المملكة العربية السعودية

وزارة التعليم
Ministry of Education

قررت وزارة التعليم تدريس
هذا الكتاب وطبعه على نفقتها

إنترنت الأشياء

التعليم الثانوي - نظام المسارات

السنة الثانية

طبعة 1446-2024

يوزع مجاناً ولا يباع

الجزء الثاني

الوحدة الخامسة
تطبيقات إنترنت الأشياء المتقدمة

الوحدة السادسة
برمجة إنترنت الأشياء باستخدام C++

الوحدة السابعة
الرسائل في إنترنت الأشياء

الوحدة الثامنة
محاكاة شبكة مُستشعرات إنترنت الأشياء اللاسلكية

وزارة التعليم
Ministry of Education
2024 - 1446

وزارة التعليم
Ministry of Education
2024 - 1446

5. تطبيقات إنترنت الأشياء المتقدمة

سيتعرف الطالب في هذه الوحدة على التطبيقات المستخدمة لحلول إنترنت الأشياء في مجال الزراعة ومجال الرعاية الصحية. وسيتعرف أيضًا على هيكليات إنترنت الأشياء، ويكتشف بروتوكولات الشبكات المختلفة. وفي الختام سيتعرف على مفاهيم الأمان والخصوصية في أنظمة إنترنت الأشياء.

أهداف التعلُّم

بنهاية هذه الوحدة سيكون الطالب قادرًا على أن:

- يصف كيفية استخدام تقنيات إنترنت الأشياء في مجال الرعاية الصحية (IoHT).
- يُحدِّد تطبيقات الرعاية الصحية الذكية المختلفة.
- يصف مساهمة تقنيات إنترنت الأشياء في تحسين قطاع الزراعة.
- يُصنِّف طبقات إنترنت الأشياء الأحادية من آلة إلى آلة (M2M).
- يشرح وظائف طبقات الهيكلية العالمية لإنترنت الأشياء.
- يُحدِّد الخصائص الرئيسة لتقنية تحديد الترددات الراديوية (RFID) وتقنية الاتصال قريب المدى (NFC).
- يُحدِّد التقنيات والبروتوكولات المستخدمة في شبكات المنطقة الشخصية اللاسلكية (WPANS).
- يُميِّز التحديات الأمنية في شبكة الجيل الخامس من أنظمة إنترنت الأشياء.
- يُعرِّف المخاوف المتعلقة بالخصوصية الكامنة في إنترنت الأشياء وحلولها المُمكنة.

رابط الدرس الرقمي

www.ien.edu.sa

الدرس الأول

مجالات تطبيق إنترنت الأشياء

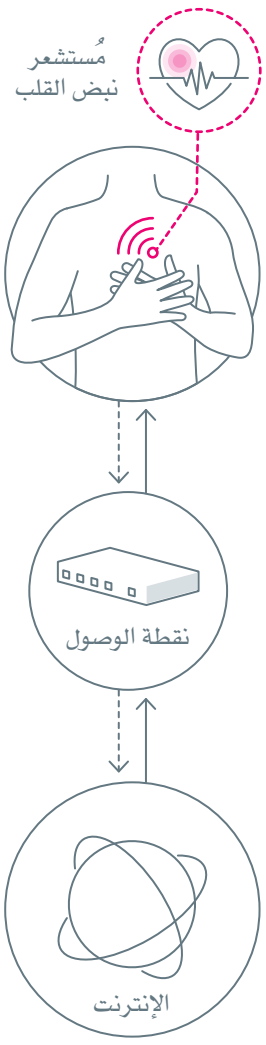
لقد سبق لك وأن تعرفت في الجزء الأول من كتاب إنترنت الأشياء على المفاهيم الأساسية **لإنترنت الأشياء** (IoT) بصفتها إحدى أهم التقنيات الناشئة، وتعرفت أيضًا على بعض التطبيقات الأساسية التي تُستخدم بها تقنية إنترنت الأشياء، وستوسع الآن معرفتك السابقة بتعلُّم تطبيقات جديدة لإنترنت الأشياء.

الرعاية الصحية الذكية Smart Healthcare

إن تطبيق إنترنت الأشياء في مجال الرعاية الصحية له أثرٌ كبيرٌ في المجتمع، فأجهزة إنترنت الأشياء المختلفة القابلة للارتداء مثل: المُستشعرات، وأجهزة مراقبة الصحة عن بُعد، وتنبيهات الطوارئ، بالإضافة إلى أجهزة مراقبة المؤشرات الصحية، جميعها تشتمل على أدوات لمراقبة صحة الأشخاص وتتبعها من خلال تقنيات مبتكرة تهدف لتعزيز جودة حياة الإنسان ورفاهيته، وذلك بدءًا من مراقبة الأطفال المرضى إلى تشخيص الأمراض المزمنة ومراقبتها لدى كبار السن، مما يتيح توفير خدمات رعاية صحية فعّالة لجميع الأعمار.

تطور الرعاية الصحية The Evolution of Healthcare

أدَّت الزيادة المُطردة في عدد السكان إلى ظهور تحديات جديدة يمكن التغلب عليها من خلال الرعاية الصحية الذكية. يشير مصطلح الرعاية الصحية الذكية إلى تطبيق التقنية لتحسين نوعية الحياة. ورغمَ أنّ القصور العام في المعرفة الرقمية لدى بعض العاملين في مجال الرعاية الصحية أدّى إلى إبطاء عملية الانتقال إلى الرعاية الصحية الذكية، إلا أنّ الحكومات والمؤسسات الخاصة تستثمر بشكلٍ واسعٍ في مشاريع دمج التقنيات لتحسين نظام الرعاية الصحية، ويعتمد النظام التقليدي للخدمات الطبية والصحية على مبدأ زيارة المريض لعيادة الطبيب أو المركز الطبي المحلي أو المستشفى عند الحاجة، أما النظام القائم على الرعاية الصحية الذكية فيتيح للمرضى التعامل مع الظروف الطارئة باستقلالية، وبذلك يتحول التركيز على الرعاية الصحية الفردية من العلاج التقليدي في المستشفى إلى الرعاية المنزلية الذكية. توفر الرعاية الصحية الذكية باستخدام أجهزة إنترنت الأشياء إمكانية مراقبة الصحة عن بُعد، وعرض تنبيهات الطوارئ، وتوفير خدمات علاجية بتكلفة معقولة، وكذلك ضمان توافر الخدمات الطبية للجميع بغض النظر عن الموقع أو البُعد والقرب من المراكز الطبية والمستشفيات. تتنوع أجهزة مراقبة الصحة هذه، بدءًا من أجهزة مراقبة اللياقة البدنية وأجهزة التتبع التي تقيس المؤشرات الصحية، مرورًا بالتقنيات المتطورة القابلة للارتداء التي تجمع العديد من المؤشرات الحيوية معًا.



شكل 5.1: مراقبة المؤشرات الصحية

إنترنت أشياء الرعاية الصحية Internet of Healthcare Things

إنترنت أشياء الرعاية الصحية (IoHT) هو أحد حلول إنترنت الأشياء التي تستخدِم تلك التقنية لربط الأشخاص بخدمات الرعاية الصحية المختلفة، ويُمكن للأطباء الأخصائيين من خلال استخدام هذه التقنية القيام بمراجعة التقارير والسجلات الطبية للمرضى عن بُعد، وتقديم التشخيص والتوصيات دون الوجود الفعلي في نفس الموقع مع المريض. يتكون إنترنت أشياء الرعاية الصحية من شبكة متصلة من التقنيات الطبية تشمل: التصوير الطبي، وتقارير المختبرات الطبية، وأجهزة مراقبة الرعاية الصحية عن بُعد، ويشمل التصوير الطبي التصوير بالأشعة السينية، **والتصوير بالرنين المغناطيسي** (Magnetic Resonance Imaging – MRI)، **والتصوير المقطعي المحوسب** (Computerized Tomography – CT)، والأنواع الأخرى من التصوير، ويشمل إنترنت أشياء الرعاية الصحية أيضًا خدمات الطوارئ كسيارات الإسعاف الذكية والعيادات الذكية.

الأجهزة القابلة للارتداء Wearables

الأجهزة القابلة للارتداء هي أشياء ذكية توضع على جسم الإنسان، ويُمكنها جمع البيانات المتعلقة بصحة الشخص وتخزينها ومعالجتها وتحليلها؛ لتوفير المعلومات المطلوبة وإرسال التنبيهات في سيناريوهات الطوارئ، ويُعدّ المرضى الذين يعانون من إعاقات مؤقتة أو دائمة، وكذلك كبار السن والأطفال من المستخدِمين الأساسيين لهذه الأجهزة، حيث تقوم المُستشعرات الحيوية المُدمجة في ملابس المريض بالتقاط البيانات وإنتاج مُخرجات كهربائية رقمية يمكن استخدامها لمراقبة المؤشرات الصحية الخاصة بذلك المريض، ويُعدّ المُستشعر البيولوجي أداة تحليلية مُصغرة مُدمجة مع مُكون حيوي يتعرف على إشارات معينة، وتختلف المُستشعرات والمُشغِّلات حسب طبيعة أنظمة المراقبة المنوطة بها، ويمكنها جمع البيانات ونقلها كالإشارات الحيوية، ودرجة حرارة الجسم، ومستوى تشبع الأكسجين في الدم (قياس التأكسج النبضي)، وحركة الإنسان، والموقع الجغرافي للشخص. توجد العديد من الإشارات الحيوية المتولدة من الجسم مثل: **مخطط كهربية القلب** (Electrocardiogram – ECG)، و**مخطط كهربية الدماغ** (Electroencephalogram – EEG)، و**مخطط كهربية العضل** (Electromyography – EMG). يمكن للمُستشعرات مراقبة المؤشرات الفسيولوجية أو الميكانيكية الحيوية للإنسان مثل: مُعدَّل ضربات القلب، ونشاط العضلات، ومُعدَّل التنفس، ودرجة حرارة الجسم، وضغط الدم، ووضع الجسم، والحركة، والتسارع، وعادةً ما تكون مُخرجات المُستشعرات الذكية وأجهزة إنترنت الأشياء عالية التعقيد، مما يستلزم تطبيق الذكاء الاصطناعي وتحليلات البيانات وتقنيات أخرى مثل الحوسبة السحابية.

مخطط كهربية الدماغ (Electroencephalogram – EEG):
جهاز تخطيط كهربيّة الدماغ هو أداة لتشخيص النشاطات غير الطبيعية في الإشارات الكهربية في الدماغ.

شكل 5.2: شبكات مُستشعرات الجسم المتصلة بشبكات إنترنت أشياء الرعاية الصحية

شبكة مُستشعرات الجسم Body Sensor Network

شبكة مُستشعرات الجسم (BSN) هي **شبكة مُستشعرات لاسلكية** (Wireless Sensor Network – WSN) تُستخدم لمراقبة جسم الإنسان، فهي عبارة عن شبكة عقدية حساسة يمكن ارتداؤها، ويمكنها الاتصال بالعُقد والكائنات الذكية الأخرى. تحتوي عُقد الاستشعار على قدرات الحوسبة والتخزين والإرسال اللاسلكي بالإضافة إلى الاستشعار. وكما يظهر في الشكل 5.2، يُرسل مُستشعر تدفق الدم بيانات المريض إلى جهاز ذكي يتصل بالإنترنت، ويُرسل هذه البيانات إلى المستشفى الذكي. وعلى الرغم من أن الأنظمة القائمة على شبكة مُستشعرات الجسم تضم مجموعة متنوعة من التطبيقات، إلا أنه يمكن استخدامها للمراقبة المستمرة وغير الجراحية للمؤشرات الحيوية، حيث تُوضع مُستشعرات لاسلكية صغيرة على الجلد، وقد تدمج في بعض الحالات بالملابس، وهذا من شأنه أن يُسهّل التعرف المبكر على المرض وتشخيصه. عادةً ما تكتشف هذه المُستشعرات بيانات عن حركة جسم الإنسان، ودرجة حرارة الجسم، ومعدل ضربات القلب، ومعدل مُوصلية الجلد، ووظائف العضلات.

تطبيقات الرعاية الصحية الذكية Smart Healthcare Applications

مراقبة ضغط الدم Blood pressure monitoring

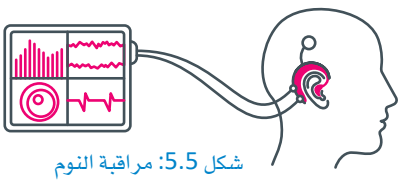
يرتبط الاختلاف في المعدل النموذجي لضخ القلب للدم بارتفاع ضغط الدم لدى البشر، ويُعدّ ارتفاع ضغط الدم مشكلة صحية عالمية ناجمة عن ارتفاع ضغط الدم في الشرايين، ويتسبب ارتفاع ضغط الدم المُزمن في العديد من المشاكل الصحية بما فيها قصور عضلة القلب، وأمراض الكلى المزمنة، وتلف العصب البصري وبالتالي فقدان البصر. تُستخدم الساعات الذكية بصفتها أجهزة إنترنت أشياء قابلة للارتداء في تتبع بيانات لياقة المستخدِم وتسجيلها وقياس معدل ضربات القلب وكذلك في مراقبة مؤشراتٍ حيوية أخرى كضغط الدم، حيث تقوم بإرسال تلك البيانات ومعالجتها، وقد أصبحت أنظمة إنترنت أشياء الرعاية الصحية القائمة على الحوسبة السحابية شائعة على نطاقٍ واسع، مما يسمح للمرضى بمراقبة ضغط الدم والسيطرة عليه باستخدام أجهزة إنترنت الأشياء.

شكل 5.3: مراقبة ضغط الدم

مراقبة الألم Pain monitoring

يُعدُّ التعرف على المشاعر والأنواع المختلفة من الألم لدى البشر أمرًا ضروريًا لأجل تقديم رعاية صحية جيدة للمرضى. يكتسب التعرف على المشاعر والألم أهمية خاصة عند التعامل مع صغار السن والمسنين وأولئك المصابين بالأمراض العقلية أو الذين يفتقدون القدرة على التعبير بشكلٍ لفظي أو وصف مشاعرهم وآلامهم بشكل واضح. تُعدُّ تعابير الوجه مؤشرًا سلوكيًا للألم نظرًا لأن الشعور بالألم يولِّد تغيرات في تعابير الوجه، فيمكن استخدامها كأسلوب تلقائي لتشخيص انزعاج الإنسان. توفر القدرة على قياس تعابير الوجه بديلًا عن الأساليب القياسية لقياس المشاعر والألم، ويمكن استخدامها مع من لا يستطيعون التعبير كمرضى العناية المركزة والرُضَّع، وفي الواقع يلاحظ الكثير من الآباء تعبيرات أوجه أطفالهم لأنها تنقل معلومات حول صحتهم، ويُعدُّ تطوير نظام آلي للتعرف على الألم باستخدام مُدخلات فسيولوجية من مُستشعرات إنترنت الأشياء وتحليل البيانات مهمًا في تقييم أنواع مختلفة من المشاعر والألم.

شكل 5.4: مراقبة الألم



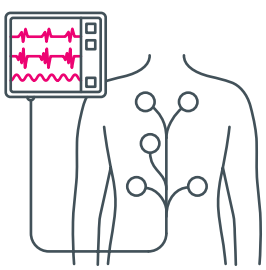
شكل 5.5: مراقبة النوم

مراقبة النوم Sleep monitoring

النوم هو حالة طبيعية ودورية من الراحة النفسية والجسدية، وقد يعاني العديد من الأفراد من اضطرابات النوم، والتي تشمل الأرق وتوقف التنفس وانقطاع النفس الانسدادي أثناء النوم، ويُعدّ **انقطاع النفس الانسدادي النومي** (Obstructive Sleep Apnea – OSA) مرضًا تنفسيًا قاتلًا أثناء النوم يُؤثر سلبًا على حياة الشخص على الصعيد النفسي والجسدي. تتوفر أنظمة عديدة للكشف عن هذا المرض، مثل أجهزة **مخطط كهربية الدماغ** (EEG) في الأذن القابل للارتداء والمتصل بشبكة إنترنت الأشياء في الغرفة. تُوفِّر هذه الأنظمة طريقة مستمرة وغير مزعجة لمراقبة النوم على مدار الساعة طوال أيام الأسبوع وذلك لتقييم جودة النوم. تُستخدم البيانات المُجمَّعة للتنبؤ بمراحل النوم باستخدام خوارزميات الذكاء الاصطناعي.

مراقبة مُخطط كهربية القلب Electrocardiogram monitoring

تلتقط المُستشعرات التي توضع على الجلد الإشارات الكهربائية الناتجة عن ضربات القلب، وتوضع الأقطاب الكهربائية الخاصة بجهاز تخطيط كهربية القلب على مواضع محددة من صدر المريض في العيادات والمستشفيات، حيث لا يمكن للمرضى استخدام مثل تلك الأجهزة في المنزل، ولذلك فقد تم تطوير العديد من الأشياء الذكية التي تُستخدم لفحوصات تخطيط القلب عن بُعد، بحيث يُمكن للأطباء معاينة بيانات المرضى من خلال هذه الأجهزة القابلة للارتداء. تحتوي بعض هذه الأشياء الذكية على تطبيقات للتنبيه والتحذير في حالات النوبات القلبية وكذلك لتقديم توصيات لصحة القلب للأشخاص المعنيين.



شكل 5.6: مراقبة مخطط كهربية القلب

مخطط كهربية القلب (Electrocardiogram - ECG):

مخطط كهربية القلب هو اختبار يقيس النشاط الكهربائي للقلب لتحديد ما إذا كان القلب يعمل بشكل صحيح.

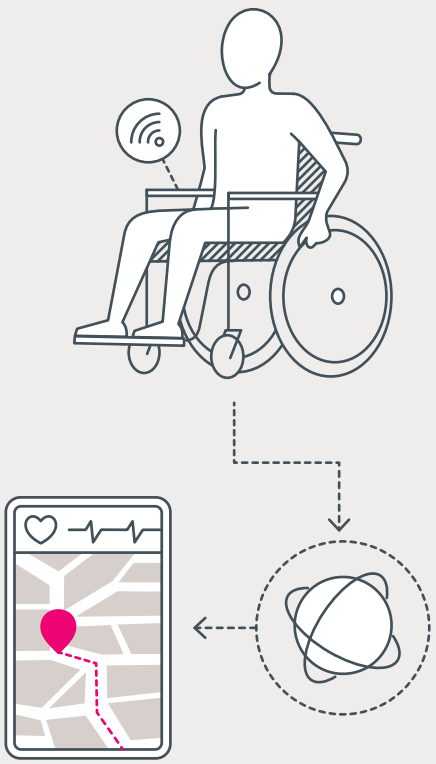
مراقبة علم الأمراض Pathology monitoring

يوصف علم الأمراض بأنه الدراسة العلمية لأصول وتأثيرات المرض والإصابة، ويقوم جهاز **مخطط كهربية الدماغ** (EEG) بتشخيص الأمراض من خلال توصيل أقراص معدنية صغيرة بأسلاك رفيعة بفروة الرأس تُرسل إشارات إلى جهاز حاسب لتخزين النتائج، ويُستخدم جهاز مخطط كهربية الدماغ على نطاق واسع لتشخيص الاضطرابات المتعلقة بالدماغ بسبب تكلفته المنخفضة وطبيعته غير الجراحية، ويُمكن لجهاز تخطيط كهربية الدماغ تشخيص بعض الاضطرابات المتعلقة بالدماغ مثل: الصرع والسكتة الدماغية. يحتاج المرضى الذين يعانون من هذه الحالات إلى عناية فورية؛ لأن أي تأخير قد يؤثر على حياتهم. وهكذا يُمكن أن يكون نظام إنترنت الأشياء الذي يراقب حالة المريض مُنقذًا للحياة في مثل هذه المواقف.

شكل 5.7: مراقبة علم الأمراض

مراقبة الأشخاص ذوي الإعاقة Disabled persons monitoring

تُعدُّ **الكراسي المتحركة الذكية** (Smart Wheelchairs – SMW) المتصلة بأنظمة إنترنت الأشياء موضوعًا بحثيًا جديدًا، ويتكون تصميم هذه الأنظمة من عنصرين: خدمة الخرائط المستخدمة للملاحة، والكرسي المتحرك للمستخدِم، وتتضمن الكراسي المتحركة الذكية تقنية **قياس المسافات ثلاثية الأبعاد** (3D LIDAR)، وذلك لرسم خرائط للمحيط الخارجي وحركتها المستقلة دون الحاجة إلى **نظام تحديد المواقع العالمي** (Global Positioning System – GPS). تستخدِم هذه التقنية كلًا من هندسة التحكم للكرسي المتحرك المزود بمحرك، ونظامًا مُدمجًا لمراقبة المرضى ذوي الحالات الحرجة، ويستخدِم النظام المُضمَّن أيضًا الخصائص الحيوية للمستخدِم لاكتشاف المواقف الخطرة المحتملة. يُصدر الكرسي المتحرك تحذيرًا عن طريق تنشيط تنبيه عند قياسه لنبضات القلب وارتفاع ضغط الدم بشكلٍ دوري.



شكل 5.8: مراقبة ذوي الإعاقة

تقنية اكتشاف الضوء والمدى (Light Detection And Ranging - LIDAR):

هي تقنية لقياس المسافات عن طريق توجيه الليزر إلى عنصر أو سطح وقياس الوقت اللازم لانعكاس الضوء إلى المُرسل.

مثال

أطلقت شركة اتصالات سعودية مشروع العيادة الافتراضية، وتُستخدم هذه العيادة من قِبل الأطباء لتشخيص مرضاهم عن بُعد، وتستخدِم هذه الخدمات أنظمة شبكات إنترنت الأشياء من خلال الأجهزة القابلة للارتداء لمساعدة الأطباء لجمع البيانات الضرورية، والتي تُرسل للمستشفيات والمراكز الطبية المحلية لمتابعة حالات المرضى.

الزراعة الذكية Smart Agriculture

لقد قُمت في الوحدة السابقة بأولى خطواتك في مجال الزراعة الذكية، وذلك من خلال إنشاء نظام لري النبات. يمكن تحسين وتطوير القطاع الزراعي وسير عمله من خلال استخدام العديد من تقنيات إنترنت الأشياء، حيث يتيح تطبيق إنترنت الأشياء في القطاع الزراعي مزايا خاصة مثل: ترشيد استخدام الموارد كالأرض والمياه والأسمدة ومبيدات الآفات؛ وكذلك تحسين الأرباح وتحقيق الاستدامة، وسلامة الغذاء وحماية البيئة، وخفض تكاليف الإنتاج.

