مجلة جامعة طرطوس للبحوث والدراسات العلمية _ سلسلة العلوم الهندسية المجلد (6) العدد(4) 2022
Tartous University Journal for Research and Scientific Studies - engineering Sciences Series Vol. (6) No. (4) 2022

برمجة ومحاكاة شبكات الحساسات اللاسلكية التي تعتمد نظام التشغيل TinyOS باستخدام TOSSIM

د. نعمى يونس*

(تاريخ الإيداع 2/1 /2022 . قُبِل للنشر في 2022/6/7)

□ ملخّص□

يعد نظام التشغيل الذي تعتمده عقدة التحسس اللاسلكية بمثابة أداة برمجية أساسية مستخدمة بهدف تطوير تطبيقات شبكات الحساسات اللاسلكية عموماً، كما يتحكم بعمل وحدة التحسس وآلية تطبيق بروتوكولات الشبكة وإدارة العتاد الصلب، من أجل برمجة أنظمة التشغيل تم استخدام لغات خاصة بها بهدف تسريع عملية تطوير وتحديث التطبيقات وكان من أكثرها شيوعاً TinyOS.

يعتمد TinyOS على مفهوم البرمجة المقادة بالحدث (event–based)، يتم من خلاله الوصول برمجياً إلى شريحة الراديو ووحدة التحسس وإدارة الاتصالات والذاكرة، كما يتم محاكاة الشبكة ومراقبة قيم وحدة التحسس واتصالات البيانات بين العقد باستخدام المحاكي TOSSIM الخاص بعقد التحسس التي تعتمد TinyOS.

لا تركز معظم أدوات محاكاة WSN على تطوير الجانب البرمجي لنظام التشغيل وتتعامل مع العقدة ككيان مغلق يدرج في سياق محاكاة الشبكة من أجل تقييم بارامترات الدراسة اعتماداً على معطيات أخرى في بيئة المحاكاة. بمقابل ذلك قدم هذا البحث المراحل التفصيلية المتعلقة بمحاكاة شبكة حساسات لاسلكية مستخدماً البرمجة الوظيفية لنظام التشغيل على مستوى العقدة الواحدة ومحاكاة العقد كشبكة متكاملة كمرحلة لاحقة، تم استخدم لغة Nesc من أجل برمجة وظائف TinyOS ضمن إطار تطبيقات التتبع والإنذار في WSN مع دراسة مراحل المحاكاة وتقييم النتائج من ناحية استجابة العقد بما يتيح المحاكي الخاص بهذا النظام.

كلمات مفتاحية: شبكات الحساسات اللاسلكية، نظام التشغيل، محاكي شبكات الحساسات اللاسلكية الخاص بشبكات الحساسات اللاسلكية التي تعتمد TinyOS، اللغة C المطورة لبرمجة الأنظمة المدمجة، طبقة الرسالة الفعالة، نظام الإنذار.

*عضو هيئة فنية_مشرف على الأعمال_ في كلية هندسة تكنولوجيا المعلومات والاتصالات قسم هندسة النظم الحاسوبية والالكترونية – جامعة طرطوس

مجلة جامعة طرطوس للبحوث والدراسات العلمية _ سلسلة العلوم الهندسية المجلد (6) العدد(4) 2022
Tartous University Journal for Research and Scientific Studies - engineering Sciences Series Vol. (6) No. (4) 2022

Programming and simulation of wireless sensor networks, which depend on TinyOS using TOSSIM

Dr. Eng. Noama Younes*

(Received 1/ 2/ 2022 . Accepted 7/ 6/ 2022)

□ ABSTRACT □

Operating System (OS) of wireless sensor node plays a crucial role in developing of wireless sensor networks applications, also controls on sensing unit and applying network protocols, management of hardware elements. We uses special languages for operating system programming in order to speed up development operation and update applications. TinyOS is lightweight operating system depending on event driven programming. it will be allowed to access to radio board and sensing unit and management of communication and memory using software library, also simulation of network and monitoring values of sensing unit and data communication between nodes using TOSSIM for TinyOS. Most of simulation tools in WSN do not focus on operating system programming side in sensor node and deal with it as closed entity which be used in network simulation and evaluating studying parameters depending on other data in simulation environment. What we have done in this article is studying operating system on single node side and in later stage simulating of nodes as integrated network, this article uses Nesc language for programming TinyOS within tracking and alarm applications in WSN with studying simulation stages and results evaluation.

Keywords: Wireless Sensor Network (WSN), Operating System (OS), TOSSIM, network embedded systemC (NeSC), Active Message Layer, alarm system.

*Technical Member. In ICTE Faculty- CESE Department – Tartous University.

1.مقدمة:

تعد شبكات الحساسات اللاسلكية (WSN) بمثابة التكنولوجيا الواعدة على كافة الأصعدة والبحوث. ركزت هذه التكنولوجيا على تطوير وحدات راديوية لاسلكية، ومتحكمات قليلة الاستهلاك للطاقة يكون بمقدورها العمل على تردد منخفض مقارنة بوحدات المعالجة التقليدية الحديثة، أيضاً فإنّها تستخدم الأنظمة الميكانيكية الالكترونية (MEMS) من أجل قياس المتغيرات الفيزيائية والكيميائية وجهد البطارية، إضافة إلى محدودية موارد الذاكرة بما يسمح لنا بتخزين عدة كيلوبايتات فقط من البيانات، يمكن الحديث عن شبكات الحساسات كتكنولوجيا من الممكن استخدامها في العديد من التطبيقات الهجينة التي تهم المجتمع كمراقبة البيئة، التحكم بالحركة، مراقبة الأبنية والجسور، تتبع الأشخاص والأهداف،... إلخ[1]. تستدعي محدودية الموارد المذكورة مسبقاً اعتماد أنظمة تشغيل بمتطلبات خاصة مثل [2] TinyOSو[3]Contiki ، وغير ذلك من نظم التشغيل المتاحة في WSN .

لقد كان الهدف الأساسي لجميع هذه الأنظمة هو ضمان مرونة العمل وموثوقيته والحفاظ على عقدة الحساس في نمط حفظ القدرة بما يتلاءم مع احتياجات التطبيق[2]، حيث التقليل من استهلاك الطاقة وزيادة زمن حياة البطارية هو أحد المطالب الأساسية لهذا النوع من الأنظمة، بدورها تختلف بيئة المحاكاة المستخدمة لدراسة شبكات الحساسات باختلاف نظام التشغيل المعتمد لدى عقدة الحساس، كما تختلف أيضاً نوعية الشرائح المكونة للعقدة وخصائصها كالوحدة الراديوية والذواكر وعناصر التحسس باختلاف نوع المنصة (Micaz, telosb,mica2,..)، [1,4].

تندرج الدراسة المرجعية المذكورة هنا وفق مستويين أساسيين متعاقبين يتمثل الأول بالإشارة إلى تطبيقات شبكات الحساسات اللاسلكية التي تعتمد نظام التشغيل TinyOS في عمليات الإنذار عن المخاطر عموماً، ثم ننتقل إلى مسار فرعي منبثق عن السابق يتعلق بمجال العناية الصحية والإبلاغ عن الحالات المرضية، وعلى صعيد الأبحاث المنجزة مسبقاً ضمن الإطار العام لتطبيقات الإنذار في شبكات الحساسات اللاسلكية، فقد اعتمدت هذه الأنظمة في البداية على الشبكة السلكية التي تعاني من عدم مرونة التخطيط والبناء، مع صعوبات أخرى مرافقة لعمليات الصيانة، وبهدف معالجة هذه الإشكالات تم اقتراح تصميم وتطوير أنظمة إنذار باستخدام شبكات التحسس اللاسلكية، حيث تم استخدام لغة Nesc في المرجع [4] من أجل برمجة نظام TinyOS 1.X ومن ثم محاكاة الشبكة باستخدام TOSSIM، إضافة إلى استخدام أدوات مساعدة للإظهار مثل TinyViz و JAVA، ومن أجل معالجة قاعدة المعطيات المجمعة استخدمت MySQL، والهدف هو تصميم نظام تتبع من أجل مراقبة مناطق الغابات باستخدام شبكات الحساسات اللاسلكية مع محاكاة الشبكة مسبقاً بهدف التأكد من سلامة أدائها الوظيفي، ينقسم البحث إلى مرحلتين تتمثل الأولى بتصميم برنامج تشغيل عقدة التحسس ويتم في المرحلة الثانية برمجة الأداء الوظيفي الأساسي المتمثل بتجميع المعلومات المتدفقة من عقد التحسس التي تتواصل لاسلكياً مع المحطة القاعدة.

كما تم في المرجع [5]، وكحل علاجي لمشكلة الحرائق وآثارها المدمرة على الممتلكات في حال عدم تداركها بشكل جيد استخدام عقد تحسس لاكتشاف حدوث الحريق وعقد أخرى للإنذار ولوحة تحكم للإنذار عن حدوثه، كما تم اعتماد برنامج LABVIEW من قبل لوحة التحكم المركزية التي تتبع خوارزمية الاكتشاف والتقييم والاكتشاف.

ضمن نفس سياق الإبلاغ عن الإخطارات تم في المرجع [6] استخدام شبكات الحساسات اللاسلكية من أجل تنفيذ شبكة مراقبة للخلايا الكهرضوئية بهدف حمايتها من عملية السرقة وذلك بسبب ارتفاع ثمنها، وهو ما يدعى نظام الإنذار المضاد للسرقة (anti-theft alarm system) حيث يتم وضع عقد التحسس المجهزة بحساس كاشف للاهتزاز يتحسس انزياح للوح الشمسي من موضع استقراره تحت سلسلة الألواح الشمسية، وذلك وفق نمط الشبكة

النجمية حيث تقوم العقدة السيد باستمرار بالاتصال لاسلكياً مع العقد التابع ليتم تجميع استجابات العقد والتواصل مع مركز تحكم من خلال وصلة RS-232، عندما تكتشف العقدة التابع خللاً من هذا النوع تقوم بإرسال عدة رسائل متعاقبة إلى مركز التحكم الذي يقوم بتشغيل إشارات أنظمة الإنذار المعنية.

