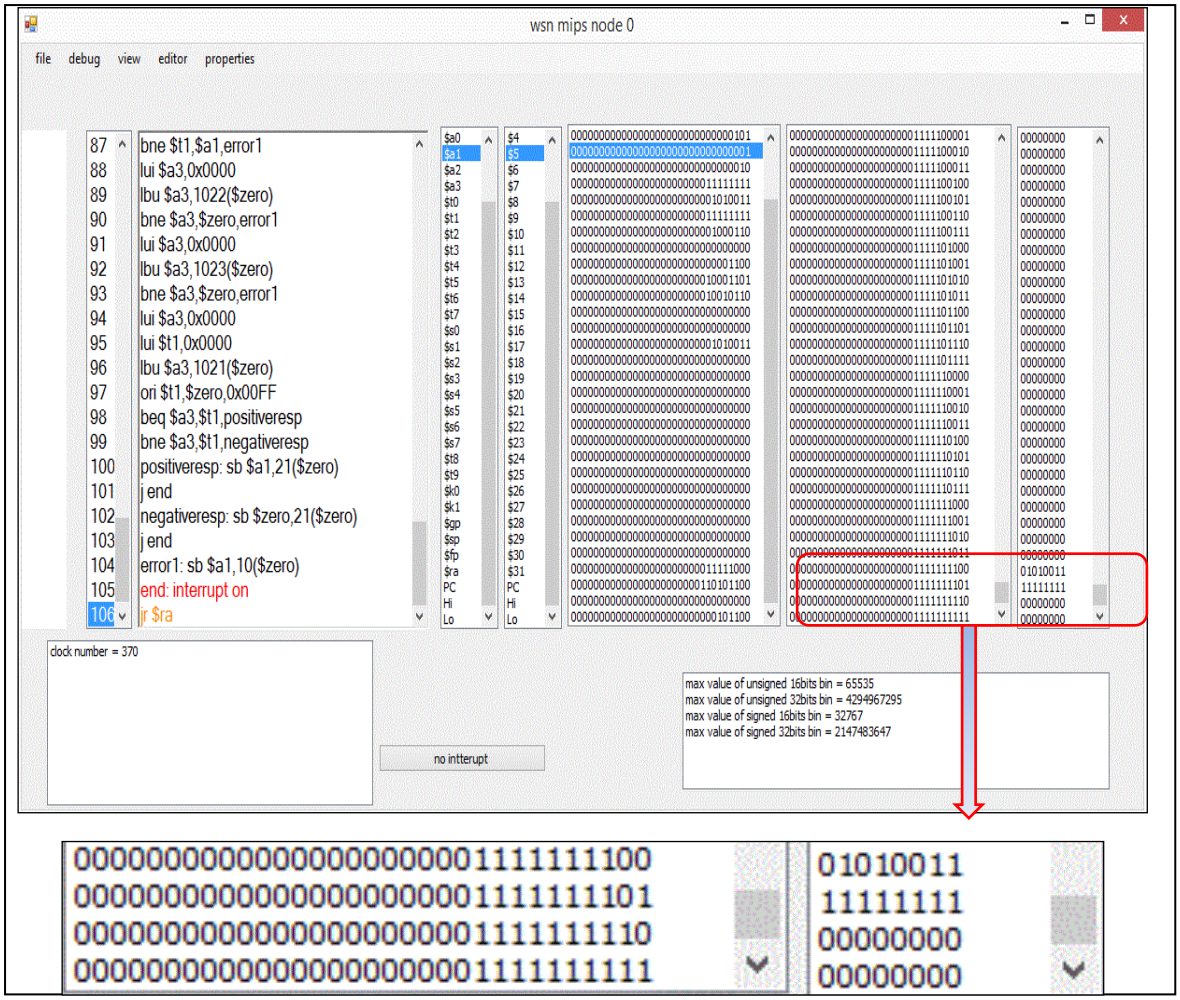
نتائج السيناريو الثالث:

هنا تتكرر نفس خطوات المذكورة ضمن الحالة الثانية باستثناء الخطوة الأخيرة، حيث أن قيمة الـ MAC تكون متساوية وعندئذٍ تقبل العقدة B طلب العقدة A بإجراء اتصال آمن معها، وذلك من خلال إرسال الرسالة المبينة بالجدول (5) لها:

الجدول (5) الرسالة المرسلة من العقدة B إلى العقدة A لإعلامها ببدء إجراء اتصال آمن معها

Message Header	Secure Connection		
01010011	11111111	00000000	**00000000**

حيث تستقبل العقدة A وكما هو مبين بالشكل (17) هذه الرسالة لتبدأ بذلك إجراء الاتصال الآمن مع العقدة B.