شكل 5.9: الزراعة الذكية باستخدام الطائرات دون طيار

تطبيقات الزراعة الذكية Smart Agriculture Applications

الزراعة الدقيقة Precision farming

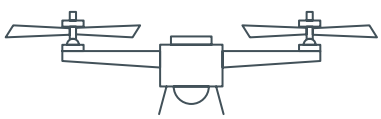
تتضمن الزراعة الدقيقة ري النباتات حسب موقعها ووفق كميّات المياه التي تحتاج إليها الأنواع المختلفة من النباتات، ويتطلب هذا النوع من الزراعة جمع وتحليل البيانات من خلال العديد من المُستشعرات مثل موقع النبات والرطوبة ودرجة حرارة الأرض، والتي يمكن الحصول عليها من خلال المسح والمراقبة الجوية، وقد اكتسبت الطائرات التي يُتحكم فيها عن بُعد، والتي تُعرَف غالبًا باسم **المركبات الجوية دون طيار** (Unmanned Aerial Vehicles – UAVs) أو **الطائرات المسيرة** (Drones)، شيوعًا كبيرًا في عمليات المراقبة الجوية. فعلى مدار السنوات الماضية، استُخدمت الطائرات دون طيار على نطاق واسع لمراقبة الحقول والمزروعات، ولتقديم حلول زراعية دقيقة وفعّالة. لقد أضحى من الممكن من خلال استخدام الاستشعار عن بُعد متابعة مجموعة متنوعة من المقاييس المتعلقة بالمحاصيل والغطاء النباتي، وذلك بالاستعانة بصور ذات أطوال موجية متفاوتة حلّت بديلًا عن صور الأقمار الصناعية التي كان يُعتمد عليها في الماضي، وقد أثبتت أنظمة الطائرات دون طيار فعاليتها في العديد من تطبيقات الزراعة الدقيقة، بما فيها رش مبيدات الآفات، والتعرف على نقص المياه، وتحديد أمراض النباتات. وهكذا أصبح بالإمكان اتخاذ العديد من القرارات بناءً على البيانات المُلتقطة من الطائرات دون طيار لتقدير تكاليف معالجة المشاكل المحددة وتحسين جودة الإنتاج.

يتمثل دور الطائرات دون طيار في التقاط البيانات بتفاصيل مكانية دقيقة، حيث تُستخدم العديد من المُستشعرات اعتمادًا على المؤشرات الزراعية التي يجب مراقبتها، ويجب أن تفي مُستشعرات الطائرات دون طيار بثلاثة متطلبات أساسية: استهلاك مُنخفض للطاقة، وخفة الوزن، وصِغَر الحجم. تعمل هذه التقنيات على إنشاء خرائط بيئية تصور طبيعة التربة، مما يسمح بتخطيط أنظمة ري أكثر كفاءة لجميع المحاصيل، وتُستخدم تقنيات **نظام تحديد المواقع العالمي** (GPS) على نطاق واسع للمساعدة في تحديد المواقع والإسناد الجغرافي للأشياء المُلتقطة بواسطة الاستشعار عن بُعد، ونظرًا لأن معلومات الاستشعار عن بُعد تُعدُّ مصدرًا رئيسًا للبيانات البيئية؛ فإنّه يتم في العادة استيرادها إلى **أنظمة المعلومات الجغرافية** (Geographic Information Systems – GISs) ودمجها مع مجموعات البيانات الأخرى.

مركبة جوية دون طيار

(Unmanned Aerial Vehicle - UAV):

يتم تسيير "المركبات الجوية دون طيار" دون طيارين أو طاقم أو ركاب.



شكل 5.10: المتطلبات الأساسية للمركبات الجوية دون طيار

جدول 5.1: أنواع المُستشعرات الهامة المُستخدمة في المركبات الجوية دون طيار (UAVs)

نوع المُستشعر	الوصف
 مُستشعرات الإضاءة المرئية	تلتقط الصور في ظروف مختلفة، بما في ذلك الطقس المُشمس والغائم، وتعتمد جودة الصور على ظروف الإضاءة.
 مُستشعرات الأشعة تحت الحمراء الحرارية	تقيس مُستشعرات الأشعة تحت الحمراء الحرارية درجات حرارة السطح. فباستخدام مُستشعرات الأشعة تحت الحمراء وعدسة بصرية، تجمع الكاميرات الحرارية طاقة الأشعة تحت الحمراء. تركز كاميرات التصوير الحراري على الإشعاع وتكتشفه عند نفس الأطوال الموجية، ثم تحوله إلى صور ذات تدرجات رمادية تمثل الحرارة، ويمكن لأجهزة استشعار التصوير الحراري المتعددة إنشاء صورٍ ملونة أيضًا.
 مُستشعرات التصوير متعددة الأطياف	تجمع المُستشعرات متعددة الأطياف الأطوال الموجية المرئية وكذلك الأطوال الموجية التي تقع خارج الطيف المرئي، بما في ذلك **الأشعة تحت الحمراء القريبة** (Near-Infrared Radiation – NIR) و**الأشعة تحت الحمراء قصيرة الموجة** (Short-Wave Infrared Radiation – SWIR) وغيرها. تقوم الطائرات دون طيار المزودة بمُستشعرات متعددة الأطياف أو فائقة الطيف بجمع معلومات امتصاص المحاصيل للمياه، وعلى الرغم من تكلفتها العالية، إلا أنّ البيانات الطيفيّة يمكن أن تكون ذات قيمة كبيرة لتقييم العديد من الخصائص البيولوجية والفيزيائية للمحاصيل.

الري الدقيق Precision irrigation

تُعدُّ تقنية الري الدقيق تقنية زراعية تحافظ على العناصر الغذائية وتُحسِّن كمية المياه التي تتطلبها النباتات من خلال تزويد جذور النباتات بقطرات الماء ببطء تحت سطح التربة أو فوقه، كما يتم زيادة إنتاجية المحاصيل باستخدام تقنيات إنترنت الأشياء الدقيقة للري، حيث تحدِّد المُستشعرات الثابتة الخصائص الفيزيائية والكيميائية للأراضي الزراعية، بما فيها الطقس ودرجة الحرارة والرطوبة، وصحة النبات، ورطوبة وحموضة التربة، ومغذيات التربة. يتم تحليل البيانات التي تُجمع لإبلاغ المزارعين بالتعديلات التي يتعين عليهم إجراؤها، كما يساعد تحليل البيانات في تحديد العناصر الغذائية المناسبة وكمياتها، وكذلك تحديد كمية المياه اللازمة للري.

شكل 5.11: تطبيق الري الدقيق

الزراعة العمودية Vertical farming

يتم في الزراعة العمودية زراعة النباتات بنطاق رأسي وليس أفقي، مما يسمح بإنتاج المزيد من المحاصيل في مساحاتٍ صغيرة، وكذلك زراعة أنواع متعددة من المحاصيل في ذات الوقت. يمكن التعامل مع الأجهزة باستخدام تقنيات إنترنت الأشياء عن بُعد باستخدام تقنيات الاتصال مثل البلوتوث والواي فاي وتقنية الاتصال اللاسلكي (RFID)، وتهدف الزراعة العمودية إلى زراعة المحاصيل في البيئات الحضرية، ويتمتع نظام الزراعة العمودية الداخلي بمناخ مثالي بعيدًا عن القلق من المؤثرات البيئية الخارجية، وتُعدُّ تقنيات إنترنت الأشياء ضرورية في بيئة الزراعة لمراقبة صحة النبات والري، حيث تتطلب الزراعة العمودية معالجة كميات هائلة من البيانات وتحليلها للمساهمة في تطور المحاصيل بشكل فعّال، كما يمكن تحسين الإنتاجية الزراعية بالزراعة العمودية مثل أتمتة العملية برمتها من وضع البذور إلى حصاد المحصول في بيئة مغلقة.

شكل 5.12: تطبيق الزراعة العمودية

مثال

من المخطط أن تكون مدينة نيوم العملاقة في المملكة العربية السعودية مدينة عمودية تُستخدم فيها أحدث التقنيات لحل مشاكل التلوث والنقل واستدامة الغذاء. ستحتوي مدينة نيوم على مبنيين يبلغ ارتفاع كل منهما 500 متر، ويبعدان عن بعضهما مسافة 200 متر، ويمتدان بالتوازي لمسافة 170 كيلومترًا. تقع المدينة العمودية المتطورة في المنطقة الواقعة بين هذين المبنيين، وتهدف نيوم إلى إنشاء أول نظام متكامل للاكتفاء الذاتي الغذائي الصحراوي. ومع ندرة المياه المتاحة، ستكون هناك حاجة لأنظمة زراعة ذكية لإنشاء مجتمعات مُكتفية ذاتيًا. تُعزّز تقنيات الزراعة الدائرية والزراعة العمودية من خلال تقنيات إنترنت الأشياء والذكاء الاصطناعي لتحسين استخدام الموارد وتعزيز الإنتاج الزراعي. تعرف أكثر على مشروع ذا لاين بمدينة نيوم من هنا: https://www.neom.com/ar-sa/about.

تمرينات

1

حدِّد الجملة الصحيحة والجملة الخاطئة فيما يلي:	صحيحة	خاطئة
1. لا تسهم تقنيات إنترنت الأشياء في تحسين مجال الرعاية الصحية.	○	○
2. يُعدُّ إنترنت أشياء الرعاية الصحية امتدادًا لإنترنت الأشياء.	○	○
3. تتصل كافة الأجهزة الطبية القابلة للارتداء بصورة مستمرة بشبكة الإنترنت.	○	○
4. يمكن لشبكات مُستشعرات الجسم أن تكون أنظمة إنترنت أشياء مستقلة.	○	○
5. يتضمن الكرسي المتحرك الذكي نظامًا مُدمجًا يستخدِم الخصائص الحيوية لمستخدِمه لاكتشاف المواقف الخطرة المحتملة.	○	○
6. يُمكن للمركبات الجوية دون طيار إجراء نوع واحد فقط من المسح للأراضي الزراعية.	○	○
7. تكتشف مُستشعرات الأشعة تحت الحمراء الحرارية أي إشعاع حراري.	○	○
8. يُستخدم الري الدقيق لتحسين استخدام الموارد اللازمة للأنظمة الزراعية.	○	○
9. لا يحتاج نظام الري الدقيق إلى الكثير من المُستشعرات في عمله.	○	○
10. تُستخدم الزراعة العمودية لإتاحة الاستخدام الأفضل للأراضي الزراعية.	○	○

2 وضِّح المقصود بإنترنت أشياء الرعاية الصحية.

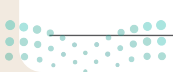
3 قارن بين أنواع البيانات التي يمكن جمعها بواسطة الأشياء الذكية القابلة للارتداء.

4 مِمَ تتكون شبكة مُستشعرات الجسم؟

5 كيف يُمكن استخدام تقنيات الذكاء الاصطناعي لحلول إنترنت الأشياء في مراقبة المشاعر والألم؟

6 صِف كيفية استخدام المركبات الجوية دون طيار في الزراعة الدقيقة لتطبيقات إنترنت الأشياء.

7 صنِّف الأنواع المختلفة لمُستشعرات المركبات الجوية دون طيار.



8 صِفْ كيفية مساهمة أنظمة إنترنت الأشياء في تطبيقات الري الدقيق.

9 ما مدى اعتماد الزراعة العمودية على حلول إنترنت الأشياء الفعّالة؟

رابط الدرس الرقمي

www.ien.edu.sa

الدرس الثاني

تقنيات شبكات إنترنت الأشياء

المقارنة بين هيكلية شبكة oneM2M وهيكلية أنظمة إنترنت الأشياء العالمي

oneM2M Architecture Versus IoT World Forum Architecture

أدى التطور السريع للاتصالات من **آلة إلى آلة** (M2M) إلى إنشاء هيكليات إنترنت أشياء مختلفة. تساعد هذه الهيكليات في تسريع اعتماد تطبيقات وأجهزة (M2M) بما فيها إنترنت الأشياء، وتُعدُّ هيكلية (oneM2M) وهيكلية أنظمة إنترنت الأشياء العالمي من هيكليات إنترنت الأشياء الأكثر شيوعًا على نطاق واسع. تصمِّم هيكلية (oneM2M) حلول إنترنت أشياء تختص بالأجهزة وتطبيقاتها فقط، بينما تراعي هيكلية أنظمة إنترنت الأشياء العالمية تقنيات أخرى مثل: تخزين البيانات ومعالجتها، والاتصال بالشبكة، والحوسبة المتطورة.

آلة إلى آلة

(Machine To Machine - M2M):

يصف مصطلح **آلة إلى آلة** (M2M) أي تقنية تُمكِّن الأجهزة المتصلة بالشبكة من تبادل البيانات وتنفيذ المهام دون أي تدخل بشري.

هيكلية oneM2M Architecture oneM2M

يُعدُّ التعامل مع مجموعة متنوعة من الأجهزة والبرامج وطرائق الوصول أحد أكبر التحديات التي تواجهها عملية تطوير هيكلية إنترنت الأشياء. تقوم هيكلية (oneM2M) من خلال إنشاء تصميم منصة أفقية بإنشاء معايير التشغيل البيني على جميع مستويات مراحل إنترنت الأشياء. بناءً على هيكلية (oneM2M)، يتم تقسيم وظائف إنترنت الأشياء إلى ثلاث طبقات: طبقة التطبيقات، وطبقة الخدمات، وطبقة الشبكة. قد تبدو هذه الهيكلية للوهلة الأولى أساسية وعامة نسبيًا؛ ولكنها رغم ذلك غنية جدًا وداعمة للتشغيل البيني عبر واجهات برمجة لتطبيقات تقنية المعلومات، وتدعم مجموعة واسعة من تقنيات إنترنت الأشياء.

الشبكات المُعرفة بالبرمجيات

(Software-Defined Networks - SDN):

الشبكة المعرفة بالبرمجيات هي إحدى هيكليات الشبكات، والتي يُتحكّم بها من خلال وحدات تحكم قائمة على البرامج أو **واجهات برمجة التطبيقات** (Application Programming Interfaces – APIs) عوضًا عن استخدام المعدات أو الأجهزة المتخصصة.

طبقة التطبيقات Applications layer

تعطي هيكلية (oneM2M) الأولوية للاتصالات بين الأجهزة والتطبيقات الخاصة بها. يحتوي هذا المجال على بروتوكولات طبقة التطبيق والتكامل مع أنظمة **ذكاء الأعمال** (Business Intelligence – BI).

طبقة الخدمات Services layer

يتم تمثيل هذه الطبقة أفقيًا عبر التطبيقات الخاصة بكل نوع من الصناعات المحددة. تتكون الوحدات الأفقية في هذا المستوى من الشبكة المادية التي تعمل عليها تطبيقات إنترنت الأشياء، وبروتوكولات الإدارة الأساسية، والأجهزة. من الأمثلة المهمة هنا: الاتصالات الخلوية، **والتبديل متعدد الاتفاقيات باستخدام المؤشرات التعريفية** (MPLS) **والشبكات الافتراضية الخاصة** (Virtual Private Networks – VPNs) **والشبكات المُعرَّفة بالبرمجيات** (SDNs) وغيرها، وتُعدُّ طبقة الخدمات المشتركة أعلى طبقة هنا.

التبديل متعدد الاتفاقيات باستخدام المؤشرات التعريفية (Multiprotocol Label Switching - MPLS):

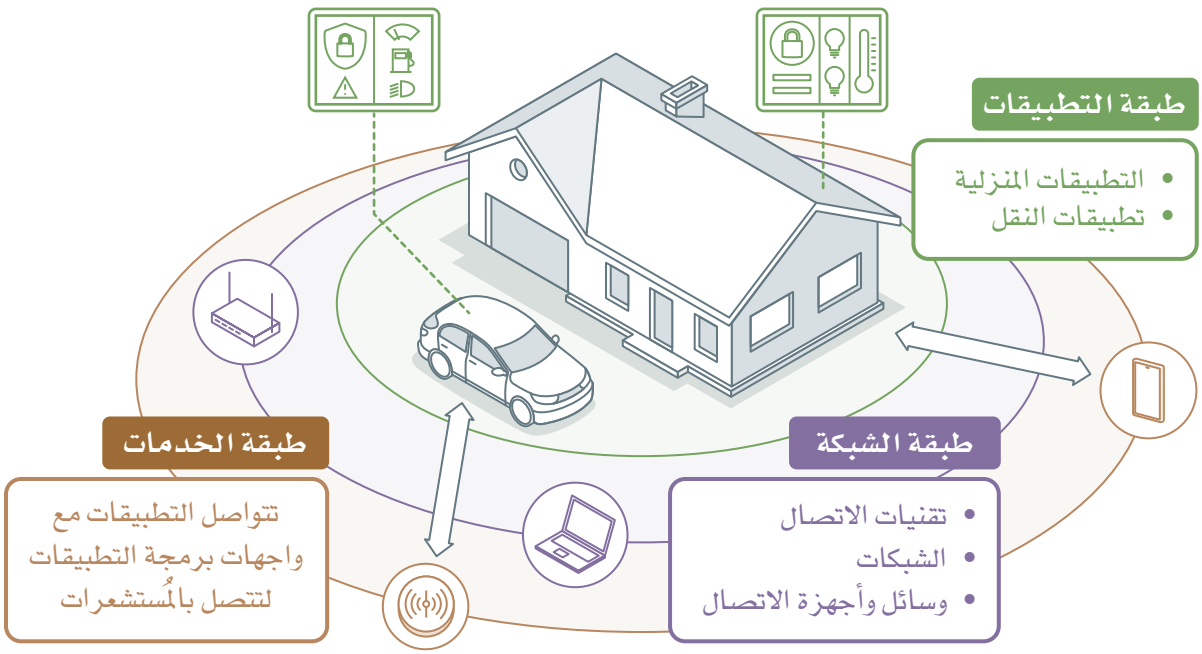
يعمل التبديل متعدد الاتفاقيات باستخدام المؤشرات التعريفية على توجيه البيانات بين العُقد بناءً على المؤشرات التعريفية والوسوم وليس عناوين الشبكة.

طبقة الشبكة Network layer

تُشكل هذه الطبقة مجال الاتصال بين أجهزة إنترنت الأشياء والنقاط النهائية، وتتكون طبقة الشبكة من كافة الأجهزة وشبكة الاتصالات التي تربط أنواعًا مختلفة من الشبكات، مثل الشبكات المتداخلة اللاسلكية وأنظمة النقطة إلى عدة نقاط.

نظام من نقطة إلى عدة نقاط (Point-to-multipoint system):

يوفر نظام نقطة إلى عدة نقاط مسارات مختلفة من عُقدة شبكية واحدة إلى عُقد وجهات متعددة.



شكل 5.13: طبقات هيكلية oneM2M

تتواصل الآلات الذكية وغير الذكية مع بعضها البعض بشكل متكرر، وفي بعض الحالات، يكون الاتصال من آلة إلى آلة غير ضروري، حيث تتصل الأجهزة فقط بتطبيقات خاصة بالاستخدام في مجال تطبيق إنترنت الأشياء عبر **شبكة منطقة ميدانية** (Field Area Network – FAN). تُعدُّ هذه الشبكة أكثر العناصر تعقيدًا في شبكة الاتصالات نظرًا لكونها مسؤولة بشكل أساسي عن توفير اتصالات **الميل الأخير** (Last-Mile) للأجهزة الطرفية. يتكون نطاق الجهاز أيضًا من جهاز البوابة الذي يوفر اتصالات بالشبكة الأساسية ويعمل كحد بين نطاقات الجهاز والشبكة.

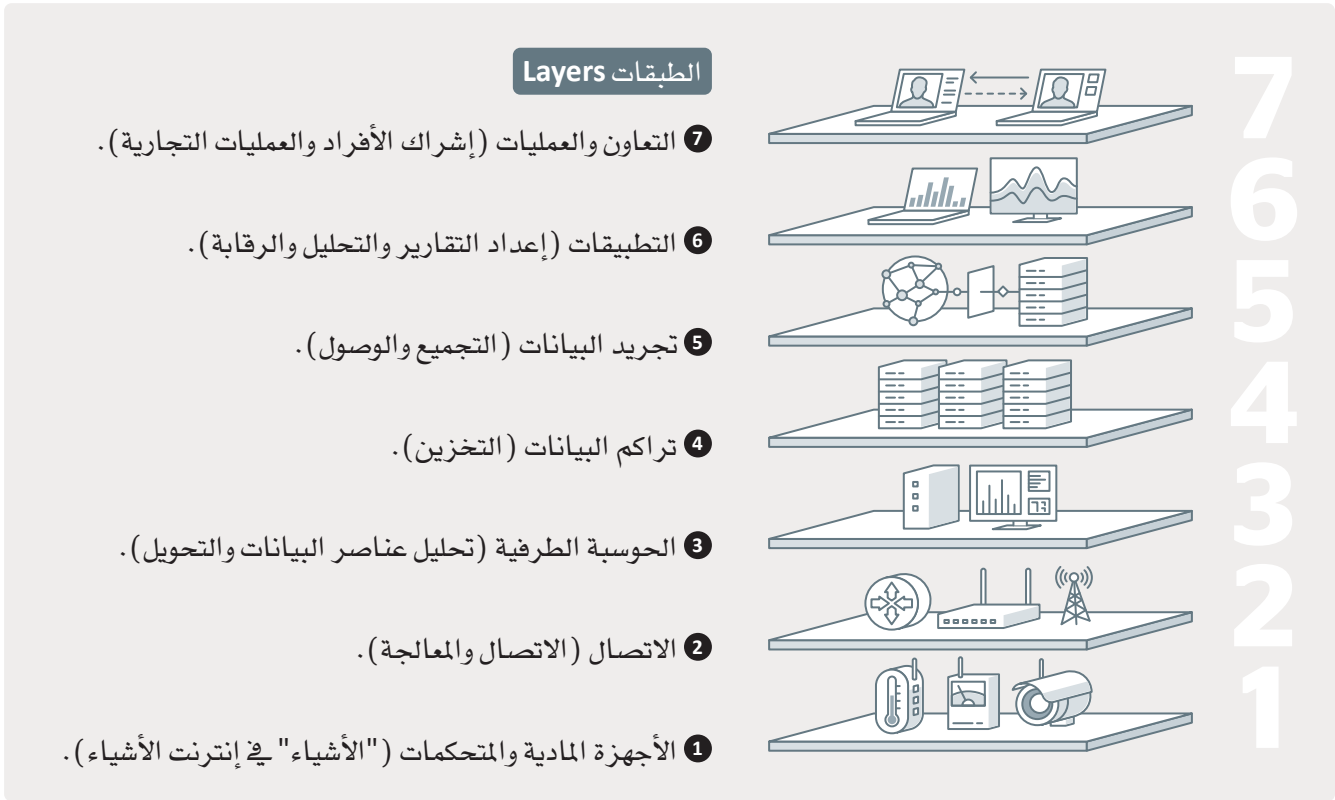
هيكلية أنظمة إنترنت الأشياء العالمي IoT World Forum Architecture

يُحدِّد النموذج المرجعي لإنترنت الأشياء –الذي قُدّم في المنتدى العالمي لإنترنت الأشياء – سلسلةً من المستويات مع تحكم رئيس من نقطة مركزية إلى طبقات الحافة، والتي تتكون من المُستشعرات والأجهزة والآلات وعُقد النهاية الذكية الأخرى. يمكن القول إنه بشكلٍ عام، تنتقل البيانات من الطبقات الطرفية إلى المركز.

يشبه النموذج المرجعي لإنترنت الأشياء نموذج شبكات الربط البيني للأنظمة المفتوحة (OSI).

يمكنك باستخدام هذا النموذج المرجعي تحقيق ما يلي:

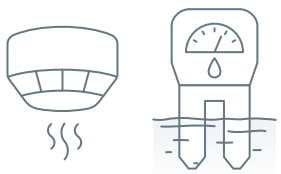
- تقسيم التحديات التي تواجه إنترنت الأشياء إلى مشاكل فرعية.
- تحديد التقنيات المختلفة في كل طبقة وطبيعة العلاقة بينها.
- تعريف نظام متكامل قائم على مكونات متعددة من مزودين مختلفين.
- تحديد **الواجهات** (Interfaces) بطريقة تعزز إمكانية التشغيل البيني.
- تحديد نموذج حماية متعدد الطبقات يُفرض في نقاط الانتقال لكل مستوى.