بمقابل ذلك في المرجع [7] وانطلاقاً من آثار الكوارث الطبيعية وتأثيرها المدمر على الممتلكات والحياة البشرية وبهدف منع هذا الأذى الضخم الناتج عنها، تم تطبيق شبكات الحساسات اللاسلكية باستخدام معيار Zigbee/IEEE802.15.4 كشبكة محطة أرصاد جوية لإرسال معلومات الطقس وإنذارات الكوارث، يستثمر النظام المقترح إمكانية شبكات الحساسات اللاسلكية على إرسال الإشارات عبر مسافات شاسعة باستخدام الطبولوجيا المتشعبة لنقل المعطيات مع تخفيض استهلاك الطاقة وبناء على ذلك يتم إقامة هكذا أنظمة في مواضع من الصعب فيها توصيل العتاد الصلب أو إيصال التغذية الكهربائية.

على المستوى الآخر المتمثل بإطار العناية الصحية الذي يندرج ضمنه التطبيق المعتمد في هذا البحث، فقد تم العمل في المرجع [8] وانطلاقاً مما تتعرض له شبكات الحساسات اللاسلكية من عوامل تعطل تؤثر على وثوقية القيم الناتجة عن وحدات التحسس، فالقياسات الخاطئة في حالة العناية الطبية تستدعي إنذارات خاطئة الأمر الذي يتطلب تدخل غير ضروري لموظفي العناية الشخصية، وبناء عليه فإن تطوير آلية للتمييز بين الأحداث الطبية الحقيقية والإنذارات الكاذبة سوف يحسن من وثوقية أداء أنظمة مراقبة المرضى عن بعد وجودة الخدمات الصحية المقدمة من قبل WSN، وبناء عليه فقد تم تقديم منهجية جديدة لاكتشاف القيم الشاذة من خلال تحليل المعطيات البيولوجية الناتجة عن وحدات التحسس الطبية بما يهدف إلى التمييز الدقيق بين الإنذارات الحقيقية والكاذبة، حيث يتم توقع القيم الناتجة عن وحدات التحسس بناء على القيم المقاسة مسبقاً ثم مقارنتها مع القيمة الفعلية من أجل حالة مقاسة من الواقع، لاحقاً تم مقارنة قيمة التفاوت مع قيمة العتبة التي تكون متكيفة أتوماتيكياً لتحديد فيما إذا كانت قيمة التحسس شاذة، تم تطبيق هذه الطريقة على قاعدة معطيات حقيقية ومقارنتها مع الطرائق الموجودة مسبقاً، حيث أثبتت النتائج فعالية الطريقة المقترحة بتحقيق معدل كشف مرتفع وانخفاض المعدل الإيجابي الخاطئ.

أيضاً وضمن هذا الإطار تم في المرجع [9] تطبيق شبكة منزلية باستخدام شبكات الحساسات اللاسلكية لاكتشاف الخلل المتعلق بأمان الحياة المنزلية من خلال تقييم بارامترات الرطوبة والإضاءة والحرارة والوزن والتنقل، كما تم التركيز على التحسين الناتج عن استخدام أسلوب الملاحظة وإجرائيات التحكم ضمن إطار إدارة أنظمة مراقبة المنازل وضمان أمنها حيث يقدم اختباراً عملياً لاستخدام هذه الأدوات ضمن المنزل.

تم في هذا البحث تطبيق شبكة حساسات لاسلكية منزلية قابلة للصيانة الدورية بهدف المراقبة المستمرة لدرجة حرارة المرضى وتقديم الخدمات الطبية اللازمة من مركز الضمان الصحي في حال تجاوز عتبة الحدود المسموحة طبياً والتي تشكل خطراً على حياة المرض أو تعد كإنذار لإمكانية حدوث الوفاة.

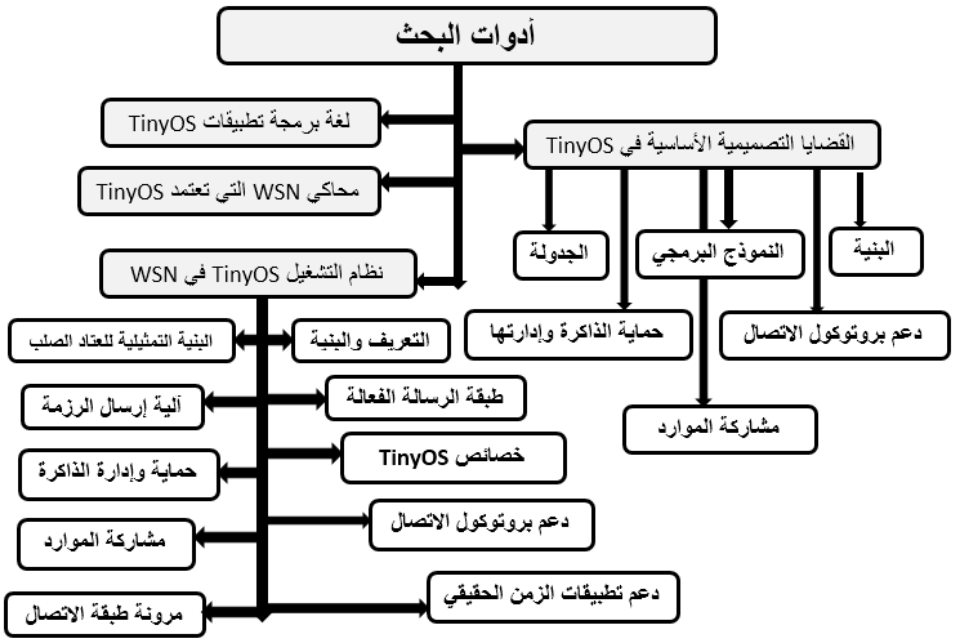
تم التركيز على TinyOS من أجل تنفيذ الفكرة المقترحة وذلك لأنه متاح للباحثين وشائع الاستخدام في المختبرات الأكاديمية كما استخدمت اللغة Nesc من أجل برمجة نظام التشغيل Tinyosبهدف محاكاة السلوك الوظيفي لعقدة التحسس اللاسلكية في منطقة توزع العقد(Sensor field)، وتم اختبار الشبكة وظيفياً باستخدام TOSSIM.

2.أهمية البحث وأهدافه:

على اعتبار أن عقدة التحسس هي نموذج للنظام المدمج (embedded system) الذي يتكون من ثلاث وحدات رئيسية (تحسس، معالجة، اتصال)، فإن عملية التحسس لبارامترات الوسط المحيط تعد كمهمة أساسية تنجزها عقدة الحساس بهدف تجميع البيانات ومن ثم إرسالها لاسلكياً إلى مركز المعالجة النهائي من أجل اتخاذ القرار المناسب والإجراءات العلاجية اللازمة، تنبثق أهمية البحث المنجز من أهمية تطبيقات المراقبة والتتبع ضمن مجال العناية الصحية مع تقديم محاكاة برمجية لشبكة حساسات لاسلكية مقترحة ضمن إطار المعالجة الطبية، تم نمذجة أداء العقدة باستخدام المقاطع البرمجية المزودة في مكتبة TinyOS لشرائح basicsb و mts300 و im2sb من أجل تنفيذ عمليات التحسس المطلوبة لبارامترات الوسط الفيزيائي كالضوء والحرارة ..إلخ من خلال الروابط الوظيفية المزودة لكل منها، أيضاً استخدام الروابط الوظيفية لوحدات الإرسال والاستقبال (Radio units) التي تختلف تبعاً لمنصة الحساس المستخدمة كالشريحة cc1512 في منصة Micaz و cc100 في منصة Mica2 لإنجاز عمليات التبادل على مستوى الشبكة. يركز البحث على جوانب التصميم البرمجي لنظام التشغيل TinyOS باستخدام اللغة Nesc على مستوى العقدة الواحدة ثم دراسة مراحل المحاكاة على مستوى الشبكة باستخدام TOSSIM مع تقييم نتائج كل مرحلة.

3.طرائق البحث ومواده:

يبين الشكل الآتي مخطط توزع الدراسة النظرية المستخدمة بهدف انجاز البحث.



الشكل(1): مخطط توزع الدراسة النظرية المستخدمة لإنجاز البحث

1.3. لغة برمجة تطبيقات TinyOS ((NesC) Network embedded systems C):

تعد هذه اللغة كامتداد للغة C تم تصميمها بهدف تسهيل بناء المفاهيم وتنفيذ التطبيق في TinyOS، تبنى تطبيقات الـNesC من خلال كتابة وتجميع العناصر (components)، أما العناصر فهي وحدات برمجية تتألف من الخصائص والتطبيق، كما يتم ربط خصائص الحالة بين العنصر المستخدم والعنصر المزود، يعرف الرابط الوظيفي كعلاقة ثنائية الاتجاه بين العناصر، ومن أجل تحقيق ارتباط بين عنصرين فإنه ينبغي على أحدهما أن يزود الارتباط

في حين ينبغي على الثاني أن يستخدمه وعندئذ يحصل ما يدعى بالارتباط "wiring"، تتضمن الارتباطات الوظيفية مفاهيم برمجية تدعى الأوامر والأحداث، إن العنصر الذي يزود الارتباط الوظيفي يقوم بتطبيق أوامره، ويتوجب على العنصر الذي يستخدم الارتباط أن يطبق الأحداث، عندئذ تتواصل العناصر فيما بينها من خلال طلب الأوامر وتفعيل الأحداث، تدعم NesC العديد من الميزات المستخدمة في TinyOS على نطاق واسع[10]:

1.العناصر العامة (Generic components): من الممكن الحصول على أمثال منها عدة مرات في التطبيق الواحد وتأخذ نماذج محدودة من البيانات أو الثوابت كمضامين لها.