الشكل (17) استقبال العقدة A للرسالة الواردة ضمن الجدول (5) من العقدة B للإعلام ببدء إجراء اتصال آمن معها

الخلاصة:

تم ضمن هذا البحث استعراض العمليات المتعلقة بنواة منصة المحاكاة المقترحة والمعالجات الصغرية المعرفة ضمنها، كما تمَّ تطبيق إحدى خوارزميات أمن التوجيه وهي خوارزمية المصادقة بين عقدتين مستقلتين من عقد الحساسات، وذلك ضمن بيئة المحاكي المقترح بوجود مترجم مبسط Simplified Compiler، الأمر الذي أتاح لنا إمكانية مراقبة وتتبع تسلسل تنفيذ العمليات على المستوى المنخفض واختبار آلية مبسطة لتحديد وجهة الرسالة، حيث تم تطبيق ثلاثة سيناريوهات مختلفة من أجل تقييم أداء منصة المحاكاة المقترحة.

وقد بينت النتائج مرونة وفعالية المنصة المقترحة في تتبع سير العمليات ضمن عقد الحساسات اللاسلكية على مستوى لغة التجميع (Assembly Language).

التوصيات والمقترحات.

بناءً على النتائج المذكورة والتي تستعرض عمليات المنصة المقترحة ومرونتها كبيئة افتراضية مصمَّمَة لمحاكاة عقد الحساسات على المستوى المنخفض، فإنَّنا نوصي بتطوير النواة المتعلقة بهذه المنصة في المستقبل وذلك من خلال:

1- إضافة أنواع أخرى من المعالجات الصغرية المستخدمة ضمن مجال WSN، وذلك بهدف دراسة أثر تغيير نوع المعالج ضمن عقدة الحساسات على الشبكة كلياً.
2- التركيز على بارامترات أخرى متعلقة بمعالج العقدة مثل استهلاك الطاقة المتعلق بالتعليمة الواحدة.
3- استكمال بناء العمليات الوظيفية ضمن نواة المنصة المصمَّمة، من خلال تطوير الوظائف المتعلقة بباقي طبقات الشبكة، للوصول إلى محاكي متكامل يُستخدَم ضمن مجال شبكات الحساسات اللاسلكية، ويأخذ بالحسبان تفاصيل عمليات المستوى المنخفض والبارامترات المتعلقة بها.

قائمة المراجع.

[15]	Abdul Satar, M., & Ishak, D. (2011, May,17- 19). "Application of Proteus VSM in modelling brushless DC motor drives". *Proceedings of 2011 4th International Conference on Mechatronics (ICOM)*. Kuala Lumpur, Malaysia.
[3]	Charfi, W., Masmoudi, M., & Derbel, F. (2009, March, 23- 26). "A layered model for wireless sensor networks". *6th International Multi- Conference on Systems, Signals and Devices*. Djerba. Tunisia.
[10]	Chen, M., Miao, Y., & Humar, I.(2019). OPNET IoT Simulation. *Introduction to OPNET Network Simulation*. Retrieved from https://www.springerprofessional.de/en/introduction- to- opnet- network- simulation/17637482.
[18]	Esber, Gh., Alkubeily, M., Sulaiman, S., & Mehrez, A.(2020, September). "Windows Form Application for Virtual Minimized Platform Kernel for Wireless Sensor Network Simulator". *Far East Journal of Electronics and Communications*, Vol. 23(Issue 2), No. 2, Pages 61- 71.
[12]	Hunt, B., Lipsman, R., Rosenberg, J., Coombes, K., Osborn, j., Stuck, G.(2001). *A Guide to MATLAB for Beginners and Experienced Users*. Cambridge, England. Retrieved from https://kkpatel7.files.wordpress.com/2014/10/a- guide- to- matlab.pdf.
[6]	Issariyakul, T., & Hossain, E.(2012). Introduction to Network Simulator NS2(2nd Ed). *Introduction to Network Simulator NS2*(pp.21- 23). New York, USA. Retrieved from http://www.mathcs.emory.edu /~cheung/ Courses/ 455/Syllabus/A3- NS/Book/Introduction- to- Network- Simulator- NS2- 2012.pdf.
[14]	Microchip Technology Inc. (2019). MPLAB X IDE User's Guide. *What is MPLAB X IDE?*(pp.21- 23). USA. Retrieved from http:// ww1. Microchip .com/ downloads/ en/ DeviceDoc/ 5000207E.pdf.
[11]	Mittal, S. (2012, September)."OPNET: An Integrated Design Paradigm for Simulations". *Software Engineering : An International Journal (SEIJ)*. Vol. 2, No. 2.
[4]	Nakov, S., Kolev, V., & Co.(2013). Fundamentals Of Computer Programming With C#. *Introduction to Programming*. Sofia, Bulgary. Retrieved from https://www.introprogramming.info/wp- content/uploads/ 2013/07/Books/ CSharpEn/