شكل 5.14: طبقات هيكلية إنترنت الأشياء العالمي

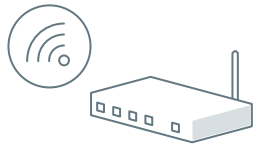
الطبقة الأولى: طبقة الأجهزة المادية والمتحكمات
Layer 1: Physical Devices and Controllers Layer

إن أول طبقة في النموذج المرجعي لإنترنت الأشياء هي طبقة الأجهزة المادية والمتحكمات. تحتوي هذه الطبقة على "الأشياء" الخاصة بإنترنت الأشياء، مثل الأجهزة الطرفية والمُستشعرات التي تُرسل البيانات وتستقبلها. يمكن أن تتراوح هذه "الأشياء" في حجمها من مُستشعرات صغيرة للغاية إلى آلات تصنيع ضخمة، والمهمة الرئيسة لهذه الطبقة هي إنتاج البيانات والسماح بالتحكم عبر الشبكة.



شكل 5.15: طبقة الأجهزة والمتحكمات

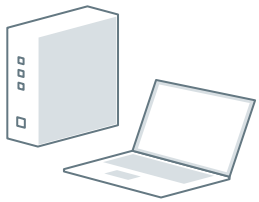
الطبقة الثانية: طبقة الاتصال Layer 2: Connectivity Layer



شكل 5.16: طبقة الاتصال

يتمثل دور طبقة الاتصال في نقل البيانات بطريقة موثوقة وفي الوقت المناسب، ويشمل هذا عمليات النقل بين أجهزة الطبقة الأولى والشبكة، وكذلك عمليات النقل بين الشبكة وطبقة الحوسبة الطرفية (معالجة معلومات الطبقة الثالثة). تشتمل طبقة الاتصال على جميع أجزاء الشبكات في إنترنت الأشياء، ولا تميز بين شبكة (الميل الأخير) – الشبكة بين المُستشعر أو نقطة النهاية وبوابة إنترنت الأشياء، والتي سيتم تناولها لاحقًا في هذا الفصل– وشبكة البوابة، والشبكة الرئيسة.

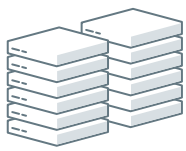
الطبقة الثالثة: طبقة الحوسبة الطرفية Layer 3: Edge Computing Layer



شكل 5.17: طبقة الحوسبة الطرفية

تلعب الحوسبة الطرفية دور الطبقة الثالثة في الهيكلية. تركز هذه الطبقة على تقليل البيانات وتحويل تدفقات بيانات الشبكة إلى معلومات جاهزة للتخزين والمعالجة بمستويات أعلى، وتتمثل إحدى الأهداف الأساسية لهذا النموذج المرجعي في بدء معالجة المعلومات بالقرب من حافة الشبكة بقدر الإمكان وبأسرع ما يمكن، كما تقوم الطبقة الثالثة أيضًا بفحص البيانات لمعرفة ما إذا كان يمكن تصفيتها أو تجميعها قبل نقلها إلى طبقة أعلى. يسمح هذا أيضًا بإعادة تنسيق البيانات أو فك تشفيرها، مما يُسهِّل المعالجة الإضافية بواسطة الأنظمة الأخرى.

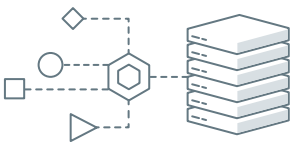
الطبقة الرابعة: طبقة تراكم البيانات Layer 4: Data Accumulation Layer



شكل 5.18: طبقة تراكم البيانات

يتم في هذه الطبقة التقاط وحفظ البيانات حتى تتمكن البرامج من الوصول إليها عند الضرورة، كما تُحوّل البيانات المستندة على الأحداث إلى تنسيقات يمكن الاستعلام عنها بواسطة خدمات أخرى.

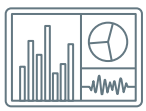
الطبقة الخامسة: طبقة تجريد البيانات Layer 5: Data Abstraction Layer



شكل 5.19: طبقة تجريد البيانات

في هذه الطبقة يتم التوفيق بين تنسيقات البيانات المتنوعة وضمان اتساق الدلالات من المصادر المتنوعة، وفي هذه الطبقة يتم التحقق باستخدام الحوسبة والمحاكاة الافتراضية من أن مجموعة البيانات تحتوي على بيانات كاملة، كما تُدمج البيانات في موقع واحد أو عدة مخازن بيانات.

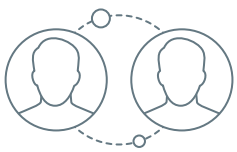
الطبقة السادسة: طبقة التطبيقات Layer 6: Applications Layer



شكل 5.20: طبقة التطبيقات

في هذه الطبقة تُستخدم البرامج لتفسير البيانات، حيث تتيح البرامج والتطبيقات مراقبة التقارير، وإنشاءها، وتنظيمها اعتمادًا على تحليل البيانات.

الطبقة السابعة: طبقة التعاون والعمليات Layer 7: Collaboration and Processes Layer



شكل 5.21: طبقة التعاون والعمليات

يتم هنا "استهلاك" وتوزيع بيانات التطبيق، وتتبع فائدة إنترنت الأشياء من حقيقة أن المشاركة في بيانات إنترنت الأشياء تتضمن في كثير من الأحيان العديد من الخطوات، ويمكن من خلال هذه الطبقة الحصول على بيانات تسهم في تغيير عمليات الشركة وتحسينها، وذلك بالاستفادة من مزايا إنترنت الأشياء.

بروتوكولات وشبكات الاتصالات قصيرة المدى
ShortRange Communication Network and Protocols

تحديد الترددات الراديوية والاتصال قريب المدى RFID and NFC

تُعدّ تقنية **تحديد الترددات الراديوية** (Radio Frequency Identification – RFID) وتقنية **الاتصال قريب المدى** (Near Field Communication – NFC) من تقنيات الاتصال التي تسمح بالاتصالات قصيرة المدى بين أجهزة إنترنت الأشياء والشبكة، ويتم استخدام تقنيتَي (RFID) و(NFC) لتخزين البيانات واستردادها عن بُعد، وتشتمل هذه التقنيات على جهاز إرسال وجهاز استقبال لاسلكي، حيث تستخدِم الحقول الكهرومغناطيسية للتعرف تلقائيًا وتتبع الرقاقات المُدمجة بالأشياء الذكية. تُرسل الرقاقة البيانات الرقمية وتستقبلها عندما يتم تنشيطها بواسطة نبضة كهرومغناطيسية تصدر من قارئ (RFID) أو (NFC) قربها. تتيح (RFID) تتبع الأدوات والمعدات والمواد في المخازن والمركبات والأشخاص، وذلك من خلال الرقاقات المُرفقة بها. يُمكن لأجهزة قراءة الرقاقات قراءة الرقاقة القريبة منها، حتى لو لم تكن مرئية، كما يمكن قراءة عدد كبير من الرقاقات في ذات الوقت سواء كانت ظاهرة أو مخفية داخل صندوق أو حاوية مثلًا، وذلك خلافًا **للرموز الشريطية - الباركود** (Barcodes)، والتي يجب أن تكون ظاهرةً أمام جهاز القراءة ولا يمكن قراءتها إلا واحدة تلو الأخرى. تُستخدم تقنية (NFC) على نطاقٍ واسع لتبادل البيانات بين الأجهزة في نطاق يبلغ حوالي 4 سنتيمترات، وتُستخدم هذه التقنية في عمليات الدفع غير التلامسية ببطاقات الائتمان، وكبديل لمفاتيح المكاتب التقليدية وغرف الفنادق، وفي ربط إعداد بعض الأجهزة مثل سماعات الرأس، ويتمثل الاختلاف الرئيس بين تقنيتَي (RFID) و (NFC) في أن (NFC) صُمِّمت لتبادل البيانات بشكل آمن، مما يجعلها مناسبة للمعاملات المالية، بينما تُستخدم (RFID) بصورة أساسية في التطبيقات التي نحتاج فيها إلى تحديد عناصر معينة لاسلكيًا.

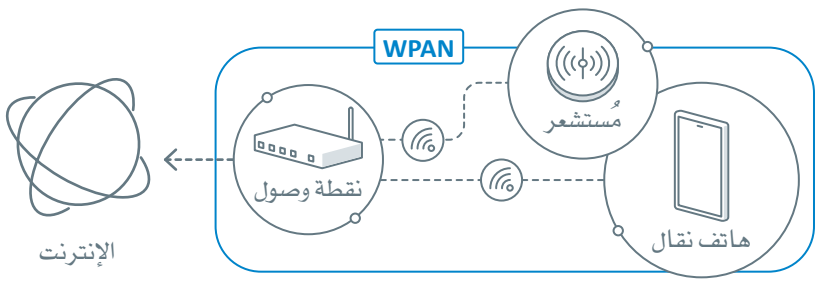
جدول 5.2: مقارنة بين (RFID) و(NFC)

الخاصية	RFID	NFC
تردد الاستخدام	125 كيلوهيرتز إلى 2.45 جيجاهيرتز.	13.56 ميجاهيرتز.
نطاق الاتصال	بحد أقصى 100 متر.	في حدود 10 سم (مسافة قصيرة).
الاتصال	اتصال أحادي الاتجاه.	اتصال ثنائي الاتجاه.
الميزة	التعرف خلال مسافات بعيدة.	أمان عالي.

شبكات المنطقة الشخصية اللاسلكية وبروتوكولاتها
Wireless Personal Area Networks (WPANS) and Protocols

تتطلب المُستشعرات والكائنات الأخرى المتصلة بالإنترنت وسيلة معينة لنقل البيانات واستقبالها. سيتم في هذا الدرس تناول **شبكات المنطقة الشخصية** (PAN) وتقنية الاتصالات بعيدة المدى. يمكن للمُستشعرات والمُشغِّلات في بيئة إنترنت الأشياء الاتصال عبر الأسلاك أو من خلال **شبكات المنطقة الشخصية اللاسلكية** (WPAN).

شبكة المنطقة الشخصية (Personal Area Network - PAN):
شبكة المنطقة الشخصية هي إحدى شبكات الحاسب المستخدمة لتوصيل الأجهزة الإلكترونية داخل مساحة عمل المستخدِم.



شكل 5.22: شبكة المنطقة الشخصية اللاسلكية

بروتوكولات شبكات المنطقة الشخصية اللاسلكية غير المستندة إلى عنوان

Non-IP Based WPANS Protocols

زيجبي Zigbee

يُعدُّ بروتوكول زيجبي أحد بروتوكولات (WPAN) القائم على أساس معيار (IEEE 802.15.4) الذي صُمّم لشبكات إنترنت الأشياء التجارية والسكنية ذات التكلفة والطاقة والمساحة المحدودة. يمكن لزيجبي تكوين الشبكات، واكتشاف الأجهزة، وتأمين وإدارة الشبكة، ولكن بروتوكول زيجبي لا يوفر خدمات نقل البيانات أو بيئة لتنفيذ تطبيقات معينة. تُعدُّ زيجبي **شبكة متداخلة** (Mesh Network) ذاتية الإصلاح، ويوضِّح الجدول الآتي المكونات الرئيسة لهذه الشبكة:

جدول 5.3: المكونات الرئيسة لشبكة زيجبي

المُكون	الوصف
مُتحكم زيجبي (Zigbee Controller – ZC)	جهاز عالي القدرة يُستخدم لبناء وظائف الشبكة والبدء بها على شبكة زيجبي، قادر على تعيين عناوين الشبكة المنطقية والسماح للعُقد بالانضمام إلى الشبكة أو مغادرتها.
مُوجه زيجبي (Zigbee Router – ZR)	يُعالج هذا المُكون الاختياري جزءًا من الشبكة المتداخلة عن طريق تعيين عناوين الشبكة المنطقية والسماح للعُقد بالانضمام إلى الشبكة أو الخروج منها.
جهاز زيجبي طرفي (Zigbee End Device – ZED)	يُعدُّ هذا الجهاز بمثابة نقطة نهاية بسيطة ومباشرة ذات قدرة على التواصل مع الوسيط. من هذه الأجهزة مفتاح الإضاءة ومُنظم الحرارة.

يُعالج زيجبي ثلاثة أنواع مُختلفة من حركة البيانات:

مُستشعر

1. البيانات الدورية: يُحدِّد معدل التسليم الدوري للبيانات أو إرسالها من خلال التطبيقات (على سبيل المثال المُستشعرات التي تُرسل البيانات بصورة دورية). تُنتج بيانات متقطعة عند حدوث تطبيق أو مُحفزات خارجية بوتيرة عشوائية.

مفتاح الإضاءة

2. البيانات المُتقطعة: يُعدُّ مفتاح الإضاءة مثالًا جيدًا على البيانات المتقطعة المثالية لزيجبي.

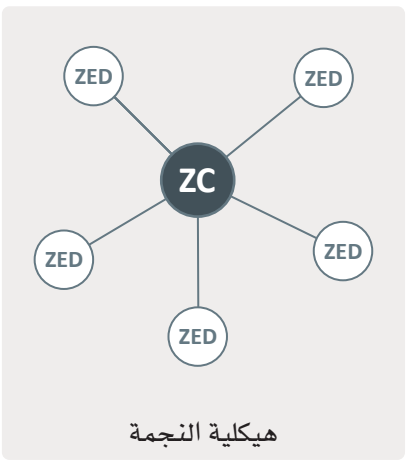
الفأرة

3. بيانات زمن الانتقال المنخفض المتكررة: يُعيّن زيجبي فترات زمنية للإرسال، ويمكن أن يكون بزمن انتقال منخفض جدًا، مما يجعله مناسبًا لأجهزة الفأرة ولوحات المفاتيح.

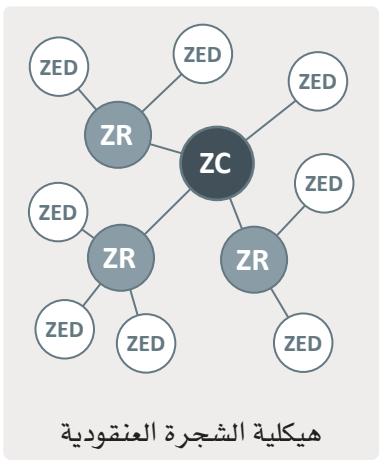
توجد ثلاث هيكليات أساسية لزيجبي:

جدول 5.4: هيكليات زيجبي

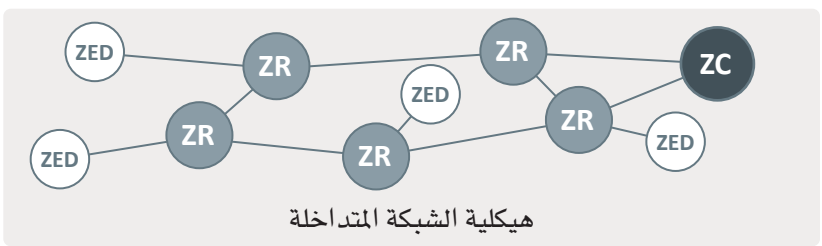
الهيكلية	الوصف
هيكلية النجمة (Star Topology)	يحتوي مُتحكم زيجبي على واحد أو أكثر من أجهزة الزيجبي. يمتد إلى نقطتين فقط، مما يَحد المسافة بين العُقد. ويتطلب أيضًا وسيلة ارتباط يمكن الاعتماد عليها مع نقطة عُطل مفردة في متحكم زيجبي.
هيكلية الشجرة العنقودية (Cluster Tree topology)	وهي شبكة **متعددة النقاط** (Multi-Hop) تستخدِم أجهزة **الإرشاد** (Beacons) لتوسيع التغطية والمدى. تُعدُّ أجهزة الزيجبي بمثابة نقاط نهاية، ويمكن أن تحتوي عقدة **مُوجه الزيجبي** (ZR) وعقدة **مُتحكم الزيجبي** (ZC) على عُقد فرعية. تتواصل العُقد الفرعية مع العُقد الرئيسة فقط، ويمكن للعُقد الرئيسة التواصل مع العُقد الفرعية **لأعلى** (Upstream) أو **لأسفل** (Downstream) منها، وتُشكِّل **نقطة العُطل المركزية** (Central Ffailure Point) مشكلة في هذه الهيكلية.
هيكلية الشبكة المتداخلة (Mesh Topology)	يمكن توجيه أي جهاز مصدري إلى أي جهاز بصفته وجهة، وذلك باستخدام طرائق **التوجيه المستند إلى الأشجار** (Tree-Based Routing) و**التوجيه المستند إلى الجداول** (Table-Based Routing). يجب تشغيل موجات متحكمات الزيجبي وموجهاتها طوال الوقت لتنفيذ وظائف التوجيه، مما يؤدي إلى استنزاف عمر البطارية. يُسمح للموجهات الموجودة في نطاقٍ محدد بالتفاعل بشكل مباشر، وتكمن الفائدة الأساسية في هذه الهيكلية في إمكانية توسُّع الشبكة ووجود مسارات متعددة للبيانات.



هيكلية النجمة



هيكلية الشجرة العنقودية



هيكلية الشبكة المتداخلة

شكل 5.23: هيكليات زيجبي

القفزة (Hop):
تحدث القفزة عندما تُمرّر حزمة من قطاع في الشبكة إلى قطاعٍ آخر.

المنارة (Beaconing):
ترمز المنارة في الشبكات إلى ما يشبه جهاز الإرشاد، وذلك عبر القيام بالبث الرقمي بصورة دورية.

البلوتوث Bluetooth

البلوتوث هي تقنية اتصال لاسلكية منخفضة الطاقة تُستخدم على نطاق واسع في الأجهزة الإلكترونية مثل: الهواتف المحمولة ووحدات التحكم في الألعاب ولوحات المفاتيح، وقد استُخدم البلوتوث على نطاق واسع في إنترنت الأشياء لإرسال الإشارات عند تشغيلها في وضع **الطاقة المنخفض** (Low Energy – LE)، وذلك في أجهزة **الإرشاد** (Beacons)، والمُستشعرات اللاسلكية وأنظمة تتبع المركبات والأصول الأخرى، وأجهزة التحكم عن بُعد، وأجهزة المراقبة الصحية، وأجهزة الإنذار. تتميز شبكات البلوتوث الشخصية اللاسلكية بحصول ما يطلق عليه اسم: **الأحداث** (Events)، ويُعدّ **الإعلان** (Advertising) و**التوصيل** (Connecting) بمثابة الحدثين الرئيسين في تلك الشبكات.

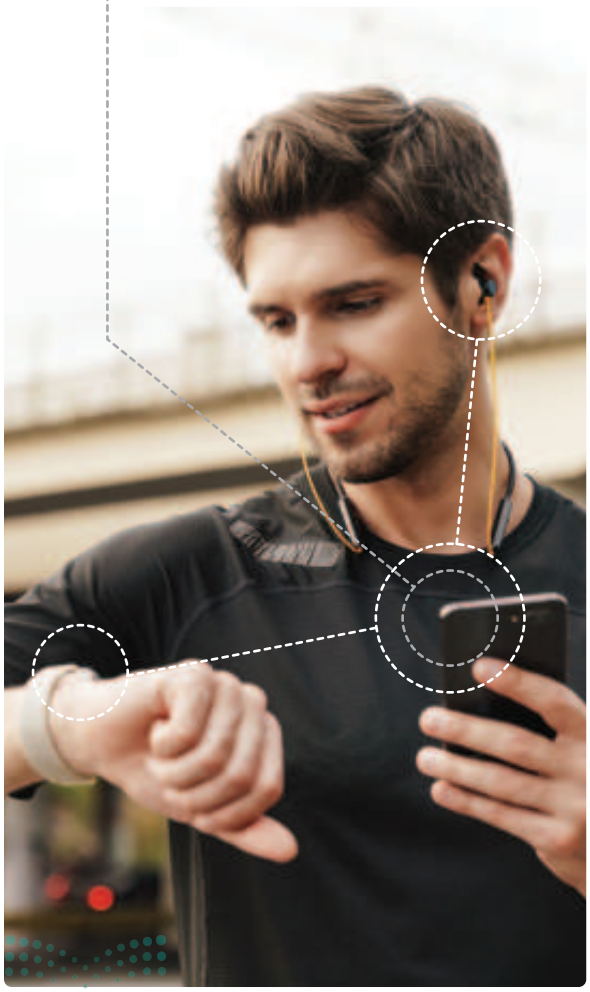
الإعلان Advertising

يبدأ الإعلان بوجود جهاز يطلب القيام بعملية اقتران مع الأجهزة الأخرى التي تقوم بالمسح، أو بإرسال رسالة تحتوي على معلومات التعارف.

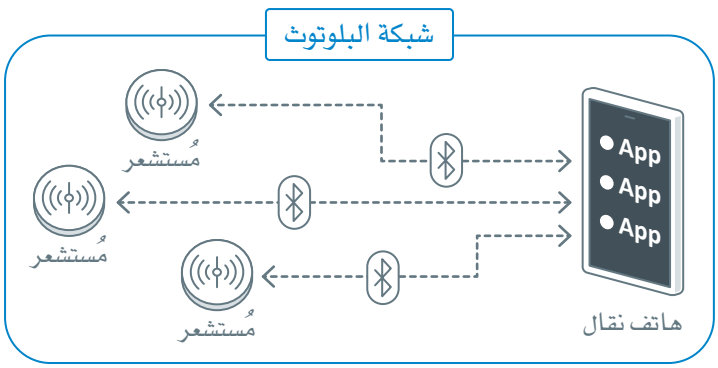
التوصيل Connecting

يصف هذا الحدث عملية اقتران الجهاز بجهازٍ آخر يسمى بالمُضيف.

يمكن للجهاز في وضع الطاقة المنخفضة إجراء اتصال كامل باستخدام قناة الإعلان فقط. قد يكون هذا الاتصال أيضًا اتصالًا رسميًا ثنائي الاتجاه بين الأجهزة، وستبدأ الأجهزة حينها بالقيام بما يسمى بإجراء التكوين لإجراء الاتصال، وذلك من خلال الاستماع إلى حزم الإعلانات. في هذه الحالة، يُعدّ الجهاز المُستمع بادئًا، وإذا أرسل المُعلن حدثًا إعلانيًا قابلًا للاتصال، يجوز للجهاز البادئ إرسال طلب اتصال باستخدام نفس القناة التي استُلمت الحزمة الإعلانية من خلالها. يمكن للمُعلن أن يقرر بعد ذلك ما إذا كان سيُنشئ الارتباط أم لا، وإذا أُنشئ الارتباط، ينتهي الحدث الإعلاني، وحينها يُشار إلى البادئ باسم: رئيسي (Master) والمُعلن باسم: **فرعي** (Slave). يطلق على مثل هذا الاتصال بمصطلحات البلوتوث تسمية **بيكونيت** (Piconet)، حيث تحدث أحداث الاتصال بين الجهاز الرئيس والفرعي على نفس قناة البداية، وبعد نقل البيانات وانتهاء حدث الاتصال، يمكن تغيير التردد لتحديد قناة جديدة للمُرسل والمُستقبل.