2.المهام (Tasks): عبارة عن مقاطع برمجية يستغرق تنفيذها زمناً ملموساً ولكن تنفذ بناءً على جدولة مسبقة، يتم تنفيذ المهام المقحمة في الكود البرمجي لاحقاً بواسطة جدول الـTinyOS عندما يكون المعالج بحالة راحة.

3.المقاطع الأتوماتيكية (Atomic sections): تعد بمثابة مقاطع برمجية ذات حساسية عالية للزمن يتم تطبيقها عن طريق إلغاء تفعيل المقاطعات ولهذا السبب ينبغي عليها أن تكون قصيرة قدر الإمكان.

2.3.محاكي شبكات الحساسات الخاص بنظام التشغيل TinyOS (TOSSIM):

يحتاج الباحثون إلى دراسة وتقييم تطبيقات شبكات الحساسات اللاسلكية إما بواسطة عقد حساسات حقيقية أو أداة محاكاة معتمدة. تم في هذا البحث التركيز على استخدام بيئة محاكاة خاصة بنظام التشغيل Tinyos في WSN علماً أنه يوجد لكل نظام تشغيل محاكي خاص به، فمثلاً لدينا محاكي الـCooja الذي يعتمد نظام تشغيل contiki[3]، بالمقابل تكمن الفائدة الأساسية لمحاكي TOSSIM في معالجة الكود البرمجي المكتوب بلغة NesC لنظام TinyOS، يعتمد المحاكي TOSSIM (TinyOS Simulator) منطق الحدث المتقطع، ويحتفظ بترتيب الأحداث بناء على زمن التنفيذ في نسق الانتظار ثم يتم تنفيذها تدريجياً، يتمثل الحدث على مستوى المحاكاة بمقاطعة العتاد الصلب أو إرسال أو استقبال رزمة بيانات.

يعدّ TOSSIMمكتبة محاكاة يتم فيها كتابة سيناريو المحاكاة إما بلغة .Python (من أجل السماح بنموذج تفاعلي للمحاكاة) أو++C، يوجد العديد من النماذج الراديوية التي من الممكن لها أن تستخدم لمحاكاة سلوك الشبكة بما يتوافق مع الواقع الفعلي لحقل النشر مثل (CpmModelC)، والذي يستخدم خوارزمية مقاربة النمط (CPM algorithm) التي تعتمد على الاستخراج الإحصائي لقيم الضجيج الاختبارية كما أنه يضمن نموذج المحاكاة الأكثر دقة من خلال استثمار ميزات الضجيج المترابطة مع الزمن[11]. يتم باستخدامه مراقبة الاتصالات اللاسلكية بين عقد الحساسات، كما يقدم عدة نماذج من الاتصالات في شبكات الحساسات اللاسلكية.

3.3.القضايا التصميمية الأساسية في نظام التشغيل في WSN:

تتمثل القضايا الأساسية المتعلقة بتصميم نظام التشغيل في WSN على النحو الآتي:

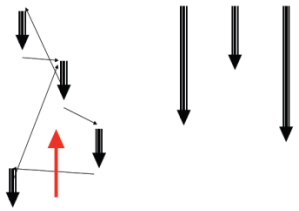
1.3.3.البنية (Architecture): تعرف البنية على أنها هيكلية البناء لنظام التشغيل والتي لديها تأثير كبير على حجم كيرنل (OS Kernel) وعلى الطريقة التي يزود بها الخدمات إلى برامج التطبيقات. نذكر هنا بعضاً من هذه البنى المعروفة في أنظمة التشغيل كالبنية الموحدة (monolithic architecture)، البنية كيرنل

المصغرة (micro-kernel-architecture)، بنية الآلة الافتراضية (Virtual machine architecture)، والبنية الطبقية (Layered architecture).

ليس لدى البنية الموحدة أي تركيبة خاصة، حيث يتم تطبيق الخدمات المزودة من قبل نظام التشغيل بصورة مستقلة، كما تزود كل خدمة رابط وظيفي من أجل الخدمات الأخرى، بالنسبة لبنية كيرنل المصغرة (Micro Kernel architecture) والتي يتم فيها تضمين كيرنل بآلية وظائفية مصغرة مما يعني تخفيضاً في حجمه، تعد الآلة الافتراضية بمثابة خيار بنيوي آخر يعتمد على استيراد بيئات افتراضية للمستخدم تمتلك خصائص العتاد الصلب التي يحتاجها والفائدة الأساسية هي قابلية النقل، لكن بالمقابل يحصل هنا انخفاض واضح في مستوى أداء النظام. وأخيراً تطبق بنية نظام التشغيل الطبقية خدماتها على شكل طبقات وتتجلى فوائدها في سهولة الإدارة والفهم وضمان الوثوقية إلا أنها تعد كبنية قليلة المرونة من منظور نظام التشغيل، نستنتج بناء على ما سبق أنه ينبغي على نظام التشغيل الخاص بشبكة WSN أن يستخدم البنية التي تنتج حجم صغير لكيرنل وتأثير صغير على الذاكرة كما يجب أن تسمح البنية بتطوير كيرنل في حال كان ذلك مطلوباً، يجب أن تكون البنية مرنة لكي يتم تحميل الخدمات المطلوبة للتطبيق على النظام التشغيل[12].

2.3.3.النموذج البرمجي (Programming model):

يمتلك النمط البرمجي المزود من قبل نظام التشغيل تأثيراً كبيراً على تطوير التطبيقات، يوجد لدينا نموذجين أساسيين هما: النموذج المقاد بالحدث (event driven) والبرمجة من نمط الـ(multithreading) كما هو موضح بالشكل(2) حيث يعد الأخير كآلية أساسية لتطوير التطبيق والأقرب للمبرمج لكنه بالمقابل يستهلك الموارد بشكل كثيف مما يجعله غير مناسباً للعناصر ذات الموارد المحدودة كعقد الحساسات، أما البرمجة المقادة بالحدث فإنها مناسبة أكثر لهذا النوع من العناصر لكنها بالمقابل لا تكون ملائمة لمطوري التطبيقات، وبناء عليه فقد ركز الباحثون اهتمامهم على تطوير نموذج برمجي من نمط multithreading ومخفف لأنظمة التشغيل الخاصة بـWSN، حيث تقدم العديد من أنظمة التشغيل الحديثة دعماً لهذا النوع من النماذج البرمجية[12].



الشكل(2): منطق الحدث المتقطع مقابل multithreading[14]

3.3.3.الجدولة (Scheduling): تحدد الجدولة الترتيب الذي سيتم وفقه تنفيذ المهام من قبل وحدة المعالجة المركزية (CPU)، إن الهدف الأساسي لعملية الجدولة في أنظمة الحاسوب التقليدية هو تقليل التأخير وزيادة السرعة وضمان التوازن في استخدام الموارد. إن اختيار خوارزمية الجدولة المناسبة فيWSN يعتمد على طبيعة التطبيق، ففي حالة تطبيقات الزمن الحقيقي يجب تطبيق خوارزميات الجدولة للزمن الحقيقي، أما من أجل تطبيقات الزمن غير الحقيقي فتكون خوارزميات الزمن غير الحقيقي كافية لتحقيق متطلبات التطبيق، وعلى اعتبار أنه يتم استخدام WSN في تطبيقات الزمن الحقيقي والغير حقيقي فإنه ينبغي على نظام التشغيل في WSN أن يزود خوارزمية الجدولة التي تلبي متطلبات التطبيق بحيث تكون فعالة من ناحية القدرة وآلية التعامل مع الذاكرة [12].

4.3.3 حماية الذاكرة وإدارتها(Memory Management and Protection) : يشير مصطلح إدارة الذاكرة في نظام التشغيل التقليدي إلى استراتيجية تخصيص وتحرير مواضع التخزين للعمليات و multithreading المختلفة، إن أكثر التقنيات شيوعاً والمستخدمة في إدارة الذاكرة هما الإدارة الساكنة للذاكرة والإدارة الديناميكية لها.

في حين تعد الإدارة الستاتيكية كتقنية بسيطة ومقيدة في حالة التعامل مع موارد ضخمة من الذاكرة، كما أنها ستنتج أنظمة تفتقر إلى المرونة بسبب عدم إمكانية تحقيق تخصيص لمواضع التخزين في الذاكرة بالتزامن مع فترة التشغيل، بمقابل ذلك فإن الإدارة الديناميكية للذاكرة ستنتج أنظمة أكثر مرونة وذلك بتخصيص وتحرير مواقع الذاكرة أثناء فترة التشغيل. يقصد بحماية الذاكرة طريقة التحكم بسماحية الوصول إلى ذاكرة الحاسب وتعد كجزء من بنى مجموعة التعليمات الحديثة وأنظمة التشغيل، لم يتم تطبيق استراتيجية من أجل إدارة الذاكرة في النسخ الأولى لأنظمة التشغيل في WSN حيث افترضت أنظمة التشغيل حينها أن وجود تطبيق وحيد فقط ينفذ على عقدة حساس واحدة لا يخلق حاجة ملحة لتطبيق آلية الحماية للذاكرة، ولكن مع ظهور تطبيقات جديدة لـWSN فقد أصبحت شبكات الحساسات الحديثة تقدم دعماً للmultithreading، مما يعني أخيراً أن مسألة إدارة الذاكرة أصبحت حالياً قضية أساسية في أنظمة تشغيل الحساسات[2].