	Fundamentals- of- Computer- Programming- with- CSharp- Nakov- eBook- v2013.pdf.
[8]	Osterlind, F., Dunkels, A., Eriksson, J., Finne, N., & Voigt, Th. (2006, November, 14- 16). "Cross-level sensor network simulation with COOJA". *In Proceedings of the First IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp 2006)*. Tampa. Florida. USA.
[2]	September, 4- 6). "Wireless Sensor Network Applications: A Othman, M., & Shazali, Kh. (2012, Study in Environment Monitoring System". *International Symposium on Robotics and Intelligent Sensors 2012 (IRIS 2012)*. Kuching, Sarawak, Malaysia.
[21]	Polastre, J., Szeczyk, R., & Culler, D. (2005, April, 25- 27). "Telos: Enabling ultra- low power wireless research". *In Proceedings of IPSN/SPOTS*. Los Angeles, CA, USA.
[19]	Price, Ch. (1995, September). *MIPS IV Instruction Set* (3.2Ed.). Retrieved from https://www.cs.cmu.edu/afs/cs/academic/class/15740- f97/public/doc/mips- isa.pdf.
[22]	Sahoo, S., Mishra, P., & Satpathy, R. (2012, January)."Secure Routing in Wireless Sensor *Journal of Computer Science Issues*, Vol. 9(Issue 1), No 2. Networks". *IJCSI International*
[13]	Sanchez, J., & Canton, M. (2013, October, 28). *Microcontrollers: High- Performance Systems and Programming*(1st Ed.). Retrieved from http: //caxapa.ru /thumbs /646374/%5 BJulio_Sanchez_Maria_P._ Canton%5 D _Microco.pdf.
[17]	Sanderson, P. & Vollmar, K.(2006, March, 1- 5). "MARS: An Education- Oriented MIPS Assembly Language Simulator". *Proceedings Of The 37th SIGCSE Technical Symposium On Computer Science Education (SIGCSE '06)*. Houston. Texas, USA.
[5]	Shu, L., Wu, Ch., Zhang, Y., Chen, J., Wang, L., & Hauswirth, M. (2008, December, 13- 15). . *Second International* "wireless sensor networks "NetTopo: beyond simulator and visualizer for *Conference on Future Generation communication and Networking*. Hainan Island, China.
[1]	Singh, M., Amin, S., Imam, S., Sachan, V., & Choudhary, A. (2018, October, 12- 13). "A Survey of Wireless Sensor Network and its types". *International Conference on Advances in Computing, communication Control and Networking (ICACCCN)*. Greater Noida, India.
[20]	Texas Instruments Incorporated, TII.(2016, December). MSP430x1xx Family User's Guide. *RISC 16- Bit CPU* (pp.35- 109). Retrieved from https:// www. ti. com /litv/pdf/slau049f.
[7]	Varga, A., & Open Sim Ltd.(2016). OMNet++ Simulation Manual. *Overview*(pp.3- 5). Retrieved from https://doc.omnetpp.org /omnetpp / Simulation Manual.pdf.
[9]	Zahra, F., & Zaman, N.(2020, January)," Cooja Simulator Step by Step Manual *". In proceedings of Tylor University Conference*. Malaysia.
[23]	Zapata, M., Zilan, R., Ordinas, J., Bicakci, K., & Tavli, B.(2010, September). "The future of security Pages *Telecommunication Systems*. Vol.45(Issue 1), in Wireless Multimedia Sensor Networks". 77- 91.

[16]	Zhenwei, H., & Ke- fei, s. (2011, August, 12- 14)."Design of thermostat system based on Proteus simulation software". *Proceedings of 2011 International Conference on Electronic & Mechanical Engineering and Information Technology (EMEIT 2011)*. Harbin, China.