شكل 5.25: توصيل البلوتوث



شكل 5.24: شبكة البلوتوث

بروتوكولات شبكات المنطقة الشخصية اللاسلكية IP Based WPANS Protocols

الاصدار السادس لبروتوكول الإنترنت عبر شبكات المنطقة الشخصية اللاسلكية منخفضة الطاقة 6LoWPAN

تُصمّم شبكات (IP) عبر أنظمة اتصالات ترددات لاسلكية منخفضة الطاقة لتعمل مع الأجهزة ذات الطاقة والقدرات المحدودة التي لا تتطلب خدمات شبكات ذات نطاق ترددي عال. يتوافق هذا البروتوكول مع العديد من اتصالات شبكات (WPAN) بما فيها معايير (IEEE.802.15.4)، وتقنيات **البلوتوث** (Bluetooth)، وتقنيات **الترددات اللاسلكية الفرعية واحد جيجاهرتز** (sub-1 GHz RF)، بالإضافة إلى **الاتصالات عبر خطوط الكهرباء** (Power Line Controller – PLC). تتمثل الميزة الأساسية لبروتوكول (6LoWPAN) في أن معظم المُستشعرات الأساسية تعمل بتوافق مع نظام عنونة (IP) وبذلك يمكنها أن تعمل كعناصر في الشبكة عبر موجهات الشبكة المحلية أو الواي فاي أو شبكات الجيل الثالث وشبكات (LTE)، وشبكات الجيل الرابع. يمكن لعنونة (IPV6) تغطية ما يصل إلى 50 مليار جهاز متصل بالإنترنت، مما يسمح لها بالاستمرار كنظام للعنونة في المستقبل، وبالتالي إتاحة التوسع المطلوب في نشر إنترنت الأشياء.

تُعدّ شبكات (6LoWPAN) شبكات متداخلة تبنى على جوانب شبكات أكبر. تتميز هذه الشبكات بهيكلياتها المرنة مما يسمح بوجود شبكات **مخصصة** (Ad hoc) و**مفككة** (Disjoined) دون اشتراط الارتباط بالإنترنت أو بأنظمة أخرى. يمكن لهذه الشبكات الارتباط بالشبكة الرئيسة أو بالإنترنت من خلال ما يسمى **بموجهات طرفية** (Edge Routers). يمكن للموجهات الطرفية المختلفة توصيل شبكات (6LoWPAN) متعددة من خلال ما يُعرف باسم **التوجيه المتعدد** (Multi-Homing)، ويمكن إنشاء الشبكات المخصصة دون الحاجة إلى الوصول إلى الإنترنت من الموجه الطرفي، حيث تُنشئ الموجهات الطرفية شبكات (6LoWPAN) متداخلة على محيط الشبكات التقليدية الأكبر حجمًا، ويمكنها أيضًا تسهيل تبادلات عناوين (IPV6) إلى (IPV4) عند الضرورة. يتم التعامل مع حزم البيانات بشكل مشابه لشبكة عناوين (IP)، والتي تُقدِّم بعض المزايا مقارنة بالبروتوكولات المعروفة الأخرى. تشترك جميع العُقد داخل شبكة (6LoWPAN) في بادئة (IPv6) التي أنشأها الموجه الطرفي. يُسجّل العُقد مع الموجهات الطرفية بشكل مستمر خلال مرحلة اكتشاف الشبكة. تحكم مرحلة اكتشاف الشبكة التفاعل بين المضيفين والموجهات في شبكة (6LoWPAN) المحلية. وتُمكّن خاصية **التوجيه المتعدد** (Multi-Homing) عدة موجهات (6LoWPAN) من تشغيل الشبكة؛ على سبيل المثال، عندما يتطلب تجاوز العطل أو التجاوز عن الخطأ وسائط مختلفة (الجيل الرابع وواي فاي).

بروتوكول التشعب Thread

التشعب هو بروتوكول لشبكات إنترنت الأشياء يعتمد على (6LoWPAN) IPV6. الهدف الأساسي لهذا البروتوكول هو إتاحة أتمتة المنازل والشبكات المنزلية، ويمكن وصف التشعب بأنه عنوان (IP) يستند إلى معايير وهيكلية (IEEE 802.15.4) و(6LoWPAN). يتشابه التشعب مع زيجبي ونسخ 802.15.4 الأخرى، ولكنه يختلف من حيث قابليته لعنونة (IP). يعتمد هذا البروتوكول على البيانات والطبقات المادية للمعايير التقنية 802.15.4 وخصائص الأمان والتوجيه لشبكات (6LoWPAN). يعتمد التشعب أيضًا على هيكلية **الشبكة المتداخلة** (Mesh Network) مما يجعله خيارًا عمليًا لأنظمة الإضاءة المنزلية الذكية، وذلك بسعة تصل إلى 250 جهاز لكل شبكة. يتميز بروتوكول التشعب بقابلية عنونة (IP) في الأجهزة المختلفة بما فيها المُستشعرات الصغيرة للغاية، وأنظمة تشغيل الأتمتة المنزلية، كما يتميز بتوفيره في استهلاك الطاقة؛ لأن البروتوكول لا يتطلب استمرار تنفيذه في طبقة الشبكة. يعني هذا أيضًا أن الموجه الطرفي الذي يستضيف الشبكة المتداخلة لا يحتاج إلى التعامل مع بروتوكولات طبقة التطبيقات، مما يحد من متطلبات الطاقة والمعالجة، ويُعدُّ هذا البروتوكول آمنًا جدًا نظرًا لكونه متوافقًا مع (IPV6) ولكون جميع الاتصالات مُشفرة باستخدام **معيار التشفير المتقدم** (Advanced Encryption Standard – AES).

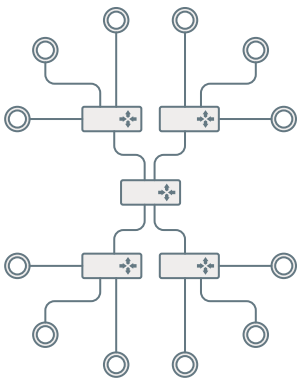
شكل 5.26: شبكات المنطقة الشخصية اللاسلكية

شبكات وبروتوكولات الاتصالات بعيدة المدى
Long Range Communication Networks and Protocols

تربط **شبكات المنطقة الشخصية اللاسلكية** (WPAN) و**شبكات المنطقة المحلية اللاسلكية** (Wireless Local Area Networks – WLAN) المُستشعرات بشبكة محلية، ولكن ليس بالضرورة بشبكة الإنترنت أو بأنظمة الشبكات الأخرى. تشمل بيئة إنترنت الأشياء المُستشعرات والمُشغِّلات والكاميرات والأدوات الذكية المُدمجة والمركبات، وحتى الروبوتات التي تعمل في الأماكن النائية. لقد أصبح من المُسلَّم به أنّ علينا الاعتياد على التعامل مع **شبكة المنطقة الواسعة** (Wide Area Network – WAN) على المدى الطويل.

تقنية لوراوان LoRaWAN

تُعدُّ التقنيات اللاسلكية **منخفضة الطاقة واسعة النطاق** (Low Power Wide Area – LPWA) مثالية لنقاط النهاية (الأجهزة المختلفة) طويلة المدى التي تعمل بالبطارية. عادةً ما يشار إلى هيكلية (LoRaWAN) باسم هيكلية **نجمة النجوم** (Star of Stars). تقوم نقاط النهاية بتبادل الحزم عبر بوابات تعمل كجسور، وذلك بوجود خادم شبكة (LoRaWAN) مركزي. تتصل نقاط النهاية مباشرة بواحدة أو بأكثر من البوابات، بينما تتصل المداخل **بالشبكة الخلفية** (Backend Network) عبر اتصالات (IP) العادية. يمكن في هذه التقنية استلام الحِزم نفسها ونقلها بواسطة العديد من البوابات، وفي حال تلقي حزم مُكررة، يكون خادم الشبكة مسؤولًا عن إلغاء التكرار، وتوفر تقنيات (LPWA) المفتوحة المتوفرة خيارات جديدة لشبكات الشركات الخاصة والبث ومُقدِّمي الخدمات المتنقلة وغير المتنقلة لنشر البُنى التحتية لإنترنت الأشياء وحلولها. تتوسع بيئة نقاط النهاية بسرعة، وستكون بلا شك العامل الحاسم بين تقنيات وحلول التقنيات اللاسلكية **منخفضة الطاقة واسعة النطاق** (LPWA) المختلفة مثل (LoRaWAN)، ويُعدّ بناء البنى التحتية وتطويرها المحلية منها والإقليمية أمرًا حيويًا لتفعيل استخدام إنترنت الأشياء على نطاق استهلاكي أوسع، ويتحمل مسؤولية ذلك الأشخاص المسؤولون عن المدن الذكية، وهيئات تنظيم البث والإذاعة، ومقدمو خدمات الاتصالات الخلوية والعادية.



شكل 5.27: هيكلية LoRaWAN "نجمة النجوم"

الشبكات الخلوية (الجيل الخامس) Cellular Networks (5G)

من أكثر أنواع الاتصالات شيوعًا استخدام الترددات الخلوية خاصةً البيانات الخلوية. فقبل تطور التقنية الخلوية، كانت تغطية أجهزة الاتصالات المحمولة محدودة، واستُخدمت مساحة ترددات مشتركة، فالأجهزة كانت بمثابة أجهزة إرسال لاسلكي ثنائية الاتجاه. ثم أصبحت الشبكات الخلوية ممتازة في نقل البيانات في كلا الاتجاهين بسرعات عالية، ولكن على حساب النطاق واستهلاك البطارية. يُعدُّ **الجيل الخامس** (5G) الجيل التالي من تقنية الاتصالات القائمة على بروتوكول الإنترنت والتي يتم تطويرها لتحل محل شبكات الجيل الرابع الخلوية، بالإضافة إلى ذلك تعمل شبكات الجيل الخامس على تحسين النطاق الترددي ووقت الاستجابة والكثافة وتقليل نفقات المستخدِم، وتهدف إلى أن تكون معيارًا شاملًا واحدًا يشمل جميع الخدمات والفئات الخلوية، بدلًا من إنشاء خدمات وتصنيفات مميزة لكل حالة استخدام.

جدول 5.5: السمات الرئيسة لشبكات الجيل الخامس الحديثة

الميزات	الوصف
	النطاق العريض المتنقل المحسّن (Enhanced Mobile Broadband - eMBB)
	اتصالات فائقة الموثوقية وذات زمن انتقال منخفض (Ultra-Reliable and Low-Latency Communications - URLLC)
	اتصالات نوع الآلة الضخمة (Massive Machine Type Communications - mMTC)

تمرينات

1

حدِّد الجملة الصحيحة والجملة الخاطئة فيما يلي:	صحيحة	خاطئة
1. تحتوي هيكلية شبكة oneM2M على طبقة بيانات.	○	○
2. يمكن استخدام خدمات الشبكة الافتراضية الخاصة (VPN) في طبقة الخدمات لهيكلية (oneM2M).	○	○
3. يمكن أن تحتوي طبقة التطبيقات على خدمات المراقبة في أنظمة إنترنت الأشياء العالمي.	○	○
4. تُستخدم تقنيات (NFC) للاتصالات بعيدة المدى بين الأجهزة.	○	○
5. يتصل بروتوكول زيجبي عبر قنوات شبكة ببروتوكول (UDP).	○	○
6. يُعدُّ موجه زيجبي مسؤولًا عن خصائص الإصلاح الذاتي للشبكات المتداخلة.	○	○
7. يُرسل الحدث الإعلاني لاتصالات البلوتوث حزم بيانات إلى الأجهزة المجاورة.	○	○
8. لا يُعدُّ التشعب (Thread) بروتوكولًا قائمًا على الشبكة.	○	○
9. لا تحتاج أنظمة شبكات المدن الذكية إلى شبكات وبروتوكولات اتصالات بعيدة المدى.	○	○
10. تُصنَّف شبكات الجيل الخامس (5G) بأنها منخفضة استهلاك الطاقة.	○	○

2 صنِّف الطبقات الرئيسة لهيكلية (oneM2M) لأنظمة إنترنت الأشياء.

3 حلِّل الطبقات الرئيسة لهيكلية أنظمة إنترنت الأشياء العالمي.

4 حدِّد الخصائص الرئيسة لتقنية تحديد الترددات الراديوية (RFID) وتقنية الاتصال قريب المدى (NFC).

5 صنِّف النوعين الرئيسين لشبكات المنطقة الشخصية اللاسلكية (WPANS)، واعرض بعض الأمثلة على كل نوع.

6 حدِّد المكونات الرئيسة الثلاثة لشبكة زيجبي (Zigbee).

7 قارن بين الحدثين الأساسيين اللذين يحدثان أثناء الاتصال بالبلوتوث.

8 قدِّم وصفًا لبروتوكولي (WPANS) الرئيسين المُستندين إلى عنونة (IP).

9 وضِّح هيكلية "نجمة النجوم" التي تستخدمها شبكات (LoRaWAN).

10 قدِّم تحليلًا لكيفية تطور شبكات الجيل الخامس (5G) من تقنيات شبكات الجيل الرابع (4G).

رابط الدرس الرقمي

www.ien.edu.sa

الدرس الثالث

أمان وخصوصية أنظمة إنترنت الأشياء

الأمان Security

يُشكِّل الإنترنت وأنظمة إنترنت الأشياء والخدمات السحابية والأنظمة الإلكترونية الملموسة (CPSs) والأجهزة المحمولة ملامح الحياة الحديثة في القرن الحادي والعشرين، فقد أسهمت التقنية في التواصل بين أطراف العالم بما يعود بالفائدة على كافة المجتمعات. ولكن هذا التطور التقني أدّى إلى تَمكُّن مجرمي الإنترنت من استغلال العديد من نقاط الضعف لتهديد مستخدِمي هذه التقنيات. يتزايد تأثير إنترنت الأشياء على المؤسسات ونماذج الأعمال، ويعتمد إنترنت أشياء الشركات على ثقة المستهلك. ومع ذلك تُقدّم العديد من المنتجات والخدمات التقنية إلى الأسواق بصورة متسارعة مع اهتمام غير كافٍ بأمان وخصوصية المستخدِمين، فالأمان يُعدُّ جزءًا مُهمًا من عملية التصميم من أدنى المستويات، إلى أعلاها؛ ولهذا يجب إنشاء السياسات والبروتوكولات والمعايير الأمنية بموازاة أي تطور تقني لدعمه وحمايته.

يعرض الجدول الآتي أُسس الأمان في إنترنت الأشياء:

النظام الإلكتروني الملموس (Cyber Physical System - CPS):
هو نظام محوسب يتحكم أو يراقب آلية معينة باستخدام خوارزميات محوسبة.

جدول 5.6: أُسس الأمان في إنترنت الأشياء

الأساس	الوصف
الثقة	السماح للمستخدِمين أو الخدمات المُصرَّح لها فقط بالوصول إلى الجهاز أو البيانات.
التحقق من الهوية	التحقق من هوية الأفراد والخدمات و"الأشياء".
الخصوصية	الحفاظ على خصوصية جهاز المستخدِم ومعلوماته الشخصية وبياناته الحساسة.
الحماية	حماية الأجهزة والمستخدِمين من الأضرار المادية والمالية والمتعلقة بالسمعة.

تحديات أنظمة إنترنت الأشياء المُرتكزة على المستخدِم

User-Centered Challenges of IoT Systems

شكل 5.28: المساءلة في إنترنت الأشياء

أصبحت التدابير الأمنية التقليدية غير كافية لتوفير الأمان الكافي للعالم الحديث المتطور تقنيًا والمُتصل معًا، فعلى النقيض من الأجهزة الإلكترونية التقليدية، فإنّ أجهزة إنترنت الأشياء تتفاعل معًا ومع الخدمات على الإنترنت. لا يمكن تحقيق الفوائد المرجوة من تطبيق الأنظمة والتقنيات الحديثة دون الحصول على ثقة المستخدِمين بهذه التقنيات الحديثة؛ لذلك تُعدّ المساءلة أمرًا بالغ الأهمية لزرع الثقة بين المستخدِمين ومُنشئي أنظمة إنترنت الأشياء، ويسهم تعقيد تدفق البيانات الموزعة، وآليات التوافق غير الكافية، ونقص المعلومات بالنسبة للمستخدِمين في ظهور الحاجة إلى وجود نظام للمساءلة في إنترنت الأشياء.

الأمان في إنترنت الأشياء والجرائم الإلكترونية

IoT Security and Cybercrime

تُعدُّ البُنية التحتية للإنترنت بمثابة المُنشآت الحيوية داخل الحدود الإقليمية للدول ذات السيادة. تَمرُّ البيانات المتدفقة عبر هذه البنية التحتية عبر العديد من الدول الأخرى، مما يشكِّل مصدرًا للقلق فيما يتعلق بسلامة البيانات، فالجرائم الإلكترونية لا تعرف الحدود، بل وتتجاوز الحدود الجغرافية بسهولة. ومن الملاحظ أن القوانين المتعلقة بحماية البيانات وأمن المعلومات تتباين بين الدول المختلفة؛ لذا، تمثِّل الفجوة الواسعة بين التشريعات القانونية والتقنية عقبة رئيسة في مكافحة الجريمة الإلكترونية، ويواجه نظام العدالة لمكافحة هذه الجرائم تحديات كثيرة ويتسم بالبطء وعدم القدرة على تنظيم هذا الفضاء الإلكتروني. كما أن سرعة تبني التقنية في المجتمعات تفوق السرعة التي تُوضع بها السياسات والتشريعات لتنظيم وضبط هذه التقنية؛ لهذا السبب، يُتحكّم في الفضاء الإلكتروني والتقنية من خلال دمج مجموعة من القوانين غير المُلائمة والقديمة والمتناقضة أحيانًا، كما يصعب تحقيق توافق دولي أو إقليمي في الآراء حول معايير وقوانين السلامة الإلكترونية نظرًا لأن لكل دولة معاييرها ومعتقداتها وممارساتها المستقلة، مما يقرر رؤى مختلفة لعملية تنظيم الفضاء الإلكتروني. تعزز بعض الدول سيادتها على الفضاء الإلكتروني بحجة أن سياساتها الوطنية العامة تنطبق على الفضاء الإلكتروني أيضًا، وبأن الدولة يجب أن تكون قادرة على تنظيم كيفية استخدام الأفراد والشركات للإنترنت داخل حدودها.

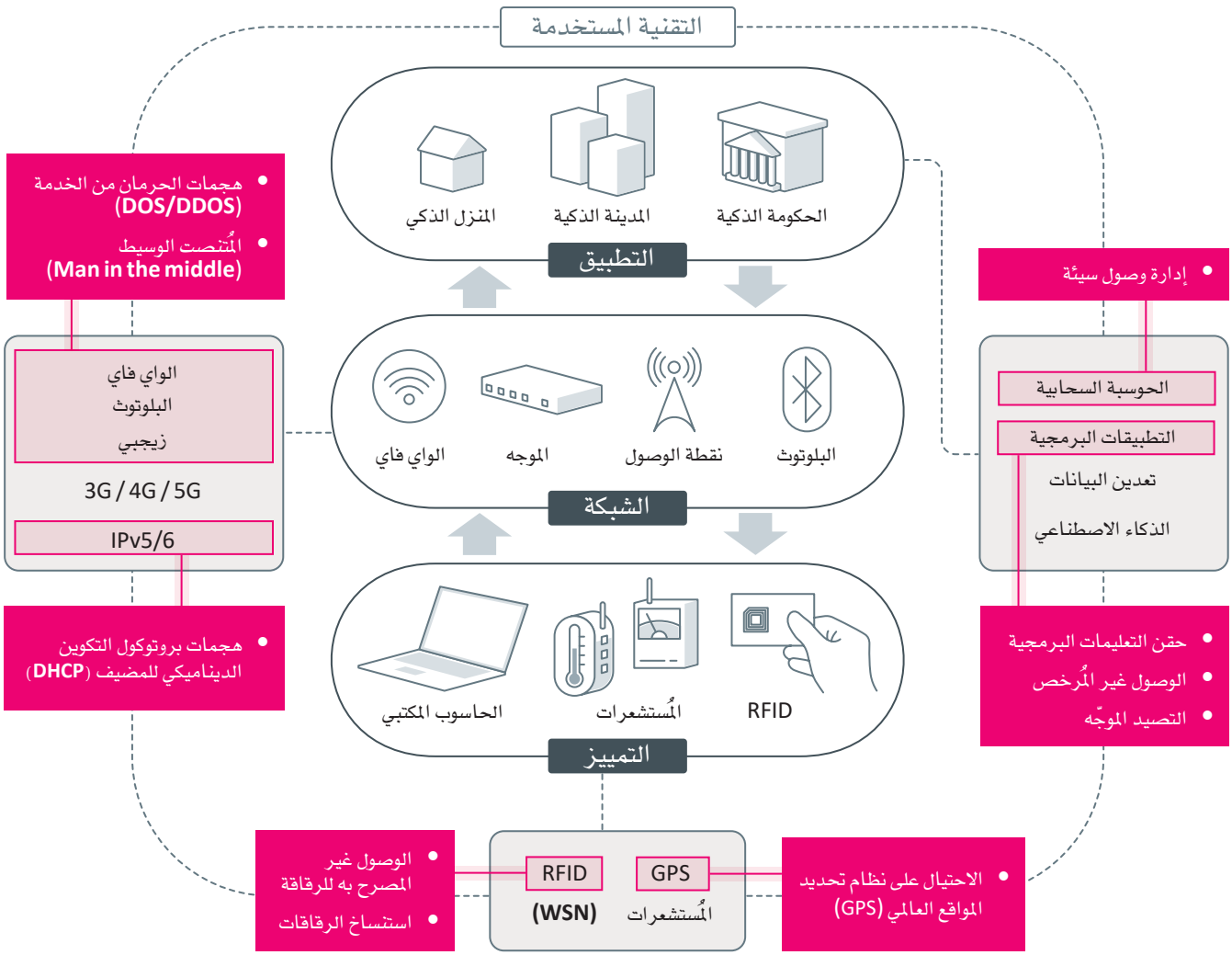
شكل 5.29: هجوم واختراق برامج الفدية الضارة

التحديات الهيكلية للأمان في إنترنت الأشياء

Architectural Challenges of IoT Security

يتطلب إنترنت الأشياء مجموعة من المعايير ووجود هيكلية مُحددة جيدًا مع واجهات ونماذج بيانات وبروتوكولات تناسب التنوع في الأجهزة والبروتوكولات والخدمات المَعنية. يمكن حدوث العديد من الهجمات عند اتصال أجهزة إنترنت الأشياء بالخدمات السحابية وتبادلها للبيانات لأول مرة، كما قد تتسبب خصائص أجهزة إنترنت الأشياء المختلفة في وجود مخاطر ومشكلات أمنية، حيث تقدم قابلية التنقل والاعتماد المتبادل والخصائص المماثلة الأخرى تحديات ومخاطر مختلفة، وتشمل هذه التحديات والمخاطر نقاط الضعف في **البرامج الثابتة** (Firmware)، والتخزين، وقوة المعالجة، وهجمات الشبكة والقواعد والمعايير، والتي تتطلب المزيد من الدراسات الإضافية. يتطلب إنترنت الأشياء المزيد من أجهزة النقل بين عنونة (IPv4) إلى (IPv6)، مما يستلزم زيادة **عرض النطاق الترددي** (Bandwidth) للشبكات، حيث يؤدي اعتماد عنونة (IPv6) وتقنية الجيل الخامس، واستخدام الجيل الجديد من الاتصالات فائق السرعة إلى ظهور مخاطر وصعوبات إضافية.