5.3.3 دعم بروتوكول الاتصال (Communication Protocol Support): يعبر مصطلح الاتصالات في بيئة نظام التشغيل عن آلية الاتصالات المتبادلة بين العمليات ضمن النظام الواحد ومع العقد الأخرى في الشبكة أيضاً. تضمن أنظمة التشغيل في WSN رابطاً برمجياً للتطبيق (API) لتمكين برنامج التطبيق من تنفيذ عمليات التواصل، وعلى اعتبار أنه من الممكن لعقد الحساسات أن تكون هجينة من ناحية الموارد، فإنه ينبغي على بروتوكول الاتصال المزود من قبل نظام التشغيل أن يأخذ ذلك بعين الاعتبار وأن يقدم خدمات خاصة بالنقل والتشبيك وتنظيم الوصول إلى الوسط (MAC) من أجل الاتصالات الممكنة في الشبكة [12].

6.3.3 مشاركة الموارد (Resource Sharing): تعبر عن مشاركة الموارد وتخصيصها كما لديها تأثير كبير على آلية تنفيذ عدة عمليات معاً بصورة متزامنة. بمقدور معظم أنظمة التشغيل في WSN أن تزود حالياً نوعاً من multithreading المتزامن مما يتطلب تقنية مناسبة لتنظيم عملية مشاركة الموارد، كما قد نحتاج في بعض الحالات إلى تأمين وصول متسلسل إلى الموارد من خلال استخدام البدايات المتزامنة.

وأخيراً لابدّ أن نشير إلى أنه يتوجب على نظام التشغيل أن يدعم بروتوكول طبقة النقل الذي يلبي متطلبات العمل في الزمن الحقيقي بحيث يضمن المراقبة المستمرة لظروف الشبكة، وأن يخفض من احتمال حصول الاحتقان فيها بحيث يحقق تتدفق البيانات في الزمن الحقيقي مستوى الجودة المطلوب[13]. وإلى أبعد من ذلك يحتاج نظام التشغيل أن يضمن تطبيق بروتوكول التوجيه الذي يبني مساراته بناء على متطلبات جودة الخدمة للتطبيق، كما يجب أن يدعم خوارزمية الـMAC التي تجدول الرزم بناء على درجة الأولوية [14].

4.3.نظام التشغيل TinyOS في شبكات الحساسات اللاسلكية

1.4.3. التعريف: يعد TinyOS كنظام تشغيل مفتوح المصدر مخصص لتطبيقات شبكات الحساسات اللاسلكية [2]. يتصف بالمرونة ويعتمد مفهوم العنصر (component)، كما أنه نظام تشغيل يعتمد البرمجة

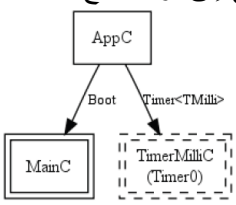
المقادة بالحدث باستخدام لغة معدلة عن اللغة C تدعى nesC (network embedded systemC)، يحقق TinyOSإمكانية التحكم والنفاذ إلى وسط الاتصال و EEPROM ومكونات العتاد الصلب الأخرى.

يدعم TinyOS تنفيذ عدة برامج بصورة متزامنة مع تأثير محدود على مساحة الذاكرة وقابلية للتطوير والتوسع بناء على العناصر المختلفة، يمتلك TinyOS مكتبة للعناصر الخاصة به، والتي تتضمن العناصر الأكثر استخداماً كبروتوكولات الشبكات أو عناصر التحكم بالحساسات. يعمد التطبيق إلى ربط العناصر التي يحتاجها من خلال خصائص الربط البيني وبذلك يكون بمقدورنا اختبار نظام تشغيل مصغر يتكون فقط من تلك العناصر التي تحتاجها التطبيقات فعلياً، لا يعد TinyOS نظام التشغيل الوحيد المطبق والمعمول به في شبكات الحساسات اللاسلكية، ولكنه الأكثر شيوعاً في المجال الأكاديمي، فهو معتمد من قبل آلاف المطورين عبر العالم، ويدعم العديد من منصات الحساسات. تم إصدار النسخة TinyOS–1.0 في أيلول عام 2002، وبهدف معالجة العديد من المشاكل التي كانت في الجيل الأول، فقد تم إصدار النسخة الثانية TinyOS–2.X عام 2006 [15].

2.4.3. بنية TinyOS: يندرج TinyOS ضمن تصنيف البنية الموحدة، وتبعاً لمتطلبات التطبيق يتم تجميع العناصر المختلفة معاً بصورة منتظمة لتحديد أداء عقدة الحساس عقب عملية تهيئة المنظومة البرمجية. إن العنصر هو الكينونة المستقلة التي تزود واحداً أو أكثر من الروابط الوظيفية على هيئة الأوامر والأحداث والمهام حيث تستخدم الأوامر والأحداث لتحقيق التواصل فيما بين العناصر في حين تستخدم المهام للتعبير عن التزامن ضمن العنصر الواحد، فالأمر هو طلب تنفيذ خدمة معينة بينما يشير الحدث إلى انتهاء تلك الخدمة، هنالك دائماً ما يسمى عنصر الإعداد عالي المستوى (Configuration Component) والذي نعرف فيه مجموع العناصر التي سنستخدمها، حيث يتضمن عادة عنصر(MainC) الذي يعد مسؤولاً عن تهيئة العتاد الصلب وتسلسل تهيئة العناصر(boot sequence) وعنصر التطبيق الأساسي.

يضاف إلى ذلك أنّه بمقدور العنصر عالي المستوى تهيئة عناصر الخدمة التي يحتاجها مثل الحساسات، والمؤقتات، وخصائص الراديو وربطهم بعنصر التطبيق. إن العناصر المجمعة عند مستوى الإعداد الأعلى بمقدورها أيضاً أن تحقق تهيئة العناصر الأخرى التي نحتاجها، ويكون ذلك بمثابة مسار من العنصر عالي المستوى إلى أي عنصر آخر مستخدم، يسمح لنا ذلك باختبار نظام مصغر يشمل فقط العناصر التي تم استدعاؤها[16].

وكما نلاحظ من المثال الآتي الموضح في الشكل(3) لعنصر الإعداد المبسط الذي يعرف ثلاث عناصر مستخدمة هي MainC، AppC، ومثل من العنصر العام ()TimerMilliC والذي تمت تسميته بـTimer0، أيضاً يعرف هذا الإعداد بأن أي عنصر AppC يجب أن يستخدم الرابط الوظيفي المزود من قبل عنصر MainC والرابط Timer0 المزود من قبل Timer0. إن ما يحصل عندما ننتهي من عملية البرمجة هو إعادة التهيئة لعقدة الحساس وبعدها تبدأ عقدة الحساس بتنفيذ تابع ()Main المطبق في العنصر RealMainP والذي يعد مسؤولاً عن تسلسل التهيئة في TinyOS، فالعناصر التي تحتاج إلى ارتباط مع تسلسل التهيئة تحتاج أيضاً أن ترتبط بعنصر MainC.



الشكل(3): بنية الإعداد المبسطة في TinyOS [15]

تنفذ هذه العملية وفق مراحل أساسية هي (تهيئة الجدولة، تهيئة المنصة، تهيئة العتاد البرمجي، بداية الحلقة الرئيسية)، أما بالنسبة للتزامن فهناك نوعان من المقاطع البرمجية في TinyOS المتزامن والغير متزامن، تنتج الوظائف البرمجية الغير متزامنة عن مقاطعات العتاد الصلب ويتم تنفيذها بطريقة تنافسية، بمقابل ذلك فإن المقاطع المتزامنة سيتم الوصول إليها فقط من خلال ما يعرف بالمهام (Tasks) حيث لا تتنافس المهام في آلية التنفيذ مع بعضها ويتم تنفيذها أتوماتيكياً حسب ترتيبها بالنسبة للمهام الأخرى ضمن نسق الجدولة الخاصة بالمهام[15].

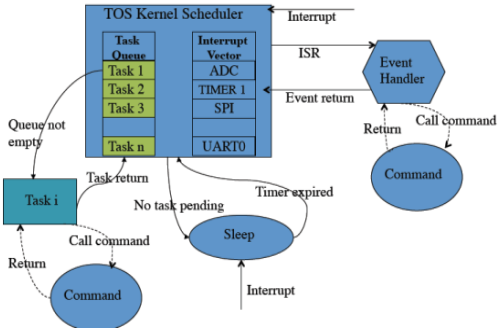
3.4.3.البنية التمثيلية للعتاد الصلب في TinyOS(Hardware Abstraction Architecture in TinyOS): تستخدم البنية التمثيلية للعتاد الصلب (HAA) في TinyOS بهدف تسهيل عملية نقل التطبيقات بين منصات الحساسات، يصنف التمثيل البرمجي للعتاد الصلب وظائفياً في ثلاث طبقات منفصلة تمتلك كل طبقة مسؤولياتها الخاصة بها. تعتمد في ذلك على الروابط الوظيفية المزودة من قبل الطبقات الأدنى والتي تكون خاصة بكل نوع من منصات الحساسات، وتزود خدماتها لمعالجة العتاد الصلب في الطبقات الدنيا، تستخدم عندئذ الطبقة العليا المستقلة عن العتاد الصلب من قبل التطبيقات بصورة مباشرة [16].

4.4.3 طبقة الرسالة الفعالة (Active Message Layer): تعد طبقة الرسالة الفعالة كالطبقة الأساسية المستقلة للعتاد الصلب (Basic Hardware Independent Layer). تزود كل منصة حساس عنصر الرسالة الفعال الخاص بها (Active Message Component) حيث يتم تحقيق وظائف الاتصالات عبر هذه الطبقة من خلال أربعة عناصر، تأخذ هذه العناصر كمضمون لها ما يدعى معرف الرسالة (AM type) وهي: AMSenderC, AMReceiverC,AMSnooperC، AMSnoopingReceiverC، الذي يميز بين الرزم التي تتعلق بتطبيقات مختلفة كما تسمح لهذه التطبيقات بالاتصال بالقنوات المختلفة[17].