توضِّح الرسوم التوضيحية الآتية كيفية تطور هيكلية نظام بسيط إلى نظام مُعقَّد. وكما تلاحظ فإن إضافة كل طبقة من التعقيد تتسبب في ظهور نقاط ضعف جديدة لمكونات النظام.



شكل 5.30: الثغرات الأمنية في أنظمة إنترنت الأشياء

شبكات الجيل الخامس وأمان إنترنت الأشياء

5G Networks and IoT Security

تُعدّ تقنية **الجيل الخامس** (5G) تقنية واعدة بصفتها المستقبل الواعد للتطور العالمي للاتصالات المتنقلة. إن تقنية الجيل الخامس هي المكون الرئيس لاتصال المجتمع بالشبكات وبأنظمة **إنترنت الأشياء واتصال آلة إلى آلة** (IoT / M2M)، مما يُتيح الوصول السريع إلى المعلومات والخدمات، وتهدف تقنية الجيل الخامس إلى تحقيق الاتصالات المحمولة بين البشر في كل مكان باستخدام أي جهاز أو تطبيق محوسب يمكنه أن يتصل بالإنترنت، مثل **إنترنت الأشياء** (IoT)، و**ويب الأشياء** (Web of Things – WoT)، ونظرًا لتطور شبكات الجيل الخامس، أصبح من الطبيعي ظهور مشكلات تتعلق بتأثير الجيل الخامس على الأمان في اتصالات أجهزة إنترنت الأشياء، وأصبحت هناك حاجة إلى برمجيات وسيطة لإنترنت الأشياء، ومعايير أمنية لتنفيذ طرائق جديدة لربط مختلف الشبكات والأجهزة المُعرّفة. وهكذا ومع وجود بُنية تحتية للشبكة أفضل وأسرع، سيكون هناك تفاعل أكبر بين الأشياء، لا سيما مع توزيع المعالجة عبر الخدمات السحابية، مما سيؤدي إلى إحداث تأثير كبير فيما يتعلق بأمن البيانات وتمكين تطوير تطبيقات جديدة تعمل على تحسين حياة البشر.

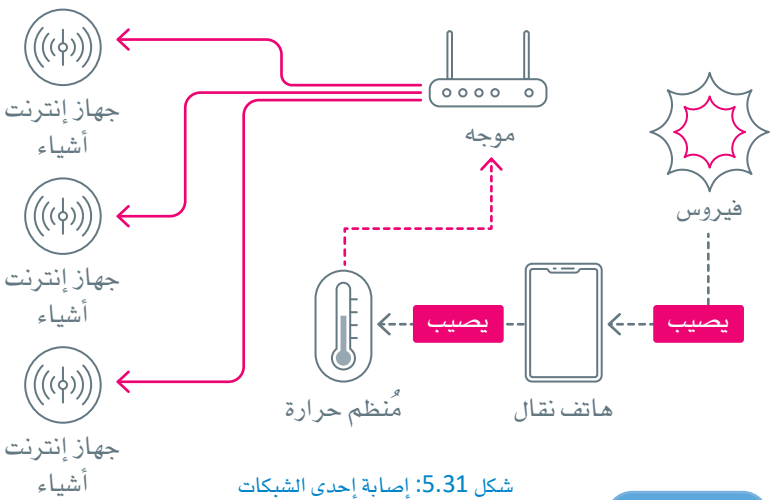
يوضِّح الجدول الآتي المخاوف الأمنية الرئيسة لشبكات الجيل الخامس الخاصة بأنظمة إنترنت الأشياء.

جدول 5.7: المخاوف الأمنية لشبكات الجيل الخامس لأنظمة إنترنت الأشياء

المخاوف	الوصف
 أمن البيانات الضخمة	تنشئ أنظمة إنترنت الأشياء كميات كبيرة من البيانات غير المتجانسة وبصورة مستمرة. تتوسع متطلبات حركة البيانات للاتصالات المتنقلة في أنظمة إنترنت الأشياء بشكل كبير، ولذلك يُعدُّ ابتكار طريقة فعّالة لإدارة هذا الكم الكبير من البيانات التي أنشأتها أنظمة إنترنت الأشياء أمرًا ضروريًا، وتوفر تقنيات شبكات الجيل الخامس إمكانية نقل البيانات بتكلفة أقل بكثير لكل بت من البيانات مقارنةً بالشبكات السابقة، ولكنها تخلق الحاجة إلى معايير بروتوكولات آمنة لإدارة وتنظيم هذا الكم الكبير من البيانات بشكل صحيح، وذلك من خلال وضع حلول أمنية تشمل إنترنت أشياء قائم على الجيل الخامس.
 حماية الأجهزة والتطبيقات	تمثل حماية العديد من الأجهزة والتطبيقات صعوبة إضافية. تتميز أنظمة إنترنت الأشياء القائمة على الجيل الخامس بقدرتها على دعم عدد أكبر بكثير من الأجهزة والتطبيقات مما هو موجود الآن، حيث ستؤدي الاتصالات بين ملايين الأجهزة والتطبيقات الإضافية إلى بروز مخاوف أمنية جديدة، فمثلًا قد يتسبب حدوث هجوم إلكتروني بسيط في منع السكان من مغادرة منازلهم وسياراتهم وغيرها من الأشياء المرتبطة بالشبكة.
 حماية قنوات الاتصال	الحفاظ على خصوصية جهاز المستخدِم والمعلومات الشخصية والبيانات الحساسة.

الخصوصية Privacy

تُشكِّل مسألة الأمان عبر الإنترنت مصدر قلقٍ وتحديًا كبيرًا في بيئة إنترنت الأشياء. ومن ناحيةٍ أخرى فإن الحفاظ على خصوصية بيانات المستخدِمين يُشكِّل تحديًا كبيرًا أيضًا يتطلب اهتمامًا إضافيًا. قد تتعرض خصوصية المستخدِمين لإنترنت الأشياء للخطر إذا تم تسريب البيانات الشخصية إلى جهات غير مُصرح لها، ونظرًا لتنوع الأجهزة المتصلة بإنترنت الأشياء ونقاط الضعف الكامنة في الأجهزة والبرامج، فإن حماية خصوصية المستخدِم النهائي تُمثل العديد من التحديات الأمنية. يسمح الكم الهائل من البيانات الشخصية المُجمَّعة من أنظمة البيانات الضخمة للمؤسسات بدمج مجموعات البيانات المختلفة، مما يزيد من القدرة على تحديد الأفراد، وتزداد القدرة على استخراج مجموعات البيانات وتحليل حجمها وتغيرها بشكلٍ يومي. وللتغلب على هذا، فإن من الحِكمة التأكد من إخفاء البيانات التي يُمكن أن تدل على شخصية صاحبها وجعل تلك **البيانات مجهولة المصدر** (Anonymized Data)، كما يجب على المؤسسات التي تستخدِم البيانات مجهولة المصدر إجراء تقييم شامل للمخاطر وتطبيق تقنيات أمنية فعّالة، ويشمل ذلك مجموعة متنوعة من الضمانات التقنية مثل: إخفاء البيانات، والتسمية المستعارة، فضلًا عن الضمانات القانونية والتنظيمية.



شكل 5.31: إصابة إحدى الشبكات

مثال

يمكن للقراصنة اختراق شبكة لإنترنت الأشياء، وجمع البيانات الخاصة عن طريق استغلال آلية التوصيل والتشغيل العالمي (UPnP)، والتي تتطلب تكوينًا بسيطًا دون الحاجة إلى وجود مصادقة للاتصال، ويستغل المتسللون هذه الميزة لإصابة جهاز، ومن ثَم إصابة شبكة إنترنت الأشياء. على سبيل المثال، يمكن للهاتف المحمول المصاب بفيروس أن يتصل بمنظم الحرارة في المنزل الذكي عبر شبكة الواي فاي. يُوصل منظم الحرارة هذا من خلال (UPnP) بالموجه الخاص بالمنزل الذكي، وبالتالي تصاب شبكة إنترنت الأشياء المنزلية بأكملها بهذا الفيروس، مما يؤدي إلى خرقٍ كامل لبيانات المعلومات الخاصة.

إخفاء البيانات (Data Masking):

يتم في عملية إخفاء البيانات تغيير البيانات الحساسة. وهكذا فإن البيانات تصبح عديمة الجدوى بالنسبة للمتطفلين غير المصرح لهم، ولكن لا يزال بالإمكان استخدامها من قِبل البرامج والموظفين المعتمدين للمزيد من التحليل.

الأسماء المستعارة (Pseudonymization):

تُستخدم الأسماء المستعارة لإدارة البيانات واستبدال محددات الهوية من تلك البيانات، وبالتالي تحل هذه الأسماء مكان حقول معلومات التعريف الشخصية في سجل البيانات، وذلك باستخدام قيم وأسماء مستعارة.

الخصوصية التفاضلية (Differential Privacy):

يُضاف مقدار عشوائي من الخصوصية، وهو عبارة عن مجموعة بيانات غير ذات أثر على دقة مجموعة البيانات. تُستخدم هذه التقنية لمنع تحديد أي معلومات شخصية للأفراد في مجموعة البيانات.

التوصيل والتشغيل العالمي (Universal Plug and Play - UPnP):

هي خدمة تُمكّن الأجهزة الموجودة على نفس الشبكة المحلية من البحث والاتصال ببعضها تلقائيًا باستخدام بروتوكولات الشبكات القياسية. تُعدُّ الطابعات والموجهات والأجهزة المحمولة وأجهزة التلفاز الذكية من أنواع أجهزة (UPnP).

شكل 5.32: الأسماء المستعارة وإخفاء البيانات

تُعدُّ حماية البيانات وأمنها أمرًا غير سهلٍ في بيئة إنترنت الأشياء، حيث يعتمد جوهر النظام على وجود واجهة اتصال بين الكائنات الذكية دون تدخُّل بشري. ونظرًا للمعدل المتسارع لتطور مثل هذه الأنظمة، فإن التأخير الملحوظ في أنظمة حماية البيانات وكذلك في وعي المُشرِّعين بالمخاطر العملية المتعلقة بالحماية والأمان ليس مُستغربًا. يوضِّح الجدول الآتي مخاوف الخصوصية الحالية في إنترنت الأشياء والحلول الممكنة لها.

جدول 5.8: مخاوف خصوصية إنترنت الأشياء والحلول الممكنة لها

مخاوف الخصوصية	الحلول المقترحة
جمع البيانات من مصادر مختلفة دون التحقق الدقيق من المُلاءمة أو الدقة.	استخدام الذكاء الاصطناعي للتحقق من دقة البيانات التي يتم جمعها.
تُمكِّن أنظمة البيانات الضخمة المؤسسات من دمج مجموعات بيانات متعددة، مما يعزِّز احتمال أن تحدِّد البيانات الأفراد الأحياء.	استخدام مجموعة متنوعة من الإجراءات الأمنية، مثل إخفاء البيانات وإخفاء الهوية والتسمية المستعارة والتجميع، بالإضافة إلى الضمانات القانونية والتنظيمية.
قد يسهم الغموض السائد في عمليات معالجة البيانات والتعقيدات المتعلقة بتحليلات البيانات الضخمة في انعدام الثقة.	تحسين مستوى الشفافية من خلال توفير معلومات حول سياسة الخصوصية قبل معالجة أي بيانات يتم الحصول عليها.
صعوبة تحديد ما إذا كانت الاستخدامات الفعلية للبيانات تتوافق مع الغرض الأصلي الذي تم جمعها لأجله.	قد تقوم المؤسسة بجمع البيانات الشخصية لغرض واحد ثم تحليلها لاحقًا لغرض مختلف تمامًا. في مثل هذه الحالة، يجب إبلاغ المستخدِمين بالتغيير وعند الضرورة يجب الحصول على الموافقة.
إن أي انتهاكات أو تهديد لخصوصية المستخدِمين سيشكل ضررًا لمصداقية المُنشئين، وتتسبب في فقدان المستخدِمين للثقة في المؤسسة والنظام ككل.	تُستخدم الأساليب التقنية مثل بروتوكولات التشفيروتقنية **سلسلة الكُتل** (Blockchain)، ويمكن أيضًا الاستعانة بأنظمة الأمان المادية كأنظمة التحكم في الوصول والمراقبة بالفيديو والسجلات الأمنية.
مراعاة حماية الخصوصية عند تصميم الأنظمة.	إن إجراء تقييم مخاطر الخصوصية يعطي تحذيرات مبكرة لاكتشاف مشكلات الخصوصية.
عدم وجود سياسات وأُطر تنظيمية وطنية وإقليمية وعالمية ذات صلة بإنترنت الأشياء، والتي إن وجدت قد تتعارض مع التطور التقني أيضًا.	من الضروري اشتراك الدول والمنظمات الدولية والشركاء الصناعيين وخبراء الأمن وإنترنت الأشياء من الصناعة والأوساط الأكاديمية في تطوير حلول لحماية البيانات الشخصية الناتجة عن إنترنت الأشياء.

تمرينات

1

حدِّد الجملة الصحيحة والجملة الخاطئة فيما يلي:	صحيحة	خاطئة
1. النظام الإلكتروني المادي هو نظام يراقب آلية محددة فقط.	○	○
2. يشمل مبدأ حماية إنترنت الأشياء القيام بالحماية المادية لأجهزة إنترنت الأشياء.	○	○
3. يتم تطبيق قوانين الأمان الإلكتروني بنفس الطريقة في كل الدول.	○	○
4. تُعدُّ تقنيات عنونة (IPv6) والجيل الخامس آمنة تمامًا.	○	○
5. يمكن إنشاء تقنيات آلة إلى آلة (M2M) دون أي تدخل بشري.	○	○
6. تُشكِّل الكائنات الذكية (أجهزة إنترنت الأشياء) المُخترقة خطرًا على مستخدميها.	○	○
7. تُعدُّ أنظمة البرمجيات الوسيطة للاتصال بين شبكات الجيل الخامس عرضةً للهجمات الإلكترونية.	○	○
8. تُشفّر البيانات الشخصية التي تُنشأ بواسطة أي كائن ذكي بشكلٍ تلقائي.	○	○
9. تقدم تقنيات إخفاء الهوية بيانات مزيفة لحماية البيانات الحقيقية.	○	○
10. يمكن أن تساعد تقنيات سلسلة الكُتل (Blockchain) في حماية البيانات في أنظمة إنترنت الأشياء الموزعة.	○	○

2 **ما المسألة الأكثر إلحاحًا بشأن التطور والانتشار السريع لأنظمة إنترنت الأشياء؟**

3 صنِّف المبادئ الأساسية لأمن إنترنت الأشياء.

4 صِف التحدي الرئيس للأمان في إنترنت الأشياء وطبيعة الجرائم الإلكترونية على الإنترنت، وكيف يُمكن التغلب على مثل هذه التحديات؟

5 ميِّز بين الأنواع المختلفة للهجمات المحتملة على كل طبقة من هيكلية إنترنت الأشياء البسيطة.

6 ما التحدي الأمني التقني الأكثر أهمية الذي أسهمت شبكات الجيل الخامس في أنظمة إنترنت الأشياء في ظهوره؟ قدِّم أفكارك أدناه.

7 كيف أسهمت تقنيات البيانات الضخمة في ظهور تحديات جديدة للخصوصية؟

8 صنِّف مخاوف الخصوصية الموجودة في أنظمة إنترنت الأشياء في الوقت الحالي.

المشروع

تُعدُّ الرعاية الصحية الذكية من أهم القطاعات التي تعمل على تحسين تقنيات إنترنت الأشياء، حيث ترتبط مجموعة متنوعة من الأجهزة والأنظمة ببعضها وتتبادل كميات كبيرة من البيانات، وتُعدُّ البيانات الطبية والحيوية للمرضى من أكثر البيانات خصوصية، والتي يجب على الشركات والحكومات حمايتها بشكلٍ جيد.

1 يستخدِم المرضى والأطباء والمراكز الطبية والمستشفيات البيانات الطبية والحيوية داخل تلك المستشفيات والمراكز، أما في الرعاية الصحية الذكية فيمكن الوصول إلى هذه البيانات من أي مكان. دَوِّن أنواع الأجهزة والخدمات والأنظمة التي تنقل البيانات الحيوية الشخصية أو تعالجها أو تُخزنها من خلال أنظمة الرعاية الصحية الذكية.

2 لا تقتصر عملية حماية البيانات الحيوية على شركات التقنية التي تقوم بتطوير أنظمة إنترنت الأشياء، فالحكومات مسؤولة عن توفير التشريعات واللوائح لحماية المواطنين من إساءة استخدام البيانات الشخصية أو اختراقها. ابحث في الإنترنت عن أمثلة للتشريعات التي فرضتها المملكة العربية السعودية لأنظمة الرعاية الصحية الذكية، وعن تشريعات مُشابهة فرضتها دولة أخرى من اختيارك.

3 بعد تدوين ملاحظاتك المتعلقة بالمشكلات المحتملة للأمان والخصوصية في الرعاية الصحية الذكية، والمقارنة بين التشريعات في المملكة العربية السعودية ودولة أخرى، قم بعرضها من خلال إنشاء عرض تقديمي باستخدام **باوربوينت** (PowerPoint).

ماذا تعلّمت

- كيفية استخدام شبكات مُستشعرات الجسم في تطبيقات الرعاية الصحية الذكية.
- تحديد أنواع مُستشعرات الطائرات دون طيار المستخدمة في الزراعة الذكية باستخدام تطبيقات إنترنت الأشياء.
- تحديد المجالات الرئيسة لهيكلية (oneM2M).
- تمييز الطبقات المختلفة للهيكلية العالمية لأنظمة إنترنت الأشياء.
- تحديد الاختلافات بين تقنية تحديد الترددات الراديوية (RFID) وتقنية الاتصال قريب المدى (NFC).
- تحديد بروتوكولات الشبكة المستخدمة في شبكات المنطقة الشخصية اللاسلكية (WPANS).
- تصنيف الأسس الرئيسة للأمان في إنترنت الأشياء.
- التعرف على تقنيات الأمان المُستخدمة في خصوصية إنترنت الأشياء.

المصطلحات الرئيسة

شبكة المنطقة الواسعة طويلة المدى	LoRaWAN
آلة إلى آلة	Machine To Machine
الاتصال قريب المدى	NFC
هيكلية oneM2M	oneM2M Architecture
شبكة المنطقة الشخصية	Personal Area Network
أسماء مستعارة	Pseudonymization
تحديد الترددات الراديوية	RFID
التشعب	Thread
مركبة جوية دون طيار	UAV
شبكة المنطقة الشخصية اللاسلكية	Wireless Personal Area Network
زيجبي	Zigbee

البلوتوث	Bluetooth
شبكة مُستشعرات الجسم	Body Sensor Network
النظام الإلكتروني الملموس	Cyber Physical System
إخفاء البيانات	Data Masking
الحوسبة الطرفية	Edge Computing
مُخطط كهربية القلب	Electrocardiogram
مخطط كهربية الدماغ	Electroencephalogram
إنترنت أشياء الرعاية الصحية	Internet of Health Things
هيكلية أنظمة إنترنت الأشياء العالمي	IoT World Forum Architecture
IP النسخة السادسة	IPv6

6. برمجة إنترنت الأشياء باستخدام لغة C++

سيتعرف الطالب في هذه الوحدة على تطبيقات الحماية الذكية. وسيتعلم كذلك كيفية برمجة جهاز تحكم الأردوينو الدقيق (Arduino Microcontroller) باستخدام لغة برمجة C++، وكيفية الانتقال من اللبنات البرمجية إلى هذه اللغة في بيئة محاكاة دوائر تينكركاد (Circuits Tinkercad). وفي الختام سيُنشئ مشروعًا للحماية الذكية بواسطة هذا الجهاز، وسيقوم ببرمجته باستخدام لغة C++.

أهداف التعلُّم

بنهاية هذه الوحدة سيكون الطالب قادرًا على أن:

- يُحدِّد ميزات ومخاطر نظام الأمان في إنترنت الأشياء.
- يتعرّف على بعض أجهزة إنترنت الأشياء الأكثر استخدامًا في أنظمة الحماية الذكية.
- يتعرّف على أنواع البيانات الشائعة في لغة C++.
- يستخدِم المُعامِلات في لغة C++.
- يستخدِم الجُمل الشرطية في C++.
- يستخدِم التكرارات في C++.
- يُنشئ دالة في C++.
- يُحوِّل اللبنات البرمجية في بيئة تينكركاد إلى أوامر C++.
- يُبرمِج نظامًا للحماية الذكية باستخدام لوحة الأردوينو ولغة C++.

الأدوات:

- بيئة محاكاة دوائر أوتوديسك تينكركاد (Autodesk Tinkercad Circuits)

رابط الدرس الرقمي

www.ien.edu.sa

الدرس الأول

تطبيقات الحماية الذكية ولغة C++

الحماية الذكية Smart Security

يُعدّ نظام الحماية الذكي وسيلة أو عملية لحماية شيء ما باستخدام مجموعة من الأدوات والمكونات التي تعمل معًا. يُمكن لأنظمة إنترنت الأشياء التعامل مع عمليات المراقبة الداخلية والخارجية للبيوت والممتلكات، وتحديد مَنْ يمكنه الوصول إلى البوابات والأبواب من خلال استخدام الأقفال الذكية المُثبتة عليها. على سبيل المثال، يُمكن الاستعانة بأجراس الباب الذكية للتعرف على الزائرين ومخاطبتهم قبل فتح باب المنزل، كما يُمكن دمج كاميرات عالية الدقة يتم تنشيطها بواسطة الحركة في هذه الأدوات لحماية المنزل، وتُوفِّر أنظمة الحماية الذكية تحذيرًا من أي تحركات غير اعتيادية، كما يُمكنها تنشيط إنذار معين أو حتى الاتصال بالشرطة.

الميزات Benefits

توجد العديد من الميزات لتركيب أنظمة الحماية المنزلية الذكية، حيث يتيح إنترنت الأشياء مراقبة المنزل وإدارته عن بُعد عبر تطبيقات الهاتف المحمول. تستخدِم أجهزة الحماية الذكية تقنيات الذكاء الاصطناعي لاكتشاف الأخطار مبكرًا لتحذير المستخدِمين واتخاذ الإجراءات المحددة كالاتصال بالشرطة مثلًا، ويستثمر الناس في أنظمة الحماية المنزلية الذكية لجعل مساكنهم أكثر أمانًا. توفر هذه التقنيات المتطورة إمكانية الدخول إلى منزلك دون الحاجة إلى المفتاح، وتمنحك تحديثات فورية في حال حصول أي أمور غير اعتيادية.

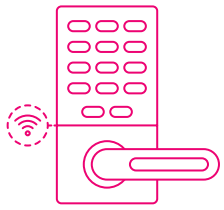
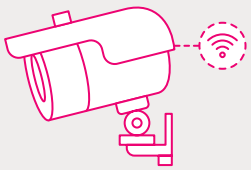
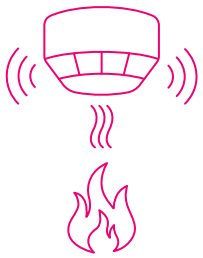
المخاطر Risks

رغم الميزات السابقة، إلا أن انعدام التشريعات الخاصة أو ضعفها باستخدام أجهزة إنترنت الأشياء وتوفير الحماية يُشكل تهديدًا خطيرًا عند تطبيق إنترنت الأشياء في المنزل الذكي. كما تبرُز أخطار الخصوصية وأمن البيانات أثناء استخدام أجهزة إنترنت الأشياء في ظل عدم وجود معايير أمان عالمية. تجمع أدوات وأجهزة إنترنت الأشياء في منزلك البيانات، ولذلك عليك حماية كل نظام يجمع معلوماتكَ الشّخصية ويحتفظ بها إذا أردت الحفاظ على خصوصيتك.

شكل 6.1: استخدام الهاتف الذكي لفتح بوابة أوتوماتيكية في مؤسسة

ستستكشف بعضًا من أكثر الأجهزة الشائعة القائمة على إنترنت الأشياء والمُستخدمة في أنظمة الحماية الذكية، مع أخذ المخاطر المختلفة بعين الاعتبار.

جدول 6.1: الأجهزة الشائعة التي تدعم إنترنت الأشياء

الأجهزة	الاستخدامات في أنظمة الحماية الذكية
 الأقفال الذكية	تعمل الأقفال الذكية على تحسين الأمان لمنزلك، وتسمح لك بالتحكم في البوابات عن بُعد، كما يمكنك وضع القيود على دخول الزوار في فترات زمنية معينة أو بناء على جدولٍ محدد، وتوفر بعض الأقفال الذكية ميزات أكثر تقدمًا كالتحكم من خلال بصمة الإصبع أو الوجه أو حتى المصادقة بمسح شبكية العين.
 الكاميرات الذكية	لا يكتمل نظام الحماية المنزلية دون استخدام الكاميرات الذكية، حيث تعمل الكاميرات كعيون رقمية لمنزلك، مما يسمح لك بمشاهدة أي نشاط داخل المنزل وخارجه بصورة فورية. توجد العديد من خيارات الكاميرا الذكية المتاحة بما فيها كاميرات **بروتوكول الإنترنت** (IP) اللاسلكية التي يمكن مراقبتها من أي مكان يتصل بالإنترنت. يمكن التقاط فيديو المراقبة للأماكن في محيط بوابات الدخول بواسطة كاميرات الباب أو البوابة.
 مُستشعرات الحرائق والدخان	تلعب أجهزة الكشف عن الحرائق والدخان دورًا مهمًا في الإنذار المبكر وإعلامك على الفور بوجود خطر ما في منزلك. غالبًا ما تُجهز المنازل الذكية بأجهزة كشف لغاز أول أكسيد الكربون، حيث توفر تنبيهات عند اكتشاف كميات كبيرة من هذا الغاز بشكل خطير. قد تقوم هذه الأجهزة بتنشيط نظام الإطفاء، أو بإخطار قسم الإطفاء للتأكد من عدم انتشار الحريق بصورة خطيرة مما قد يتسبب بخسائر أو إصابات في الممتلكات.
 مُستشعرات الحركة	تُعدُّ أجهزة الكشف عن الحركة مكونًا هامًا في نظام الحماية الذكي. تقوم هذه الأجهزة بتسجيل الاهتزازات والمعلومات وتحليلها من عدة أبعاد بواسطة هذه الأنظمة، والتي بدورها يمكنها أن تشير إلى أي حركة غير طبيعية. يمكن أن تقوم هذه المُستشعرات بتنشيط أجهزة الإنذار لإعلام المستخدِمين بالأنشطة المشبوهة سواء داخل المنزل أو في محيطه الخارجي.

لغة C++

C++ Language

ليس من السهل تحقيق أمن المعلومات، ولذلك أنت بحاجة إلى استخدام لغات برمجة قوية مثل لغة C++ لبرمجة واجهات البرامج. تُعدّ C++ لغة برمجة تجميعية عالية المستوى تتضمن العديد من ميزات البرمجة الكائنية، إضافةً إلى العديد من الإمكانيات القوية في معالجة الذاكرة، كما تتميز هذه اللغة بكفاءتها وسرعة أدائها. صُمّمت لغة C++ كتطوير للغة برمجة C.

أنواع البيانات الأساسية Basic Data Types

على عكس الكثير من لغات البرمجة الأخرى، يجب تعريف نوع المتغير في لغة C++ قبل استخدامه، لأن نوع المتغير يشير إلى نوع البيانات التي يحملها. يحتاج البرنامج في C++ إلى هذه المعلومات لمعرفة مقدار الذاكرة المطلوب تخصيصها لهذه البيانات.

جدول 6.2: أكثر أنواع البيانات شيوعًا في C++

النوع	الرمز	مثال
الأعداد الصحيحة	(int)	-4 ,5
الأعداد العشرية أو الحقيقية	(float or double)	-7.5 , 3.14
النص	(char)	'c'
البيانات المنطقية	(bool)	bool flag = true;

يمكنك تغيير نوع البيانات باستخدام مغيّر النوع، فعلى سبيل المثال (long int) يعني عدد صحيح طويل. تظهر التركيبات الممكنة لهذه المجموعات في الجدول أدناه:

double	int	char	
	✓	✓	signed
	✓	✓	unsigned
	✓		short
✓	✓		long

يُمكن للمبرمج تعريف أنواع خاصة به من البيانات بناءً على احتياجاته.

هناك بعض قواعد التسمية التي تحتاج إلى اتّباعها عند إنشائك لمتغير.

شروط تسمية المتغيرات الصحيحة:

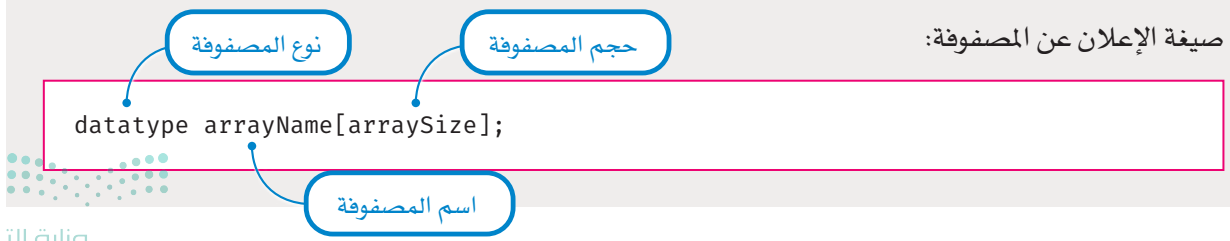
- يمكن لاسم المتغير أن يحتوي فقط على الحروف الأبجدية (A–Z ، a–z) والأرقام (0–9) والشرطة السفلية (___).
- لا يُمكن أن يبدأ اسم المتغير برقم.
- لا يُمكن أن يكون اسم المتغير هو أحد كلمات لغة البرمجة، ككلمة int مثلًا، والتي هي كلمة أساسية تُستخدم للدلالة على الأعداد الصحيحة.

يُمكن تعريف المتغيرات مع تحديد قيمتها، أو دون ذلك.

المصفوفات Arrays

يُعدُّ هيكل **المصفوفة** (Array) من أكثر هياكل البيانات شيوعًا في C++. المصفوفة هي ببساطة مُتغيِّر يمكنه الاحتفاظ بقيم بيانات متعددة من نفس النوع.

صيغة الإعلان عن المصفوفة:



لا يُمكنك تغيير نوع أو حجم المصفوفة بعد الإعلان عنها، ويمكنك الوصول إلى عناصرها باستخدام الدليل أو ما يسمى **بفهرس المصفوفة** (Array Index).

على سبيل المثال، إذا كنت تريد تخزين 10 قيم صحيحة، يُمكنك إنشاء مصفوفة تخزن فيها هذه القيم. عليك أولًا الإعلان عن نوع وحجم المصفوفة:

```
int values[10];
```

تُمثِّل "int" نوع العناصر المُخزَّنة في المصفوفة، و"values" هو اسم المصفوفة وحجمها هو 10. ولتعبئتها بالقيم يُستخدم الأمر الآتي:

```
values [10] = {0,1,2,3,4,5,6,7,8,9};
```

للوصول إلى أيٍ من عناصر المصفوفة، تحتاج إلى فهرس العنصر، فيكون الأمر:

```
int a = values[3];
```

يُعلن عن متغير باسم a، وهو عدد صحيح، وقيمته تساوي العنصر الرابع من المصفوفة "values" (تبدأ الفهرسة في C++ من 0). وعلى الرغم من إمكانية استخدام مصفوفات بأكثر من بُعد واحد، إلا أن أكثر أنواع المصفوفات شيوعًا هي المصفوفات أحادية وثنائية الأبعاد. لإنشاء مصفوفة ثنائية الأبعاد، تحتاج إلى إعلان حجم كل بُعد من أبعادها. على سبيل المثال:

```
char keys[4][2];
```

يُعلن عن مصفوفة تتكون من أربعة صفوف وعمودين، يمكنها تخزين قيم من نوع "char". ولتعبئة المصفوفة بقيمك، يجب أن تنفذ ذلك كما في المصفوفة أحادية الأبعاد:

```
keys[4][2] =
{{1,2},
{3,4},
{5,6},
{7,8}
};
```

ستحتاج هنا إلى زوج من القيم لكل صف من المصفوفة.

المُعامِلات الأساسية في C++

Basic Operators in C++

تتضمن المُعامِلات الأساسية كلًا من المُعامِلات الرياضية، ومُعامِلات الإسناد، والمُعامِلات العلائقية والمنطقية.

جدول 6.3: المُعامِلات الرياضية

المُعامِل	العملية
+	الجمع
-	الطرح
*	الضرب
/	القسمة
%	باقي القسمة

في الأعداد العشرية (float، double) يتم استخدام "/" فقط من أجل حاصل القسمة، مثلًا 5.0/2.0=2.5

في الأعداد الصحيحة (int)، يتم استخدام "/" لحساب حاصل القسمة و "٪" لحساب باقي القسمة. مثلًا 5/2= 2, 5%2=1

جدول 6.4: مُعامِلات الإسناد

المُعامِل	مثال	يكافئ
=	a = b;	a = b;
=+	a += b;	a = a + b;
=-	a -= b;	a = a - b;
=*	a *= b;	a = a * b;
=/	a /= b;	a = a / b;
=%	a %= b;	a = a % b;

جدول 6.5: المُعامِلات العلائقية

المُعامِل	الوصف	مثال
==	يساوي	5 == 3 يعطي خطأ
=!	لا يساوي	5=! 3 يعطي صواب
>	أكبر من	5 > 3 يعطي خطأ
<	أصغر من	5 < 3 يعطي صواب
>=	أكبر من أو يساوي	5=< 3 يعطي خطأ
<=	أصغر من أو يساوي	5=> 3 يعطي صواب

جدول 6.6: المُعامِلات المنطقية

المُعامِل	الوصف	مثال
&&	التعبير الأول && التعبير الثاني	AND (و) المنطقية تكون صائبة إذا كان التعبيران صائبين.
||	التعبير الأول || التعبير الثاني	OR (أو) المنطقية. تكون صائبة إذا كان أحد التعبيرين على الأقل صائبًا.
!	!التعبير	NOT (لا) المنطقية تكون صائبة فقط إذا كان التعبير خاطئًا.

التعليقات في لغة C ++
Comments in C++

تدعم جميع لغات البرمجة ميزة إضافة التعليقات داخل التعليمات البرمجية. لا تُنفّذ هذه التعليقات ضمن البرنامج، ولكنها تُستخدم لتحسين قابلية قراءة البرنامج، مما يُسهِّل على المبرمجين أو مراجعي البرامج فهمَ وظائف البرنامج. توجد طريقتان لإضافة تعليق في C++، وذلك حسب الحاجة إلى إضافة التعليق في سطر واحد أو أسطر متعددة.

استخدم // لإضافة تعليق يتكون من سطر واحد.

```
// this is a comment
int y = 10;
cout << y;
```

تعليمات غير نشطة

استخدم */ لبدء تعليق متعدد و/* لإنهائه. تُستخدم هذه الطريقة أيضًا لجعل جزء من التعليمات البرمجية غير نشط أثناء اختبار عمل البرنامج. على سبيل المثال يتم في البرنامج الآتي تخطي الجملة الشرطية if بواسطة مترجم لغة البرمجة.

```
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Enter password:");

bool correctPass = true;
char buttonPressed;
/*
int index = 4;
buttonPressed = keypad.waitForKey();
if(password[index] != buttonPressed){
    correctPass = false;
}
*/
lcd.setCursor(i, 1);
lcd.print(buttonPressed);
```

تعليمات غير نشطة

الطباعة في C++
Printing in C++

لطباعة المتغير x في C++، استخدم الأمر الآتي:

```
cout << x;
```

العبارات الشرطية في C++
Conditional Statements in C++

لتنفيذ مجموعة معينة من التعليمات البرمجية بناءً على تحقق شرط ما، يُمكنك استخدام مجموعة من الجُمل الشرطية:

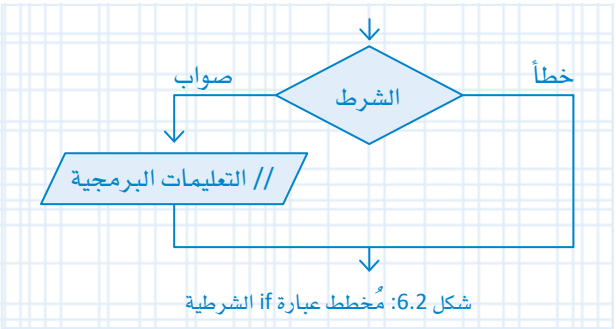
- عبارة **if**
- عبارة **if... else**
- عبارة **if... else if... else**

عبارة if الشرطية

يُستخدم هذا النوع من العبارات الشرطية إذا أردت تنفيذ مجموعة تعليمات برمجية حال تحقق شرط محدد.

صيغة عبارة if البسيطة في C++ كالآتي:

```
if (condition) {
// body of if statement
}
```



شكل 6.2: مُخطط عبارة if الشرطية

يتم أولًا فحص الشرط الموجود بين قوسين، وفي حال كانت قيمته صائبة، تُنفّذ التعليمات البرمجية الموجودة داخل الأقواس { }، أما إذا كانت خاطئة، فإنه يتم تخطي تلك التعليمات البرمجية. تعمل عبارة if كالآتي:

إذا كان الشرط صائبًا:

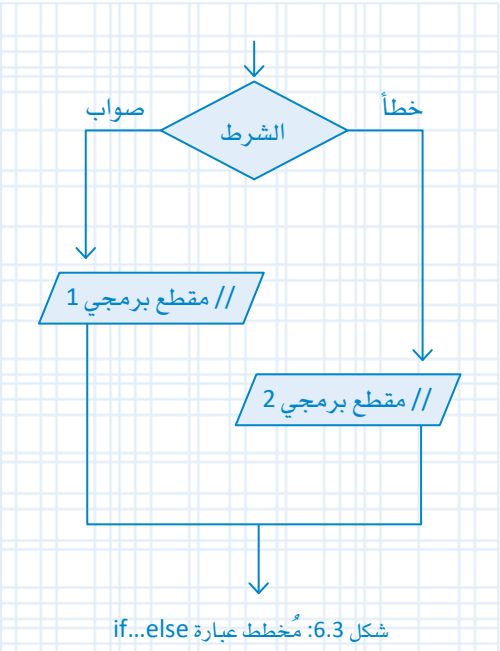
```
int number = 5;
if (number > 0) {
    // code
}
// code after if
```

إذا كان الشرط خطأ:

```
int number = 5;
if (number < 0) {
    // code
}
// code after if
```

عبارة if...else الشرطية

في هذا النوع من العبارات الشرطية، تُنفّذ مجموعة التعليمات البرمجية داخل {} if ويتم تخطي التعليمات البرمجية الموجودة داخل {} else، أو يتم تخطي التعليمات البرمجية داخل {} if وتُنفّذ التعليمات البرمجية الموجودة داخل {} else.



شكل 6.3: مُخطط عبارة if...else

تركيب عبارة if...else:

```
if (condition) {
// block of code 1 if condition is true
}
else {
// block of code 2 if condition is false
}
```

يتم أولًا تقييم الشرط الموجود بين قوسين وإذا كانت قيمته صائبة، فستُنفّذ التعليمات البرمجية الموجودة داخل {} if، وإذا كان الشرط خطأ، فسيتم تنفيذ التعليمات البرمجية الموجودة داخل {} else.

كيف تعمل عبارة if...else:

إذا كان الشرط صائبًا:

```
int number = 5;
if (number > 0) {          يتم تنفيذه
   // code
}
else {                     يتم تجاوزه
   // code
}

// code after if...else
```

إذا كان الشرط خطأ:

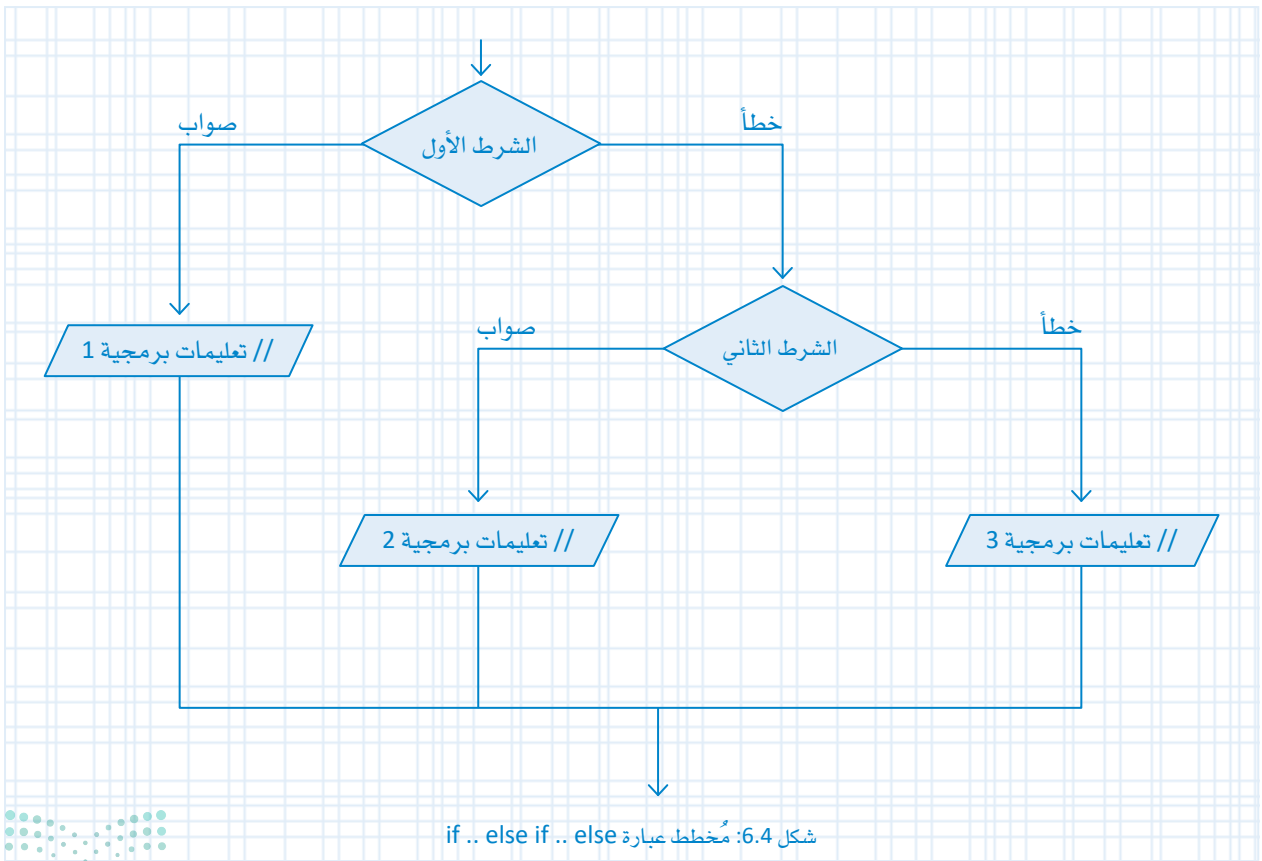
```
int number = 5;
if (number < 0) {          يتم تجاوزه
   // code
}
else {                     يتم تنفيذه
   // code
}

// code after if...else
```

عبارة if..else if..else

يُستخدم النوع الأخير من العبارات الشرطية if... else if... else عندما تحتاج إلى التحقق من أكثر من شرط واحد، أو عندما تحتاج إلى تنفيذ 3 مجموعات أو أكثر من التعليمات البرمجية وفقًا لبعض الشروط.

تركيب عبارة if .. else if .. else هو:

```
if (condition1) {
   // code block 1
}
else if (condition2) {
   // code block 2
}
else {
   // code block 3
}
```



شكل 6.4: مُخطط عبارة if .. else if .. else

كيف تعمل عبارة if .. else if .. else

إذا كان الشرط الأول صائبًا، ستُنفّذ مجموعة التعليمات البرمجية الأولى ويتم تخطي باقي التعليمات البرمجية.

إذا كان الشرط الأول صائبًا:

```
int number = 2;
if (number > 0) {                    يتم تنفيذه
    // code
}
else if (number == 0) {              يتم تجاوزه
    // code
}
else {
    // code
}
// code after if
```

إذا كان الشرط الأول خطأ وكان الشرط الثاني صائبًا، ستُنفّذ مجموعة التعليمات البرمجية الثانية، وتخطي مجموعة التعليمات البرمجية الثالثة.

إذا كان الشرط الثاني صائبًا:

```
int number = 0;
if (number > 0) {                    يتم تجاوزه
    // code
}
else if (number == 0) {              يتم تنفيذه
    // code
}
else {                               يتم تجاوزه
    // code
}
// code after if
```

إذا لم يكن أي من الشرط الأول أو الشرط الثاني صائبًا، ستُنفّذ مجموعة التعليمات البرمجية الثالثة.

كافة الشروط خطأ:

```
int number = -1;
if (number > 0) {                    يتم تجاوزه
    // code
}
else if (number == 0) {              يتم تجاوزه
    // code
}
else {                               يتم تنفيذه
    // code
}
// code after if
```

يمكنك أيضًا تضمين عبارة if داخل مجموعة التعليمات البرمجية لعبارة if أخرى. ولا يُشترط أن تكون من نفس النوع. فمثلًا:

```
// outer if statement
if (condition1) {

    // statements

    // inner if statement
    if (condition2) {
    }
}
// code after if
```

التكرارات Loops

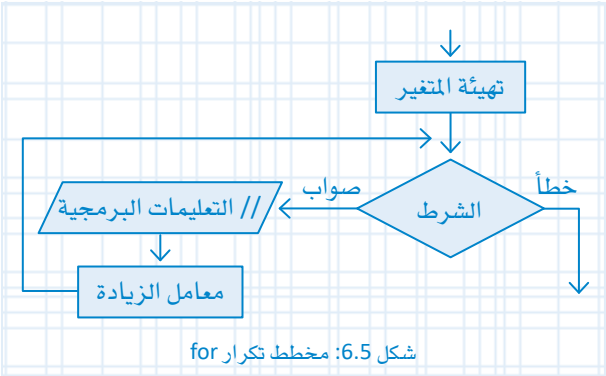
يمكنك في C++ استخدام ثلاثة أنواع من التكرارات البرمجية:

- تكرار for
- تكرار while
- تكرار do...while

تكرار for

صيغة تكرار for هي:

```
for (variable initialization;
condition; increment operation) {
// loop statements;
}
```



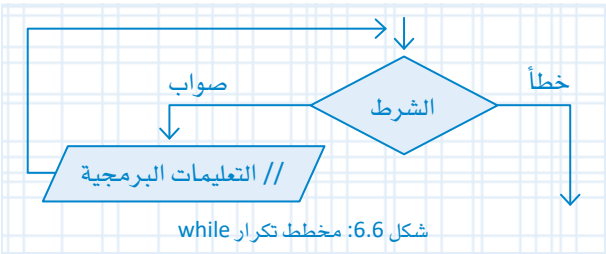
شكل 6.5: مخطط تكرار for

تُنفّذ تهيئة المتغير مرة واحدة فقط قبل بدء التكرار وتعيين قيم البداية للمتغيرات التي تشكل جزءًا من الشرط. يمكنك أيضًا الإعلان عن متغير وتهيئته في هذه الخطوة، وغالبًا يُستخدم عداد لتنفيذ التكرار عدة مرات حسب الشرط. فإذا كانت قيمة الشرط صواب، تُنفّذ جُمل التكرار ثم تتم الزيادة بتحديث قيم المتغيرات التي تمت تهيئتها. يستمر هذا حتى تتغير قيمة الشرط إلى خطأ.