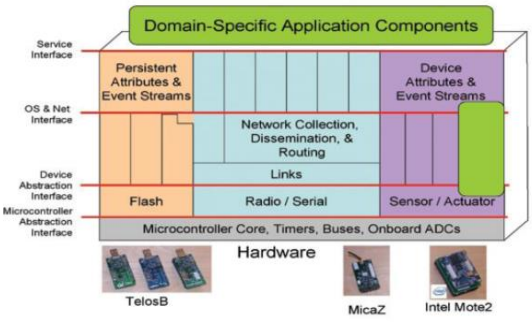
5.4.3 آلية إرسال الرزمة: يقوم العنصر CC1512transmitP بتنفيذ خوارزمية CSMA في التطبيق الخاص بالوحدة الراديوية CC1512 وذلك باعتبار أن منصة Micaz هي المستخدمة، كما أن طبقة الـCSMA هي فقط من يقدم تأخيرات راديوية عشوائية حيث لا تسمح خوارزمية CSMA أن يتم المباشرة بعملية الإرسال طالما أن القناة مشغولة. تمتلك شريحة CC1512 رجلاً خاصة لتحقيق هذا الغرض تدعى CCA Pin تكون الرجل CCA (Clear Channel Assessment) بالمستوى المرتفع فقط إذا كانت العقدة لا تستقبل بيانات مفيدة وشدة الإشارة المستقبلة هي أقل من عتبة التحسس للحامل والمبرمجة مسبقاً، حيث من الممكن برمجة عتبة التحسس للحامل من خلال مسجلات RSSI.CCA_THR (القيمة العامة هي 77db−) وقيمة MDMCTRLO.CCA_HYST (القيمة العامة هي 2 db)، وبالتالي فإن عتبة التحسس للحامل الناتجة تعرف كـCCA_HYST–CCA_THR، وهكذا سيتم اعتبار القناة خالية عندما تكون شدة الإشارة المستقبلة هي أقل من 79db− والعقدة ليست بحالة استقبال للبيانات المفيدة[18].

6.4.3 خصائص TinyOS: لم تقدم النسخ الأولى لـTinyOS أي دعم للتنفيذ المتزامن (multithreading)، في حين اقتصرت على تطوير التطبيق بناء على نموذج البرمجة المقاد بالحدث، يوضح الشكل (4) البنية الطبقية المفصلة له. بمقابل ذلك قدم TinyOS 2.1 دعماً لذلك بما يدعى بـ(TOS multithreading)، لقد أوضح الباحثون في المرجع [19] أن محدودية الموارد لدى عقدة الحساس تجعل من نظام التشغيل المقاد بالحدث أكثر تزامناً وبالرغم من ذلك فإن تطبيق تقنية multithreading تجعل المخطط

البرمجي أكثر ملاءمة، تضمن حزمة /TOS threading/ سهولة النظام البرمجي المعتمد على تقنية (threads) إضافة إلى فعالية نموذج Kernel المقاد بالحدث كما يوضح الشكل الآتي(5) بنية كيرنل الخاصة بـTinyOS.



الشكل(5): نموذج كيرنل الخاص بالنظام TinyOS[20]



الشكل(4): بنية TinyOS[20]

7.4.3 حماية وإدارة الذاكرة: تم الحديث في المرجع [21] عن أمن الذاكرة الفعال حيث لا يكون ممكناً ضمان حماية العتاد الصلب للذاكرة في عقد الحساسات محدودة الموارد، إضافة إلى استخدام لغة منخفضة المستوى وغير آمنة مثل Nesc. بمقابل ذلك استخدم TinyOS2.1 تقنية الذاكرة المحمية المدمجة معه على اعتبار أن أهداف الحماية تنحصر بالتقاط أخطاء المصفوفات والمؤشرات إضافة إلى تقديم تقنيات التشخيص المفيدة واستراتيجيات التغطية الممكنة، يستفيد TinyOS من مفهوم الوكيل (Deputy) في تطبيق منطق الذاكرة المحمية، وهو عبارة عن مترجم من مورد إلى مورد بما يضمن حماية للذاكرة ونوعها لأجل الشيفرة البرمجية المكتوبة بلغة C++، يكون TinyOS الآمن متلائماً مع النسخ الأولى لـTinyOS، يعتمد TinyOS الآمن على اقحام عبارات التفحص ضمن شيفرة التطبيق لضمان الأمان أثناء عملية التنفيذ وذلك بهدف تطبيق الأفعال الاحتياطية عند حدوث الأخطاء، كما يستخدم TinyOS طريقة الإدارة الساكنة للذاكرة.

8.4.3 دعم بروتوكول الاتصال: دعمت النسخ الأولى لـTinyOS نموذجان من بروتوكولات الاتصالات متعددة القفزات هما النشر و[22,23]TYMO، حيث يحقق بروتوكول النشر تسليم مضمون للبيانات إلى كل عقدة في الشبكة من خلال تمكين المسؤولين عن الشبكة من إعادة تنظيم الطلبات وإعادة برمجة الشبكة. أيضاً يعد TYMO كتطبيق للبروتوكول DYMO (بروتوكول التوجيه لشبكات الـAdHoc المتنقلة)، يتم في TYMO تعديل صيغة الرزمة كما يتم تطبيقه على مستوى أعلى من طبقة الرزمة الفعالة، تحدث line et al في المرجع [24] عن بروتوكول جديد لنشر البيانات يدعى DIP، إن DIP هو بروتوكول لاكتشاف البيانات ونشرها يكون متلائماً مع مئات القيم المتحسسة ومخصص لشبكات الحساسات اللاسلكية، حالياً يدعم TinyOS 2.1.1 البروتوكول 6LowPan وطبقة التشبيك IPV6 ضمن شبكات TinyOS، ومن أجل طبقة الـMAC يقدم TinyOS دعماً لتطبيقات البروتوكولات التالية: بروتوكول TDMA ذو القفزة الواحدة، البروتوكول الهجين TDMA/CSMA,B-MAC، أيضاً تطبيق الاختيار لطبقة الـMAC المتلائمة مع المعيار IEEE802.15.4.

9.4.3 مشاركة الموارد: يستخدم TinyOS تقنيتين أساسيتين لإدارة الموارد المتشاركة هما الأحداث المنتهية والأحداث المجردة (Virtualization and Completion events). يعد المورد المجرد كمثل مستقل (يستخدمه التطبيق بصورة مستقلة عن التطبيقات الأخرى)، أما الموارد التي لا يمكن لها أن تكون مجردة سيتم التعبير عنها من خلال الأحداث المنتهية، بدورها تعد طبقة الاتصالات العامة لـTinyOS كمورد مشترك بين العديد من multithreading ولا يمكن لها أن تكون مجردة، بمقدور طبقة الاتصالات العامة أن ترسل رزمة واحدة فقط في لحظة

ما حيث تفشل عمليات الإرسال للthreads الأخرى في تلك اللحظة. يتم قيادة الموارد المشتركة من خلال الأحداث المنتهية التي تعلم من خلالها multithreads المنتظرة عن انتهاء تنفيذ المهمة الحالية[20].

10.4.3 دعم تطبيقات الزمن الحقيقي: لا يقدم TinyOS أي دعم من أجل تطبيقات الزمن الحقيقي حيث يتم تنفيذ المهام في TinyOS وفق طريقة FIFO، وبناء عليه لا يعد TinyOS خياراً جيداً لشبكات الحساسات التي يتم نشرها لمراقبة الظواهر في الزمن الحقيقي. تم الحديث لاحقاً عن تطبيق العملية وفقاً لخوارزمية الجدولة EDF (Earliest Deadline First) والتي أصبحت متاحة للتطبيق في النسخ الأحدث من TinyOS. مع هذا لا تحقق الخوارزمية EDF المتطلبات المناسبة لعملية تنفيذ المهام، إذاً لا يعد TinyOS الخيار الجيد لتطبيقات الزمن الحقيقي، كما لا يقدم TinyOS أية تطبيقات لبروتوكول محدد في طبقات النقل والشبكة والMAC التي تدعم متطلبات جودة الخدمة لتدفق معلومات الوسائط في الزمن الحقيقي. يدعم TinyOS من أجل طبقة الـ MAC تقنية TDMA الأمر الذي يجعله خياراً مناسباً فيما يخص متطلبات التطبيق لدعم تدفق البيانات في تطبيقات الوسائط المتعددة. هنالك بعض الخصائص الإضافية التي يمتلكها نظام TinyOS من ناحية دعمه لنظام ملفات وحيد المستوى وذلك بناء على الاقتراض القائم بأن هنالك تطبيق واحد سيعمل مع عقدة الحساس عند أي لحظة زمنية معطاة، وبسبب محدودية موارد الذاكرة فإن نظام ملفات وحيد المستوى سيكون كافياً من أجل ذلك[20].

وأخيراً لابد من الإشارة إلى خصائص TinyOS من ناحية دعم قواعد البيانات فعلى اعتبار أن مهمة عقدة الحساس تتركز بتنفيذ عمليات التحسس والحساب والتخزين وإرسال البيانات فإن TinyOS يقدم دعماً لقواعد البيانات وفقاً للصيغة TinyDB[25]. أما دعم الأمن: فالمطلوب دائماً ضمان أمن الاتصالات في وسط البث اللاسلكي، بدوره يقدم TinyOS الحل في هذا الصعيد وفق TinySec[26]. دعم المحاكاة: يدعم TinyOS المحاكاة وفقاً للمحاكي TOSSIM حيث يتم كتابة شيفرة المحاكاة بلغة nesc وبالتالي من الممكن تطبيقها مع حساسات حقيقية [27]. دعم اللغة: يدعم TinyOS تطوير التطبيق بلغة nesc والتي تشابه إلى حد ما لغة C. دعم منصات الحساسات: يدعم TinyOS منصات التحسس الآتية Mica[28],MicaZ[28],Mica2[28] Tmote[29],TelosB[29], ، كما من الممكن الحصول على شرح مفصل حول TinyOS في المرجع [30].

11.4.3.مرونة طبقة الاتصال:

قدمت معظم أنظمة التشغيل العديد من الروابط الوظيفية في الاتصالات التي تسمح للتطبيقات بإرسال واستقبال البيانات. تستخدم هذه APIs بروتوكول الاتصال المزود من قبل نظام الشغيل ولكن ليس من الممكن اعتماد بروتوكول موحد لتقديم خدمات الاتصالات في شبكات الحساسات اللاسلكية حيث يقدم كل نظام تشغيل تطبيقه الخاص لبروتوكول الاتصال، وبناء عليه من غير الممكن تحقيق الاتصال بين عقدتي حساس لديهما نظامي تشغيل مختلفين. قدم Contiki سابقاً تطبيقاً لـIP6 المصغر(micro IPv6 stack) في شبكات الحساسات اللاسلكية، حيث يمكن ذلك عقد الحساسات من استخدام عناوين IP والتواصل مع الأجهزة الأخرى القائمة على نموذج العنونة هذا، إضافةً إلى الطريقة المتبعة في Contiki قدم TinyOS في نسخته الحديثة دعماً لموضوع العنونة IP التي تخص المعيار 6LowPan. إن تطبيق البروتوكول IP المصغر يسمح لعقدتي

حساس تستخدمان نظامي تشغيل مختلفين من التواصل مع بعضهما، ومع هذا تفتقر هذه الطريقة إلى إمكانية التعامل مع تطبيقات الزمن الحقيقي[31].

4. النتائج والمناقشة:

بهدف تتبع عمليات تحسس البيانات وإيصال الرزم ما بين عقد الحساسات ضمن إطار التطبيق المفترض فإننا سنقوم بتطبيق السيناريوين الآتيين:

- **السيناريو الأول:** تضمن تطبيقاً لعملية الإرسال الدوري إلى رأس العنقود للتأكد من حالة العقد التابعة.
- **السيناريو الثاني:** تم هنا تحقيق تتبع للقيم المقاسة من قبل وحدات التحسس وإرسال إنذار إلى قائد العنقود في حال تجاوزها قيمة عتبة محددة برمجياً بما يتناسب مع معطيات تطبيق المراقبة الطبية.

1.4 بيئة المحاكاة (Simulation Environment): تم إجراء عملية المحاكاة على شبكة مؤلفة من11 عقدة تنتشر عشوائياً ضمن مساحة مربعة تمثل حقل النشر، وتم افتراض أن العقد ثابتة أثناء فترة المحاكاة. يتم تحديد زمن المحاكاة 100 sec في ملف البايثون الخاص بقيادة عملية المحاكاة والذي يتضمن محددات الشبكة المطبقة من أجل المحاكاة، كما تم تطبيق السيناريو باستخدام نموذج الشبكة المحدد بالملف النصيnetwork.txt ومن ثم تم إدخال قيم الضجيج المحددة في الملف النصي meyer_heavy.txt أثناء مرحلة توليد نموذج قنوات الاتصال، اعتمدنا في انجاز المحاكاة على المنصة MicaZ، تم تقديم عائلة MICA كعقد حساسات في عام 2001، وكان آخر تحديث لهذه العائلة منصة MICAZ التي تستخدم الشريحة الراديوية CC1512 والتي تدعم معيار 802.15.14 وبروتوكولات ZigBEE مع تحسين معدل النقل إلى 250kps مقارنةً مع 190kbs لأجل MICA2، كما طبقت خوارزمية CSMA من أجل تنظيم آلية الوصول إلى وسط الإرسال، استخدمنا أيضاً python 2.5.1 لإدارة المحاكاة و GCC 4.1.2 بهدف تنظيم عملية الارتباط مع الملفات الأخرى.

2.4 مراحل تنفيذ المحاكاة في TOSSIM: تتمثل الفائدة من شرح الخطوات الآتية بهدف تفصيل مراحل عمل المحاكي TOSSIM بما ينسجم مع عنوان هذا البحث.

1. كتابة مخطط الـXML: إن الخطوة الأولى في عملية بناء مكتبة الTOSSIM (TOSSIM.so) هي استخدام محول Nesc لتوليد مستند XML (app.xml) والذي يوصف التطبيق من ناحية اسم ونوع كل متغير فيه.

2. تنفيذ تطبيق TinyOS: إضافة إلى تضمين خطوات التنفيذ اللاحقة فإن الخيار /sim/ يقوم بتبديل مسارات التضمين للتطبيق، فإذا كان التطبيق يمتلك التضمينات Ia-/Ib-/Ic- عندئذ يعمد الخيار sim إلى تحويل القائمة السابقة إلى Ia-Ib-Ic-I%T/lib/tossim Ia-/sim Ib-/sim Ic-/sim-. حيث يعني ذلك أن تطبيقات المحاكاة الخاصة بأي نظام سيتم استخدامها أولاً متبوعة بتطبيقات الTossim والتي تكون أيضاً متبوعة بالتطبيقات المعيارية يمرر الخيار (sim) مجموعة المضامين إلى المترجم وبالتالي فإنه يقوم بالتعريف بها لدى المترجم من أجل المحاكاة، إن نتيجة هذه الخطوة هي ملف الهدف (sim.o) الذي يوجد في الدليل الخاص بمنصة الحساس، يحتوي هذا الملف مجموعة من التوابع الخاصة باللغة C التي تختص بإعداد المحاكاة والتحكم بتنفيذها.

3. تنفيذ الروابط البرمجية: يتم لاحقاً تنفيذ الروابط الوظيفية الداعمة بلغة البايثون و C++. يبنى الارتباط الوظيفي باللغة بايثون التي تستدعي بدورها TOSSIM من خلال الارتباط باللغة C، يحتوي tossim.o الملف

البرمجي بلغة C++ في حين يتضمن الملف pytossim.o الدعم القائم بلغة البايثون، يجب تنفيذ هذه الملفات بصورة منفصلة وذلك لأن لغة C++ لا تفهم لغة Nesc و Nesc لا تفهم C++.

4. بناء الهدف المشترك: إلى جانب الخطوة الأخيرة يتم بناء المكتبة المشتركة التي تحتوي الكود البرمجي لـTOSSIMومزود الـC++ و البايثون، وقد تم استخدام python 2.5-dev لتجنب حدوث الأخطاء عند تنفيذ الربط مع الملف المشترك.

5. نسخ مزود البايثون: وأخيراً يتم نسخ الكود البرمجي باللغة بايثون والذي كان قد استدعى الملف المشترك الموجود في الدليل lib/tossim حيث يتم بناء على هذه العملية نسخه إلى الدليل المحلي.

3.4.خصائص الـTOSSIM: هناك العديد من النماذج الراديوية المستخدمة لمحاكاة سلوك الشبكة، إن بعضاً من هذه المجلدات يكون موجوداً في مجلد TinyOS (النموذج الراديوي البسيط، نموذج التداخل الثنائي، نموذج CPM).

1.3.4 طبقة الرسالة الفعالة: تعد بمثابة الطبقة العليا للبنية الراديوية لـTOSSIM، إنها تقدم خدمات الاتصال الفعالة للطبقات العليا وترتبط أيضاً بنموذج الشبكة في الطبقة الأدنى. تتجلى مسؤولية هذه الطبقة في تضمين محتوى الأجزاء الرئيسية من الرسائل المغادرة (عنوان المصدر، عنوان الهدف، الإعلام عن الرسائل القادمة إلى الرابط الوظيفي الصحيح).

2.3.4 نموذج طبقة الرزمة (Packet Layer Model – TossimPacketModelC): يؤمن النموذج الراديوي على مستوى الرزمة (Packet Gain Model) تطبيق وظيفة خوارزمية CSMA، فعند المباشرة بعملية إرسال الرزمة تنفذ هذه الطبقة خوارزمية CSMA الأساسية وتمرر الرسالة إلى الطبقة الأدنى في حال كانت القناة خالية فقط وتحفظ الحالة فيما إذا لازالت عقدة الحساس بصدد إرسال رزمة البيانات حالياً. أما عندما يشير نموذج الربح الراديوي إلى حصول استقبال للرزمة، عندئذ تتفحص TOSSIM Packet ModelC العقدة فيما إذا كانت في حالة إرسال لرزمة أخرى، وفي حال عدم صحة ذلك فإنها توجه الرزمة إلى الطبقات العليا وإلا تقوم بإسقاطها حيث لا تعد عملية التفحص هذه كافية لتحديد إمكانية تطبيق عملية إرسال رزمة أخرى بالدقة المطلوبة[18].

3.3.4 نموذج الربح الراديوي (Radio Gain Model): ينجز النموذج الراديوي مهمتين أساسيتين تتمثل الأولى بأخذ الرسالة التي تم إرسالها إلى الوسط الراديوي وجدولة أحداث الاستقبال لكل عقدة في الشبكة. أما الثانية: فعند انتهاء عملية الإرسال فإنه يطبق أحداث الاستقبال المجدولة ويقرر فيما إذا كان سيتم استقبال الرسالة وإبلاغ الطبقات العليا بحصول حدث الاستقبال هذا[18].