تكرار while

صيغة تكرار while هي:

```
while (condition) {
// loop statements;
}
```



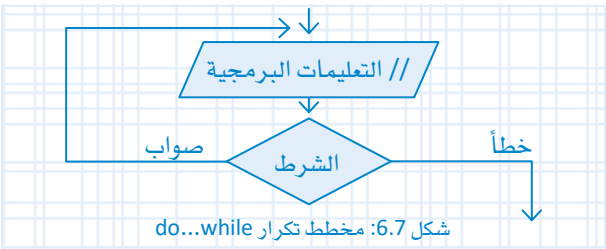
شكل 6.6: مخطط تكرار while

حيث تُنفّذ عبارات التكرار عندما يكون الشرط صائبًا، وعندما يصبح الشرط خطأ، يتوقف التكرار ويتم تخطي عبارات التكرار.

تكرار do...while

النوع الثالث للتكرارات هو do... while، وهو نوع يختلف عن تكرار while وصيغته هي:

```
do {
// statement execution;
} while (condition);
```



شكل 6.7: مخطط تكرار do...while

يختلف هذا التكرار عن تكرار while في أنه في تكرار do... while loop يُفحص الشرط بعد جُمل التكرار، وهذا يعني أن التعليمات البرمجية داخل جسم التكرار ستُنفذ مرة واحدة على الأقل. ويتوقف التكرار عند تحول الشرط إلى خطأ.

عبارات التحكم البرمجية "break" و "continue"
"break" and "continue" Statements

توجد عباراتان مفيدتان جدًا عند التعامل مع التكرارات، وهما break وcontinue واللتان تعملان مع جميع أنواع التكرارات.

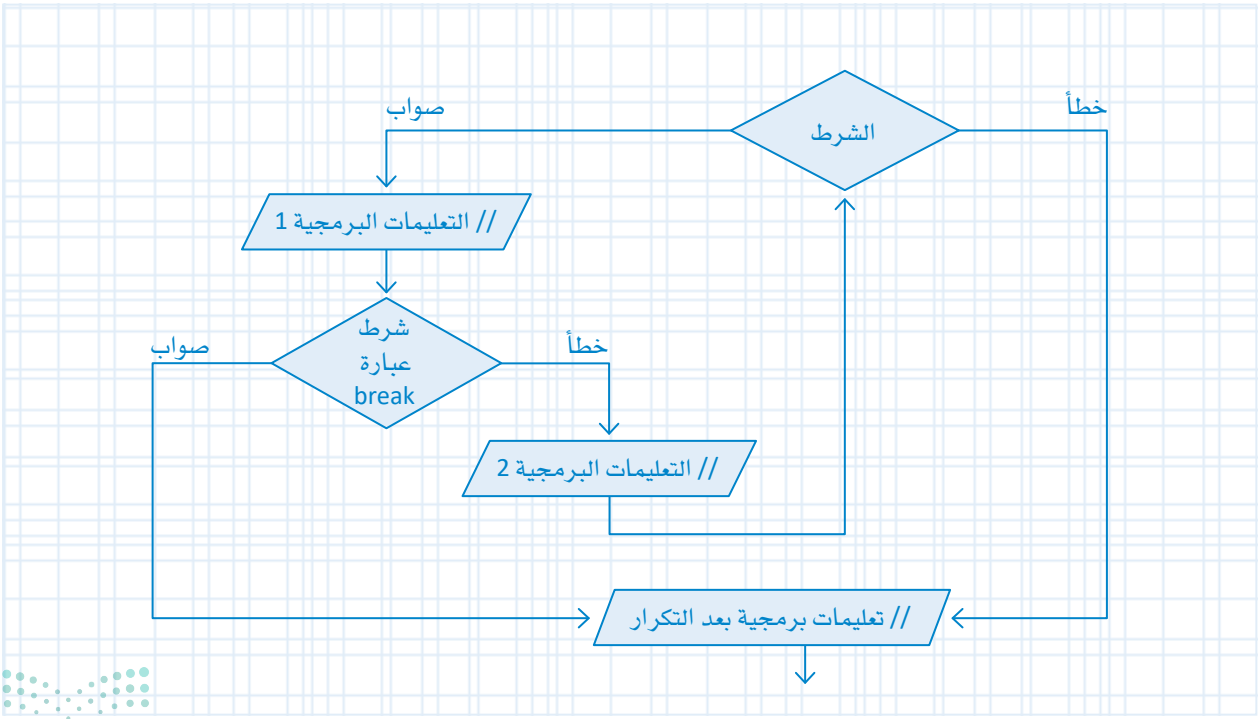
عبارة "break"

تنهي عبارة break التكرار حيث تكون موجودة.

```
while (condition) {
   // code block 1
   if (condition to break) {
      break
   }
   // code block 2
}
 // code after loop
```

```
for (init; condition; update) {
   // code block 1
   if (condition to break) {
      break
   }
   // code block 2
}
 // code after loop
```

إذا عُثر على عبارة break داخل تكرار مُتداخل، فإنها تُنهي التكرار الداخلي.



شكل 6.8: مخطط عبارة break

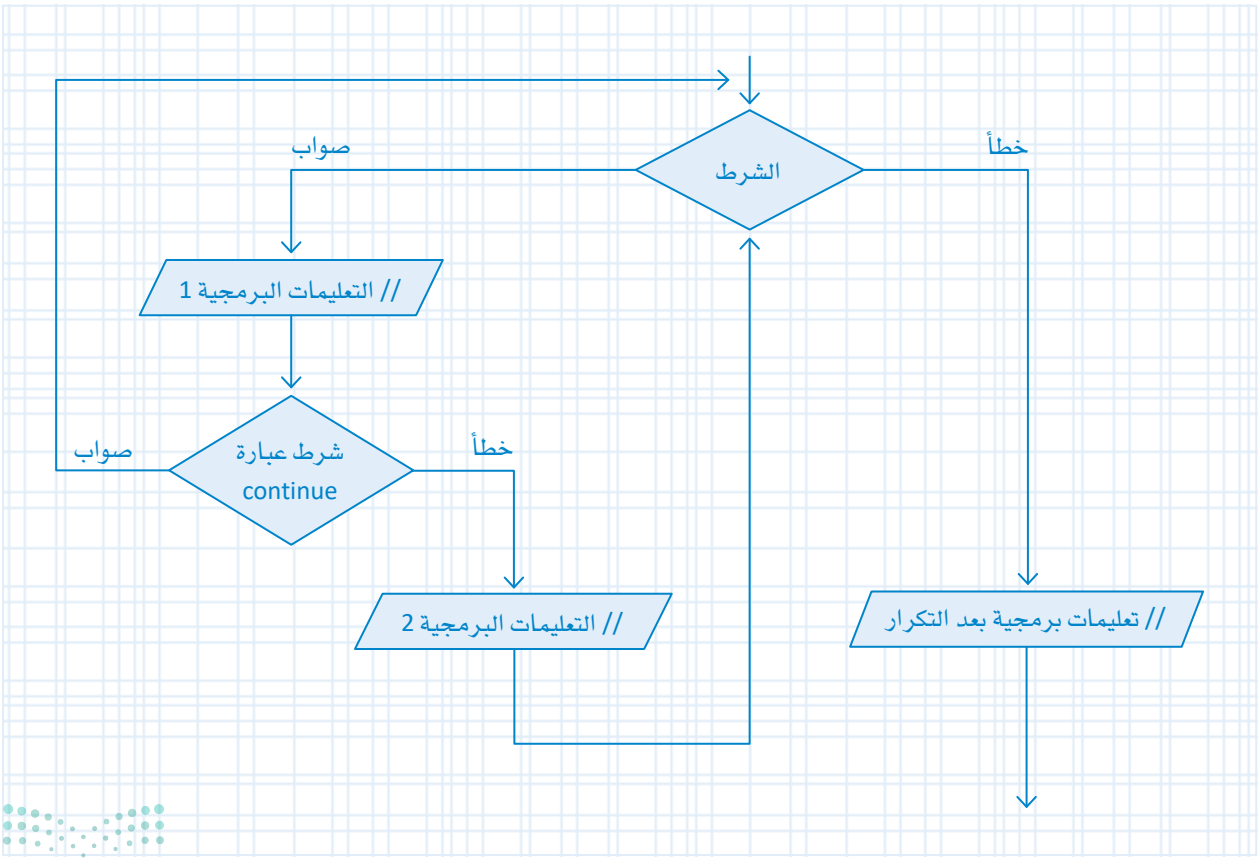
عبارة "continue"

تتخطى عبارة continue بقية التعليمات البرمجية داخل التكرار وتنتقل إلى التكرار التالي.

```
for (init; condition; update) {
    // code block 1
    if (condition to continue) {
        continue
    }
    // code block 2
}
// code after loop
```

```
while (condition) {
    // code block 1
    if (condition to continue) {
        continue
    }
    // code block 2
}
// code after loop
```

إذا وُجدت عبارة continue داخل التكرار المتداخل، سيتم تخطى التكرار الحالي في التكرار الداخلي.



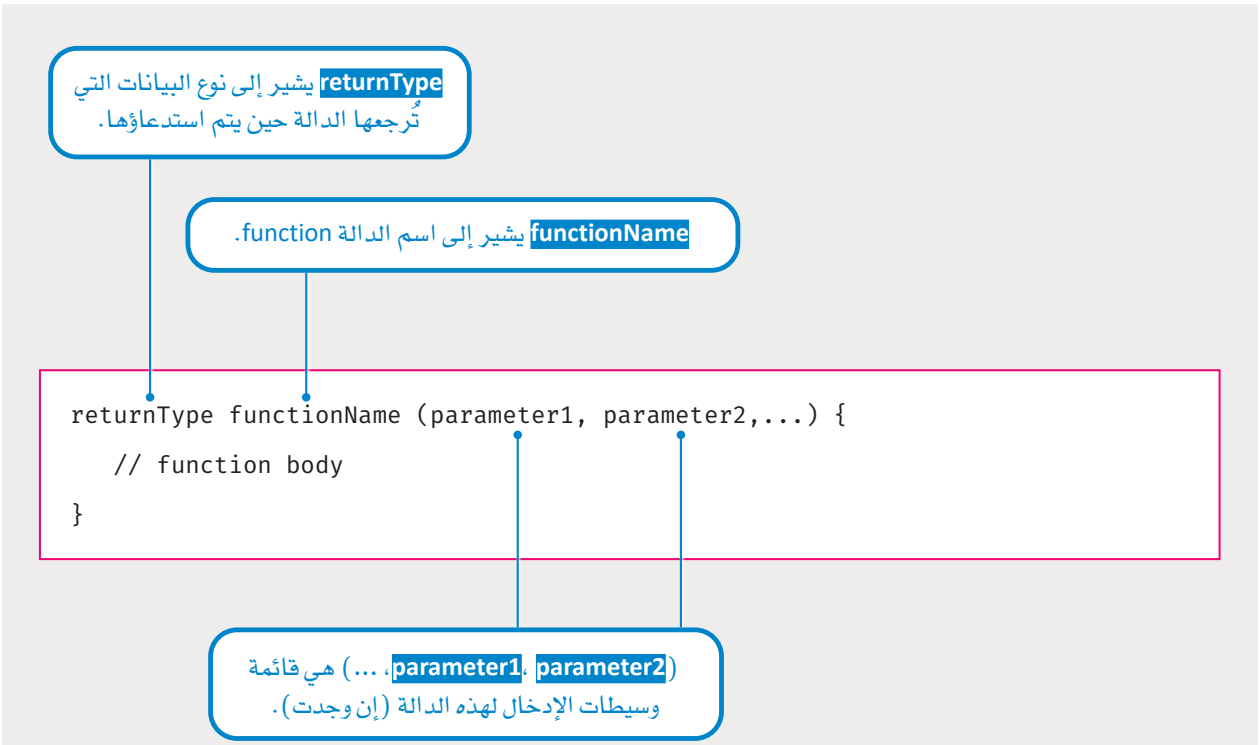
شكل 6.9: مخطط عبارة continue

الدوال في C++

Functions in C++

عند القيام بكتابة البرامج، ستلاحظ أن الكثير من المهام البرمجية قد تحتاج إلى تنفيذ عمليات معينة عدة مرات خلال تشغيل البرنامج. يمكنك بالطبع كتابة نفس سطور التعليمات البرمجية كل مرة تحتاج فيها إلى تنفيذ هذه المهام، ولكن الحل الأفضل هو تجميع هذه التعليمات البرمجية وإنشاء دالة لتؤدي هذه المهام. توجد في C++ العديد من الدوال القياسية المُضمنة والتي يُمكن للمبرمجين استخدامها. يُمكن للمبرمجين أيضًا إنشاء دوالهم الخاصة بناءً على احتياجاتهم حيث يمكنهم تسميتها بأنفسهم. يمكن لكل دالة قبول بعض المتغيرات كمُعاملات إدخال، وتنفيذ بعض التعليمات البرمجية المضمنة بين الأقواس {}، ولإنهاء الدالة توجد عبارة إرجاع (return) تُرجع قيمة.

لإنشاء دالة، تحتاج أولًا إلى الإعلان عنها:



مثال على دالة بسيطة تستقبل عددين صحيحين كوسيطين لتعيد مجموعهما:

```
// function declaration
int adding (int a, int b) {
    s = a+b;
    return s
}
```

لاستخدام هذه الدالة في برنامجك الرئيس، يمكنك استدعاؤها من خلال اسمها وتمرير عددين صحيحين لها كمُعامِلات:

```
int main () {
    int a=2;
    int b=5;
    int c;
    //calling the function and passing a, b as arguments
    c = adding(a,b);
    //cout will print the value of c
    cout << c;
    return 0;
}
```

فقط في الدالة ( )main يكون تعبير الإرجاع (return) اختياريًا، ويمكن الاستغناء عنه.

كما تلاحظ فإن الدالة main هي أيضًا دالة تقوم بإرجاع القيمة 0، وهكذا فإن نوع البيانات الذي تُرجعه الدالة هو int (عدد صحيح)، ولكنه لا يقبل أي مُعامِلات إدخال في هذه الحالة ويشار إليها بالأقواس الفارغة ( ). دالة main هي نوع خاص من الدوال في C++، حيث يوجد الجزء الرئيس من البرنامج.

يجب أن يتطابق النوع والعدد والترتيب للوسيطات التي تُمرّر إلى دالة ما مع نوع المُعامِلات الموجود في إعلان الدالة.

من الممكن ألا تُرجع الدالة أي قيمة، وفي مثل هذه الحالة يكون نوع الإرجاع "void" (فارغًا).

```
void displyNumber () {
   // code
}
```

دالّتَي ( )Setup و ( )Loop
Setup( ) and Loop( ) Functions

عند كتابة برنامج أردوينو في منصة تينكركاد، توجد دالتان يجب استدعاؤهما لتنفيذ برنامج الدائرة. تُستدعى هذه الدوال تلقائيًا عند بدء تنفيذ البرنامج، وذلك على عكس باقي الدوال التي يجب استدعاؤها يدويًا من خلال تعليماتك البرمجية.

أول دالة تُنفّذ هي ( )void setup، وتُنفّذ هذه الدالة مرة واحدة فقط في البداية، وهي مسئولة عن تكوين أجزاء الدائرة المختلفة مثل ضبط وضع أطراف الأردوينو الرقمية، وإنشاء اتصال مع الطرف التسلسلي وغيرها من الأمور.

بعد تنفيذ دالة ( )setup، تُستدعى الدالة ( )void loop بشكلٍ متكرر أثناء عمل النظام، وهذه الدالة هي التي تنفذ الوظيفة الرئيسة للدائرة.

بشكل عام، يجب أن تكتب برنامج الإعداد داخل دالة ( )void setup، وتكتب منطق البرنامج الرئيس داخل ( )void loop، وأي إعلان عن أي ثوابت أو دوال يكون خارج هاتين الدالتين.

مثال على برنامج أردوينو بلغة C++.

```
void setup() {
    int a = 10;
    int b = 20;
}

void loop() {
    for (int i = 0; i < b; i++) {
        a += i;
        cout << a;
    }
}
```

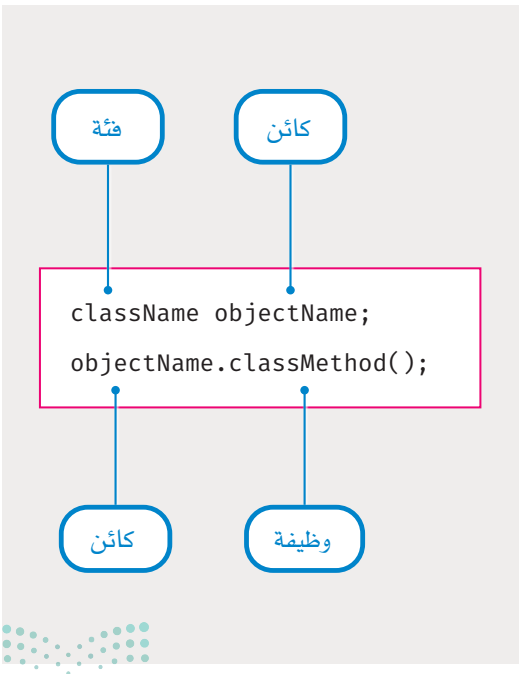
تُشغّل دالة **setup( )** مرة واحدة فقط لتكوين المتغيرات والكائنات.

تعمل دالة **loop( )** بصورة متكررة في الأردوينو.

الفئات والكائنات والوظائف

Classes, Objects, and Methods

ترتكز **البرمجة الكائنية** (Object-Oriented Programming) على إجراء جميع العمليات البرمجية على أساس "الكائنات". الكائن هو الوحدة الأساسية للبرمجة الكائنية. قد يكون لهذه الكائنات خصائص، كما يمكنها أن تنفذ بعض **الأحداث** (Actions) الأساسية. على سبيل المثال، يمكن اعتبار **محرك سيرفو** (Servo Motor) بمثابة كائن له بعض الخصائص مثل (الاسم والنوع)، ويمكنه تنفيذ بعض الإجراءات الأساسية مثل القراءة من طرف رقمي، وتدوير محركه بعدد مُعين من الدرجات وغيرها. تُسمى هذه الإجراءات التي يمكن لكل كائن تنفيذها **بالوظائف** (Methods)، وهي في لغة C++ بالأساس الدوال التي أُعلن عنها داخل جسم الكائن. من الناحية الفنية، يُعلن عن الخصائص والوظائف داخل جسم **الفئة** (Class) وليس **الكائن** (Object). لفهم الفرق بين الفئة والكائنات، يمكنك اعتبار الفئة كمفهوم، والكائنات على أنها تجسيد لهذا المفهوم. على سبيل المثال، في محاكاة الدائرة حيث سيكون هناك ثلاثة **محركات سيرفو** (Servo Motors)، فستحتاج أولًا إلى الإعلان عن فئة "Servo"، وسيكون كل من هذه المحركات الثلاثة كائنًا مؤازرًا، ويطلق عليه عادةً تسمية **العينة** (Instance) من فئة "Servo".



تمرينات

1

حدِّد الجملة الصحيحة والجملة الخاطئة فيما يلي:	صحيحة	خاطئة
1. يمكن لأجهزة إنترنت الأشياء التحكم في أبواب المنزل، وإغلاقها.	○	○
2. لا يمكنك مراقبة المنزل الذكي باستخدام الهاتف الذكي.	○	○
3. تواكب التشريعات والقوانين القضايا المتعلقة بتطبيقات الحماية الذكية لإنترنت الأشياء.	○	○
4. لا يمكن الوصول إلى أنظمة الكاميرات الذكية إلا من خلال الشبكة المنزلية.	○	○
5. يمكن لأنظمة المنزل الذكي الاتصال تلقائيًا بخدمات الطوارئ.	○	○
6. يمكن لأنظمة القفل الذكية استخدام البيانات الحيوية (البيولوجية) للتعرف على المستخدِمين.	○	○
7. تختلف لغة C++ تمامًا عن لغة C.	○	○
8. C++ هي لغة برمجة كائنية.	○	○
9. المصفوفات في لغة C++ مُحددة النوع دائمًا.	○	○
10. ليست هناك أي أهمية خاصة للدالتين ( )setup و ( )loop في برنامج الأردوينو.	○	○

2 عدِّد الفوائد التي توفرها تطبيقات الحماية الذكية في إنترنت الأشياء.

3 وضِّح المخاطر المحتملة للاستخدامات المتقدمة لإنترنت الأشياء للحماية الذكية.

4 صنِّف أكثر الأجهزة المنزلية الذكية الشائعة التي تدعم إنترنت الأشياء.

5 حدِّد الأنواع الأساسية للبيانات للبرمجة بلغة C++.

6 دوِّن القواعد الأساسية التي يجب مراعاتها عند تسمية متغيرات C++.

7 وضِّح كيفية تنفيذ تكرارات for في لغة C++.

8 صِفْ الفرق بين تكرارات while وdo... while في لغة C++.

9 وضِّح استخدام دالّتَي ()setup و ()loop في مخطط الأردوينو.

10 وضِّح الخطوات اللازمة لاختزال مُكوِّن إلكتروني يتصل بلوحة الأردوينو إلى فئة وكائن في لغة C++.

رابط الدرس الرقمي

www.ien.edu.sa

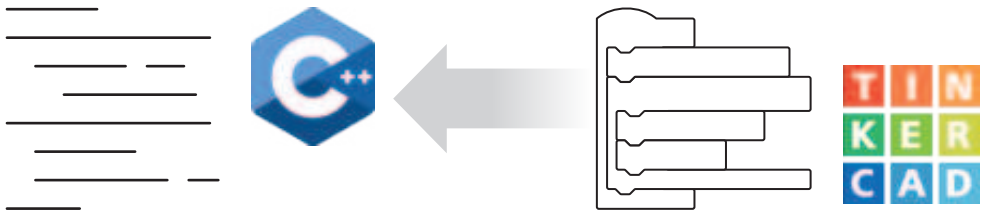
الدرس الثاني

الانتقال من اللبنات البرمجية في تينكركاد إلى لغة C++

الانتقال من برمجة اللبنات المرئية إلى البرمجة في C ++

Migrating from Visual Blocks Programming to C++ Programming

ستتعلم في هذا الدرس كيفية الانتقال من برمجة **الأردوينو** (Arduino) بلبنات **تينكركاد** (Tinkercad) البرمجية إلى برمجته باستخدام لغة البرمجة C++. تُعدُّ اللبنات البرمجية في تينكركاد مفيدة في تنفيذ النماذج الأولية والمهمات البرمجية البسيطة، إلا أن استخدام C ++ يُعدُّ ضروريًا للاستفادة الكاملة من إمكانات مُتحكم الأردوينو. ستتعلم في هذا الدرس الدوال والجُمل الأساسية لبدء برمجة متحكم الأردوينو باستخدام لغة C++.