4.4.سيناريوهات المحاكاة والنتائج: تم تنفيذ القسم العملي اعتماداً على شبكة نجمية وفق منطق (السيد–التابع) ، تمثل العقدة ذات المعرف (0) كعقدة سيد في حين بقية العقد تكون بمثابة التابع لها، ترتبط العقدة السيد بمركز الرعاية الصحية المعني بتقديم الإجراءات الصحية العاجلة عند حاجة المرضى بناء على الإنذارات الواردة. يتم تنفيذ التطبيق العملي وفق مرحلتين يتم في الأولى اختبار البث الدوري باتجاه قائد العنقود للتأكيد على سلامة العقد وإمكانية التواصل اللاسلكي مع قائد العنقود، لم يتناول البحث جانب استهلاك الطاقة حيث لا يعد كقضية ملحة من أجل هذا التطبيق طالما أن الشبكة منزلية تخضع لصيانة دورية مع إمكانية

استبدال بطارية العقدة، لاحقاً تم في المرحلة الثانية تحسس درجة حرارة المرضى ضمن إطار تطبيق الرعاية الصحية والإنذار عن الحالات الخطرة منها:

1.4.4 السيناريو1 مع تقييم النتائج: استخدم المؤقت الزمني الذي يتم التحكم به من قبل المزود البرمجي TimerMilliC() من أجل تنظيم عملية الإرسال الدوري باتجاه قائد العنقود (Sink) وفق فواصل زمنية منتظمة مع إضاءة ليدات اختبار تكون منصة التحسس مجهزة بها. يزود العنصر LedsC وظائف التحكم بها بهدف الإشارة إلى جاهزية التطبيق للعمل، إضافة إلى استخدام الارتباطات الخاصة بطبقة الشبكة من أجل عمليات الإرسال والاستقبال كما هو موضح في الشكل (6).

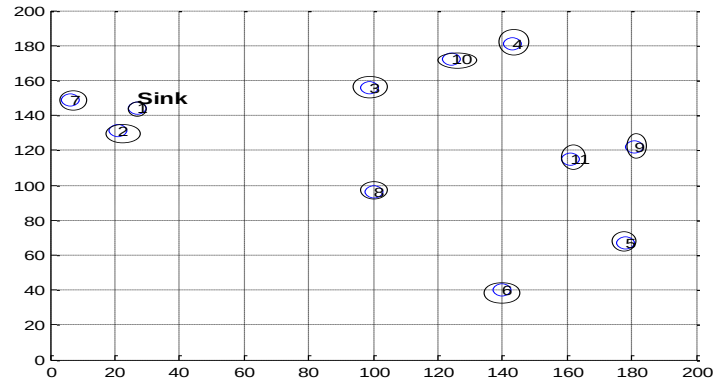
```
uses {
  interface Boot;
  interface SplitControl;
  interface Timer<TMilli> as MilliTimer;
  interface AMSend;
  interface Receive;
  interface Leds;
}
```

الشكل(6): الروابط البرمجية المدرجة في التطبيق

```
0   1   -72
1   0   -72
0   2   -72
2   0   -72
0   3   -72
3   0   -72
0   4   -72
4   0   -72
0   5   -100
5   0   -100
0   6   -100
6   0   -100
```

الشكل(7): معطيات طبقة الشبكة المعتمدة في المحاكاة

كما يوضح الشكل (7) العلاقة بين عقد الشبكة مع مقدار جودة الوصلة اللاسلكية بين كل عقدتين، كما يشير الشكل (8) إلى مخطط نشر عقد التحسس اللاسلكية مع الإشارة إلى العقدة ذات المعرف (ID==0) كقائد للعنقود ومصب للمعطيات(Sink).



الشكل(8): مخطط توضع عقد التحسس اللاسلكية

كما يوضح الشكل(9-a-b) مرحلة تهيئة قناة الاتصال وإعلان تهيئة النظام البرمجي عند تشغيل العقدة .

```
numnodes  11
Creating noise model for  0
Creating noise model for  1
Creating noise model for  2
Creating noise model for  3
Creating noise model for  4
Creating noise model for  5
Creating noise model for  6
Creating noise model for  7
Creating noise model for  8
Creating noise model for  9
Creating noise model for  10
Creating noise model for  11
```

```
Boot Time for Node  0 :  23542399
Boot Time for Node  1 :  25893616
Boot Time for Node  2 :  28244833
Boot Time for Node  3 :  30596050
Boot Time for Node  4 :  32947267
Boot Time for Node  5 :  35298484
Boot Time for Node  6 :  37649701
Boot Time for Node  7 :  40000918
Boot Time for Node  8 :  42352135
Boot Time for Node  9 :  44703352
Boot Time for Node  10 :  47054569
Boot Time for Node  11 :  49405786
```

الشكل(a–9): تهيئة النظام البرمجي للعقدة **الشكل(b–9): تهيئة نماذج الاتصال فيما بين العقد**

كما يوضح الشكل(10) مرحلة انقضاء زمن المؤقت وإضاءة ليد الاختبار

```
DEBUG (1): 0:0:1.002589371        APPS: MilliTimer.fired()
DEBUG (1): 0:0:1.002589371        APPS: LED0_ON
DEBUG (2): 0:0:1.002824493        APPS: MilliTimer.fired()
DEBUG (2): 0:0:1.002824493        APPS: LED0_ON
DEBUG (3): 0:0:1.003059615        APPS: MilliTimer.fired()
DEBUG (3): 0:0:1.003059615        APPS: LED0_ON
DEBUG (4): 0:0:1.003294736        APPS: MilliTimer.fired()
DEBUG (4): 0:0:1.003294736        APPS: LED0_ON
DEBUG (5): 0:0:1.003529858        APPS: MilliTimer.fired()
DEBUG (5): 0:0:1.003529858        APPS: LED0_ON
DEBUG (6): 0:0:1.003764980        APPS: MilliTimer.fired()
DEBUG (6): 0:0:1.003764980        APPS: LED0_ON
DEBUG (7): 0:0:1.004000101        APPS: MilliTimer.fired()
DEBUG (7): 0:0:1.004000101        APPS: LED0_ON
DEBUG (8): 0:0:1.004235223        APPS: MilliTimer.fired()
DEBUG (8): 0:0:1.004235223        APPS: LED0_ON
DEBUG (9): 0:0:1.004470345        APPS: MilliTimer.fired()
DEBUG (9): 0:0:1.004470345        APPS: LED0_ON
DEBUG (10): 0:0:1.004705466       APPS: MilliTimer.fired()
DEBUG (10): 0:0:1.004705466       APPS: LED0_ON
DEBUG (11): 0:0:1.004940588       APPS: MilliTimer.fired()
DEBUG (11): 0:0:1.004940588       APPS: LED0_ON
```

الشكل(10): الخرج النصي لمرحلة انقضاء المؤقت الزمني وإضاءة ليد الاختبار

يوضح الشكل(11): عملية الإرسال الدوري على مستوى جميع العقد بعد إقلاع النظام البرمجي في حين أن الاستقبال يحدث على مستوى قائد العنقود.

```
DEBUG (10): 0:0:1.006658580       APPS: sendDone!!
DEBUG (8): 0:0:1.007805759        APPS: sendDone!!
DEBUG (6): 0:0:1.008785092        APPS: sendDone!!
DEBUG (0): 0:0:1.009895512        APPS: receive!!
DEBUG (3): 0:0:1.010063358        APPS: sendDone!!
DEBUG (7): 0:0:1.010790223        APPS: sendDone!!
DEBUG (0): 0:0:1.011073208        APPS: receive!!
DEBUG (1): 0:0:1.011241054        APPS: sendDone!!
DEBUG (11): 0:0:1.011593381       APPS: sendDone!!
DEBUG (4): 0:0:1.012404180        APPS: sendDone!!
DEBUG (9): 0:0:1.012679525        APPS: sendDone!!
DEBUG (5): 0:0:1.012715595        APPS: sendDone!!
DEBUG (2): 0:0:1.012925752        APPS: sendDone!!
```

الشكل(11): عمليات الإرسال والاستقبال المنجزة عبر طبقة الشبكة

2.4.4 السيناريو2 مع تقييم النتائج: يعد تتبع وقياس درجة الحرارة من المراحل الأساسية من مهمة الفحص السريري للمريض، فالحرارة الطبيعية للجسم تتراوح ما بين 36.6 درجة مئوية و37.2 وفي الطقس الحار جداً قد ترتفع الحرارة إلى 0.5 درجة مئوية أعلى من الحد الطبيعي. يساهم مؤشر السخونة (ارتفاع الحرارة) على تسهيل التشخيص والاستعلام عن الحالة، تعد الحرارة العالية وشديدة (فرط الحرارة) عندما تكون أعلى من 41 درجة مئوية مما قد يؤدي إلى

الموت. بمقابل ذلك تعد الحرارة المنخفضة عندما تكون 35 مئوية أو أقل، إن ميزان الحرارة العادي لا يسجل أقل من 35 درجة مئوية ولهذا السبب هنالك موازين حرارة خاصة يجب أن تستعمل إن كان هناك اشتباه بحدوث انخفاض الحرارة، ويمكن أن يحدث ذلك نتيجة التعرض لفترة طويلة للبرد وأمراض قصور الدرق.

تم في هذه المرحلة محاكاة شبكة WSN على هيئة عنقود حسابي مهمته تتبع درجة حرارة المرضى والإبلاغ عن حالات تجاوز حدود الخطر بهدف ضمان سلامتهم واتخاذ الإجراءات الكفيلة بتحقيق ذلك.

تتطابق مراحل تنفيذ هذا التطبيق مع السيناريو السابق من ناحية مواضع عقد الشبكة المدرجة ونماذج الاتصال بين عقد التحسس اللاسلكية، إلا أنه ما يختلف في هذه المرحلة أن الإرسال مرتبط بتجاوز قيمة العتبة المحددة وعند حدوث ذلك يتم إرسال رسالة يشتمل محتواها على معرف العقدة، كما يكون من المتاح تقدير المسافة بين قائد العنقود والعقدة التابع باستخدام معامل شدة الإشارة RSSI الذي تستخدمه الوحدة الراديوية CC1512 المدمجة مع الشريحة MICAZ من أجل تقدير المسافة النسبية التي تفصل بين العقد الأخرى.

بعد الانتهاء من تهيئة التطبيق وفي حال حدوث تجاوز لقيمة العتبة لدى العقدة (1) والعقدة (4) بناء على درجات الحرارة التي تم قياسها لدى العقد سيتوجب إرسال رزم إعلام على هيئة إنذار إلى قائد العنقود كما يوضح الشكل (12) استقبال القائد (العقدة ذات المعرف 0) لكل من العقدتين.

```
DEBUG (1): RadioAlarm timer: timer fired, output data of sensing unit is 35.00
DEBUG (1): timer fired, output of sensing unit is abnormal!! 35.00.
DEBUG (1):  packet to send.
DEBUG (2): RadioAlarm timer: timer fired, output data of sensing unit is 36.60
DEBUG (3): RadioAlarm timer: timer fired, output data of sensing unit is 36.80
DEBUG (4): RadioAlarm timer: timer fired, output data of sensing unit is 41.00
DEBUG (4): timer fired, output of sensing unit is abnormal!! 41.00.
DEBUG (4):  packet to send.
DEBUG (5): RadioAlarm timer: timer fired, output data of sensing unit is 37.80
DEBUG (6): RadioAlarm timer: timer fired, output data of sensing unit is 38.00
DEBUG (7): RadioAlarm timer: timer fired, output data of sensing unit is 37.10
DEBUG (8): RadioAlarm timer: timer fired, output data of sensing unit is 37.20
DEBUG (9): RadioAlarm timer: timer fired, output data of sensing unit is 37.50
DEBUG (10): RadioAlarm timer: timer fired, output data of sensing unit is 37.90
DEBUG (11): RadioAlarm timer: timer fired, output data of sensing unit is 38.10
DEBUG (0): Received packet of length 3.
DEBUG (0): node id is  1.
DEBUG (1): send done packet of length.
DEBUG (0): Received packet of length 3.
DEBUG (0): node id is  4.
DEBUG (4): send done packet of length.
```

الشكل(12): نتائج محاكاة WSN من خرج وحدات التحسس إضافة إلى رزم الإنذار المرسلة

5.الاستنتاجات والتوصيات:

تم في هذا البحث دراسة ومحاكاة عمليات الإرسال والاستقبال وتحسس المعطيات ضمن إطار تطبيقات التتبع والإنذار عن المخاطر في شبكات الحساسات اللاسلكية. انجزت برمجة هذه العمليات التي تؤديها عقدة الحساس باستخدام الروابط الوظيفية المزودة في مكتبة نظام تشغيل الحساسات (TinyOS) من أجل محاكاة سلوك العقدة عقب عملية التهيئة لنظامها البرمجي، أما تطبيق المحاكاة وتقييم أداء الشبكة فكان من خلال المحاكي TOSSIM.

من الممكن تطوير التطبيق المقترح ليشمل عملية التوجيه متعدد القفزات باتجاه قائد العنقود، أيضاً يمكن تعميم التطبيق البرمجي على أنظمة الإنذار عن حدوث حرائق أو أي حادث يخل بقيمة أحد بارامترات الوسط المحيط المقاسة، مع الأخذ بالحسبان المجال الواقعي للبارامتر المقاس ضمن بيئة النشر.

6.المراجع العلمية:

1] AKYILDIZ, I. F.; SU, W.; SANKARASUBRAMANIAM, Y. ; CAYIRCI, E. 2002, *Wireless sensor networks: a survey*. *Computer networks*, *38*(4), 393-422.

2] LEVIS, P.; MADDEN, S.; POLASTRE, J.; SZEWCZYK, R.; WHITEHOUSE, K;, WOO, A.; CULLER, D. 2005, *TinyOS: An operating system for sensor networks.* In *Ambient intelligence*. Springer, Berlin, Heidelberg. pp. 115-148.

3] DUNKELS, A.; GRONVALL, B.; VOIGT, T. 2004, November, *Contiki-a lightweight and flexible operating system for tiny networked sensors*. In *29th annual IEEE international conference on local computer networks* (pp. 455-462). IEEE.

4] AWANG, A.; S UHAIMI,M. H. (2007, November), *RIMBAMON©: A forest monitoring system using wireless sensor networks*. In *2007 International Conference on Intelligent and Advanced Systems* (pp. 1101-1106). IEEE.

5] PIERA, P. J. Y.; S ALVA, J. K. G. (2019, July), *A wireless sensor network for fire detection and alarm system*. In *2019 7th International Conference on Information and Communication Technology (ICoICT)* (pp. 1-5). IEEE.

6] SILVANO, B.; OSCAR, R.; CLAUDIO, L.; MARCO, A. (2012), *A Wireless Sensor Network ad-hoc designed as anti-theft alarm system for photovoltaic panels*. *Wireless Sensor Network, 2012*.

7] YAWUT, C.; KiILASO, S. (2011, May), *A wireless sensor network for weather and disaster alarm systems. In International conference on information and electronics engineering, IPCSIT* (Vol. 6, pp. 155-159).

8] HAQUE, S. A.; RAHMAN, M.; AZIZ, S. M. (2015), *Sensor anomaly detection in wireless sensor networks for healthcare*. *Sensors*, *15*(4), 8764-8786.

9] SALMAN, A. D.; KHALAF, O. I.; A BDULSAHIB, G. M. (2019), *An adaptive intelligent alarm system for wireless sensor network. Indonesian Journal of Electrical Engineering and Computer Science*, 15(1), 142-147.

10] GAY, D.; LEVIS, P.; VON BEHREN,R.; WELSH , M.; BREWER, E.; CULLER, D. (2003), *The nesC language: A holistic approach to networked embedded systems. Acm Sigplan Notices*, *38*(5), 1-11.

11] TOSSIM - TinyOS Documentation Wiki. [online], [cit. 2010-11-19]. Available from: http://docs.tinyos.net/index.php/TOSSIM.

12] FAROOQ, M.O.; KUNZ,T. (2011), *Operating systems for wireless sensor networks: A survey*. *Sensors*, *11*(6), 5900-5930.

13] AKYILDIZ, I. F.; MELODIA, T.; CHOWDHURY, K.R.2007, *A Survey on Wireless Multimedia Sensor Networks*. *Comput. Networks. 51*, 921-960.

14] KiIM, H.; CHA, H.2007, *Multithreading Optimization Techniques for Sensor Network Operating Systems*. In *Proceedings of the 4th European Conference on Wireless Sensor Networks*, Delft, The Netherlands, January, pp. 293-308.

15] LUMIR, H.2009, *Design, implementation and simulation of intrusion detection system for wireless sensor networks.* Brno, spring ,PP.9-13.

16] VLADO, H.; JOSEPH, P;, JAN-HINRICH, H.; CORY, S.; ADAM, W.; DAVID, C.2005, *Flexible hardware abstraction for wireless sensor networks*. In Proceedings of Second European Workshop on Wireless Sensor Networks

(EWSN 2005), Istanbul, Turkey.

Moteiv. Tmote sky datasheet, 2006. http://www.cs.uvm.edu/~crobinso/mote/
17] tmote-sky-datasheet-102.pdf.

HYUNGJUNE, L.; PHILIP, L.2007, *Improving Wireless Simulation*
18] *Through Noise Modeling. In Information Processing in Sensor Networks.* IPSN 2007. 6th International Symposium on, pages 21 –30.

KLUES, K.; LIANG, C.J.M.; PAEK, J.; MUSALOIU, R.; LEVIS, P.;
19] TERZIS, A.; GOVINDAN, R.2009, *TOSThread: Thread-Safe and Non-Invasive Preemption in TinyOS.* In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, Berkeley, CA, USA, 4–6, pp. 127-140.

MUHAMMAD, O. F. ;THOMAS, K.2011, *Operating Systems for Wireless*
20] *Sensor Networks: A Survey Sensors*, 11, 5900-5930.

COOPRIDER, N.; ARCHER, W.; EIDE, E.; GAY, D.; REGEHR, J.2007,
21] *Efficient Memory Safety for Tinyos*. *In Proceedings of the 5th International Conference on Embedded Networked Sensor Systems (SenSys,07)*, New York, NY, USA, pp. 205-218.

TinyOS Network Working Group; Available online:
22] http://docs.tinyos.net/index.php/TinyOS_ Tutorials#Network_Protocols (accessed on 17 April 2011).

Network Protocols TinyOS documentation Wiki; Available online:
23] http://docs.tinyos.net/ index.php/Network_Protocols (accessed on 17 April 2011).

LIN, K.; LEVIS, P.2008, *Data Discovery and Dissemination with DIP*. *In*
24] *Proceedings of the 7th International Conference on Information Processing in Sensor Networks, S*t. Louis, MO, USA, pp. 433-444.

MADDEN, S.R.; F RANKLIN, M.J.; HELLERSTEIN, J.M.; HONG,
25] W.2005, *TinyDB: An Aquisitional Query Processing System for Sensor Networks*. ACM Tans. Database Syst.

KARLOF, C.; SASTRY, N.; WAGNER, D.2004, *TinySec: A Link Layer*
26] *Security for Wireless Sensor Networks*. In *Proceedings of the 2th ACM SenSys*, Baltimore, MD, USA, 3–5.

LEVIS, P.; LEE, N.; WELSH, M.; CULLER, D.2003, ToSSIM: Accurate
27] and Scale able Simulation of Entire TinyOS Applications. In Proceedings of the 1st ACM SenSys, Los Angeles, CA, USA, 5–7.

Ptolemy Project; Available online: http://ptolemy.eecs.berkeley.edu/
28] (accessed on 17 April 2011).

PoteIV Cooperation; Available online: http://www. moteiv.com (accessed
29] on 17 April 2011).

MoteIV Cooperation; Available online: http://www. moteiv.com (accessed
30] on 17 April 2011).

MONTENEGRO, G.; KUSHALNAGAR, N.; HUI, J.; CULLER, D.2011,
31] *Transmission of Ipv6 Packets over IEEE 802.15.4 Networks, RFC 4944; Available online:* http://tools.ietf.org/html/rfc4944 (accessed on 17 April 2011).