شكل 6.10: من اللبنات البرمجية في تينكركاد إلى البرمجة في C++

يوفر تينكركاد بيئة محاكاة لبرمجة الأردوينو بالنمذجة، والتي لا تتطلب وجود أردوينو فعليًا وتوصيله بجهاز الحاسب.

الإعلان عن المتغيرات والعمليات

Variable Assignments and Operations

يتم إعلان المتغيرات وتغييرها في لبنات تينكركاد البرمجية من خلال مجموعتي أوامر **المتغيرات** (Variables) و**الحساب** (Math). يوضِّح الجدول الآتي أمثلة للأوامر المتاحة.

مجموعة لبنات الأوامر المستخدمة:

الإعلان عن متغير باسم x.

C++

```
int x = 0;
```

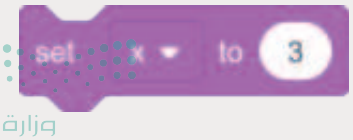
لبنة تينكركاد

تعيين قيمة للمتغير.

C++

```
x = 3;
```

لبنة تينكركاد

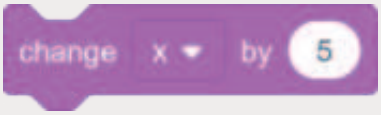


تغيير قيمة متغير بقيمة محددة.

C++

```
x += 5;
```

لبنة تينكركاد



تنفيذ عملية رياضية بين المتغيرين x وy.

C++

```
x = x - y;
```

لبنة تينكركاد

تعيين متغير ثالث z لناتج عملية رياضية بين المتغيرين x وy.

C++

```
z = x / y;
```

لبنة تينكركاد

إجراء مقارنة رياضية بين المتغيرين x وy.

C++

```
x < y
```

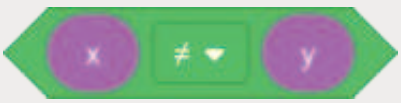
لبنة تينكركاد

إجراء مقارنة منطقية بين المتغيرين x وy.

C++

```
x != y
```

لبنة تينكركاد



إجراء عملية منطقية بين عبارتيين.

C++

```
x != y && x < y
```

لبنة تينكركاد

العبارات الشرطية والتكرارات ورسائل الإخراج
Conditional Statements, Loops and Output Messages

مجموعة لبنات الأوامر المستخدمة:

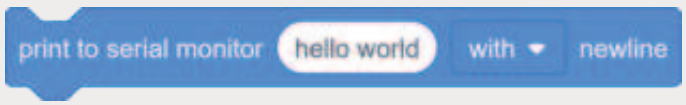
تُنشأ العبارات الشرطية والتكرارات ورسائل الإخراج في لبنات تينكركاد من خلال مجموعات أوامر **التحكم** (Control) و**الإخراج** (Output). يوضح الجدول الآتي أمثلة للأوامر المتوفرة.

طباعة رسالة على **الشاشة التسلسلية** (Serial Monitor).

C++

```
Serial.println("hello world");
```

لبنة تينكركاد



الانتظار 5 ثواني.

C++

```
delay(5000);
```

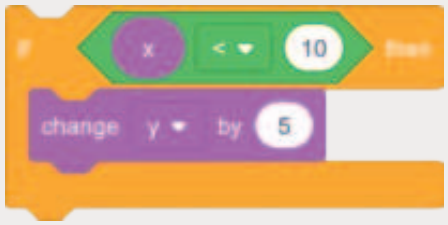
لبنة تينكركاد

تنفيذ التعليمات البرمجية داخل لبنة if إذا كان الشرط المنطقي صحيحًا.

C++

```
if (x < 10) {
   y += 5;
}
```

لبنة تينكركاد

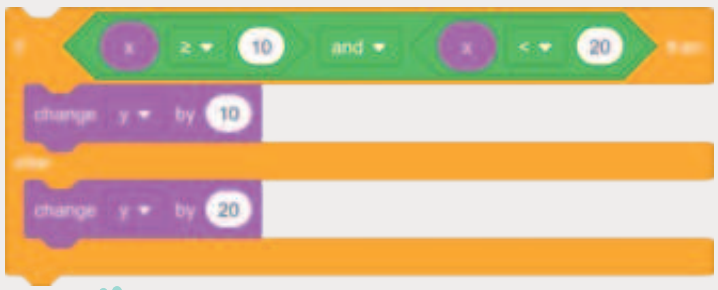


تنفيذ التعليمات البرمجية أعلى else داخل لبنة if إذا كان الشرط المنطقي صحيحًا، وإلا سيتم تنفيذ اللبنات الموجودة أسفل else.

C++

```
if (x >= 10 && x < 20) {
    y += 10;
}
else {
    y += 20;
}
```

لبنة تينكركاد

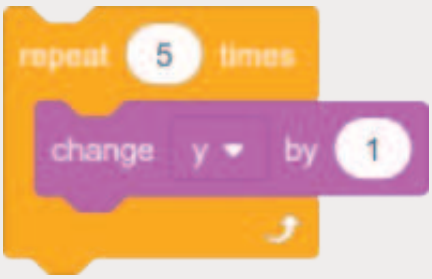


تنفيذ التعليمات البرمجية داخل لبنة for إذا كان الشرط المنطقي صحيحًا.

C++

```
for (counter = 0; counter < 5; ++counter) {
    y += 1;
  }
```

لبنة تينكركاد

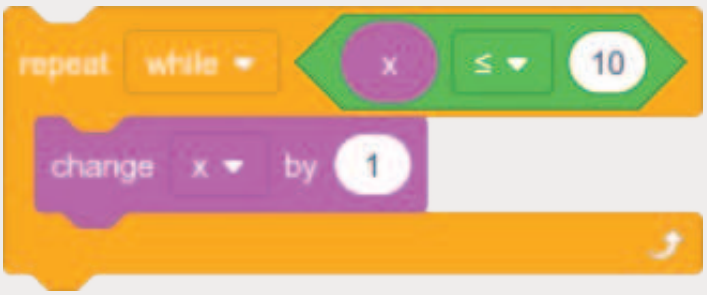


تنفيذ تكرار while في الحالة الآتية.

C++

```
while (x <= 10) {
    x += 1;
}
```

لبنة تينكركاد



مُدخلات ومُخرجات أطراف أردوينو الرقمية والتناظرية
Arduino Digital and Analog Pin I/O

يتم التفاعل مع الأطراف الرقمية والتناظرية للوحة الأردوينو في لبنات تينكركاد من خلال مجموعات أوامر **الإدخال** (Input) **والإخراج** (Output)، و**الحساب** (Math). في كل مرة يُستخدم فيها أحد أطراف الأردوينو التناظرية أو الرقمية، تتعرف لبنات تينكركاد على ما إذا كان سيتم استخدامه للإدخال/الإخراج الرقمي أو التناظري. لاستخدام طرف تحتاج إلى تحديد ذلك في دالة ()setup في الأردوينو لتوضيح ما إذا كان سيُستخدم في **الإدخال** (Input) أو **الإخراج** (Output). يتم استخدام الأطراف 3، 5، 6، 9، 10، 11 مع تعديل قيمة **عرض النبضة** (PWM). يعرض الجدول أدناه أمثلة لبعض الأوامر المتوفرة.

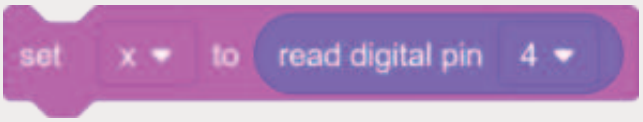
مجموعة لبنات الأوامر المستخدمة:

الحصول على قيمة الطرف الرقمي 4 وتخزينه في المتغير x.

C++

```
pinMode(4, INPUT);
x = digitalRead(4);
```

لبنة تينكركاد

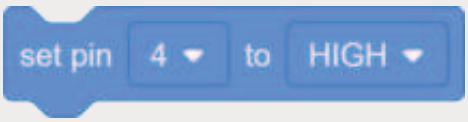


ضبط قيمة الطرف الرقمي 4 على قيمة HIGH (مرتفعة).

C++

```
pinMode(4, OUTPUT);
digitalWrite(4, HIGH);
```

لبنة تينكركاد

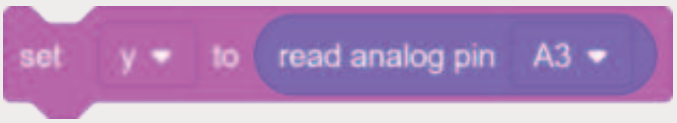


الحصول على قيمة الطرف التناظري A3 وتخزينه في المتغير y.

C++

```
pinMode(A3, INPUT);
y = analogRead(A3);
```

لبنة تينكركاد

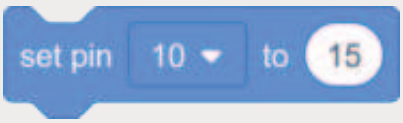


إعداد قيمة الطرف 10 من القيمة التناظرية 15 باستخدام تضمين **عرض النبضة** (PWM).

C++

```
pinMode(10, OUTPUT);
analogWrite(10, 15);
```

لبنة تينكركاد



أمثلة على الانتقال من لبنات تينكركاد البرمجية إلى لغة C++

Examples of Migration from Tinkercad Blocks to C++

ستُنشئ أمثلة بسيطة في تينكركاد للانتقال من برمجة لوحة الأردوينو باستخدام لبنات تينكركاد البرمجية إلى استخدام لغة البرمجة C++.

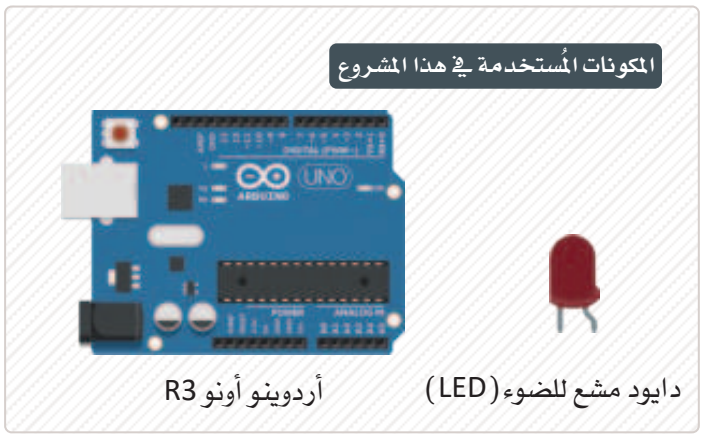
مثال الدايودات المشعة للضوء الوامضة
Blinking LEDs Example

ستُنشئ برنامجًا بسيطًا يحتوي على تكرارين يجعلان دايود مشع للضوء يومض 5 مرات و10 مرات بشدة مختلفة.

المكونات المطلوبة:

- 1 لوحة أردوينو أونو R3.
- 1 **دايود مشع للضوء** (LED).

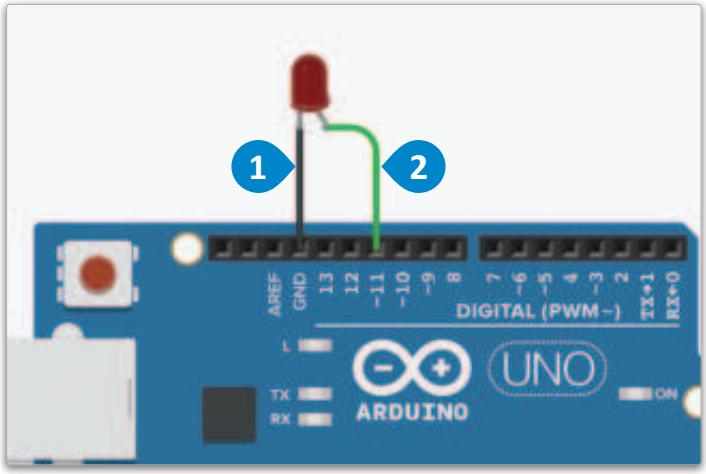
يمكن البرمجة بلغة C++ في تينكركاد وذلك بتحويل نوع التحرير من لبنات (Blocks) الى نص (Text).



شكل 6.11: مكونات المشروع

لتوصيل الدايود الضوئي:

- قُم بتوصيل Cathode (مهبط) LED (الدايود المشع للضوء) بالطرف GND (الطرف الأرضي) للوحة الأردوينو، وغيِّر لون السلك إلى black (الأسود). 1
- قُم بتوصيل Anode (مِصعد) الدايود المشع للضوء (LED) بالطرف Digital (الرقمي) 11 للوحة الأردوينو، وغيِّر لون السلك إلى green (الأخضر). 2

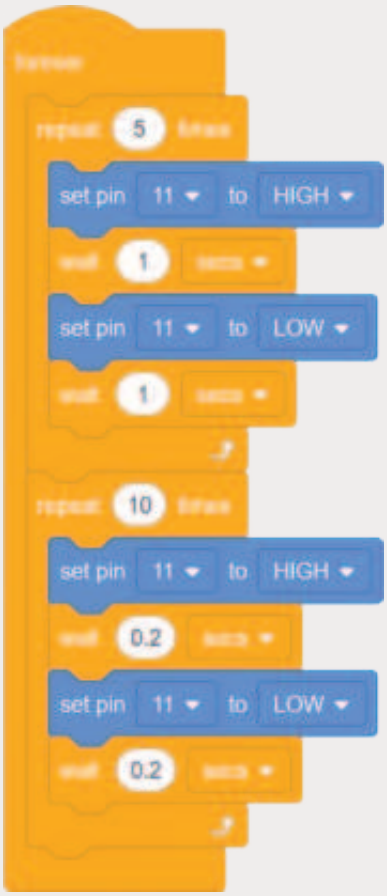


شكل 6.12: توصيل الدايود المشع للضوء

برمجة الأردوينو

عند تشغيل البرنامج، سيومض الدايود المشع للضوء 5 مرات في الثانية الواحدة، ثم سيومض الدايود المشع للضوء 10 مرات وذلك بفارق 200 ملي ثانية بين كل منها.

لبنات تينكركاد



C++

```
int counter;
int counter2;
void setup() {
  pinMode(11, OUTPUT);
}
void loop() {
  for (counter = 0; counter < 5; ++counter) {
    digitalWrite(11, HIGH);
    delay(1000); // Wait for 1000 millisecond(s)
    digitalWrite(11, LOW);
    delay(1000); // Wait for 1000 millisecond(s)
  }
  for (counter2 = 0; counter2 < 10; ++counter2) {
    digitalWrite(11, HIGH);
    delay(200); // Wait for 200 millisecond(s)
    digitalWrite(11, LOW);
    delay(200); // Wait for 200 millisecond(s)
  }
}
```

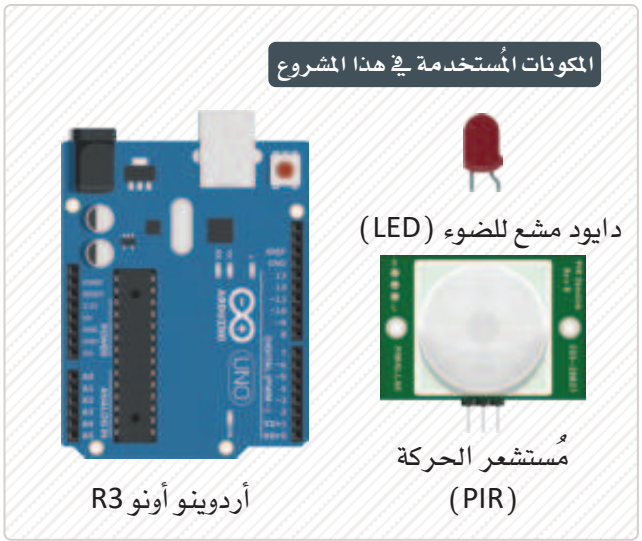
مثال إنذار مُستشعر الحركة

Passive Infrared Sensors (PIR) Alarm Example

ستقوم بتوسيع المشروع السابق بإنشاء جهاز إنذار PIR يُرسل إشارة لإضاءة ثلاثة دايودات ضوئية مشعة في تتابع سريع.

المكونات المطلوبة:

- **لوحة أردوينو أونو R3** (Arduino Uno R3).
- **دايود مُشع للضوء** (LED).
- **مُستشعرالحركة** (PIR).



شكل 6.13: المكونات المستخدمة بالمشروع

لإنشاء الدائرة:

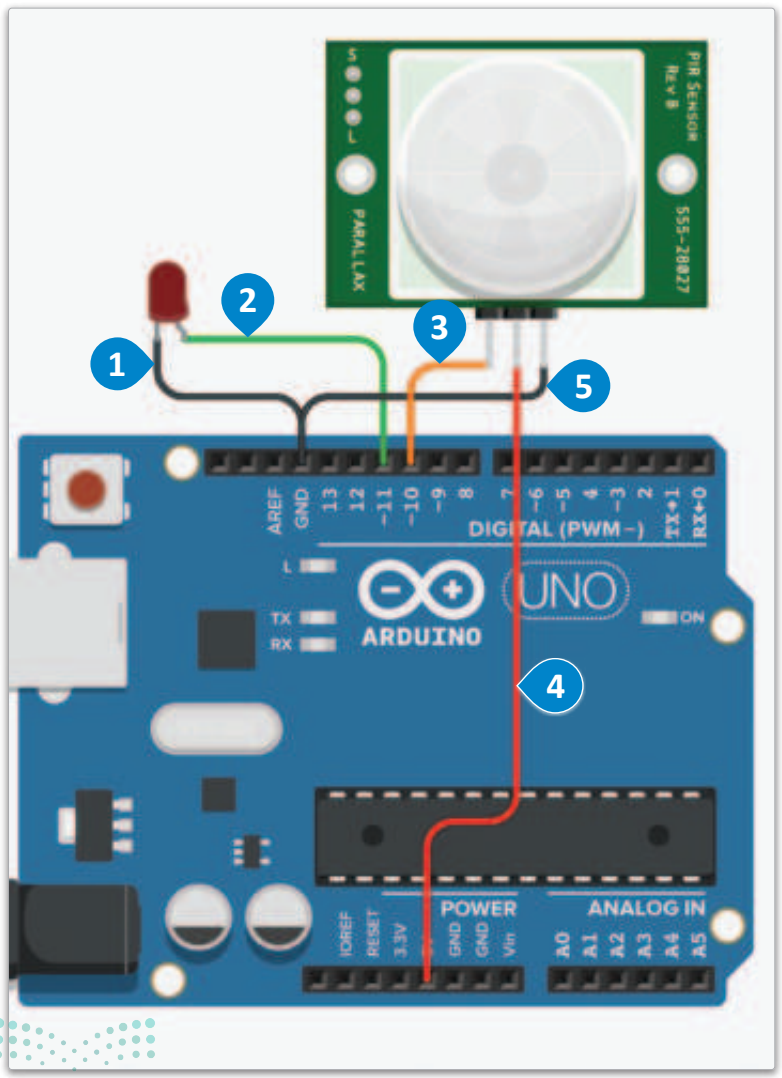
> قم بتوصيل Cathode (مهبط) لـ LED (دايود مُشع للضوء) بالطرف GND (الطرف الأرضي) للوحة الأردوينو، وغيِّر لون السلك إلى black (الأسود). 1

> قم بتوصيل Anode (مصعد) لـ LED (لدايود مُشع للضوء) بالطرف Digital (الرقمي) 11 للوحة الأردوينو، وغيِّر لون السلك إلى green (الأخضر). 2

> قُم بتوصيل طرف إشارة PIR (مُستشعر الحركة) بالطرف Digital (الرقمي) 10 للوحة الأردوينو، وغيِّر لون السلك إلى اللون orange (البرتقالي). 3

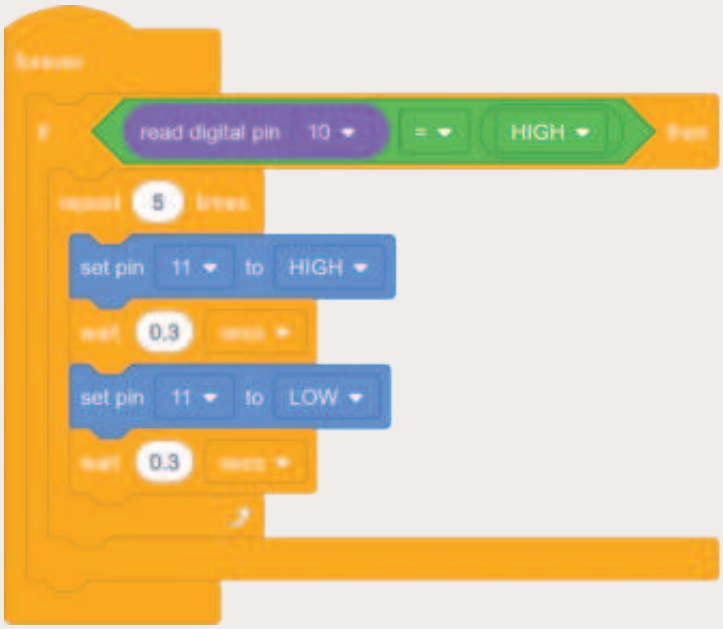
> قُم بتوصيل طرف إشارة PIR (مُستشعر الحركة) بمصدر 5V (5 فولت) من لوحة الأردوينو، وغيِّر لون السلك إلى red (الأحمر). 4

> قُم بتوصيل GND (الطرف الأرضي) لـ PIR (مُستشعر الحركة) بالطرف GND (الطرف الأرضي) للوحة الأردوينو، وغيِّر لون السلك إلى black (الأسود). 5



شكل 6.14: توصيل الدائرة

لبنات تينكركاد



برمجة الأردوينو

سيتحقق البرنامج مما إذا كان مُستشعر الحركة (PIR) قد اكتشف كائنًا في مجال رؤيته، وعند اكتشافه لشيء ما، سيُرسل إشارة للدايود المشع للضوء ليومض خمس ومضاتٍ سريعة متتابعة.

اختر وضع البرمجة نص (Text) في محرر التعليمات البرمجية لرؤية النص الناتج بلغة C++.

C++

```
int counter;
void setup() {
  pinMode(10, INPUT);
  pinMode(11, OUTPUT);
}
void loop() {
  if (digitalRead(10) == HIGH) {
    for (counter = 0; counter < 5; ++counter) {
      digitalWrite(11, HIGH);
      delay(300); // Wait for 300 millisecond(s)
      digitalWrite(11, LOW);
      delay(300); // Wait for 300 millisecond(s)
    }
  }
}
```

مثال محرك التيار المستمر
DC Motor Example

ستُنشئ دائرة بسيطة للتحكم في محرك DC وِفق درجة الحرارة المحيطة به. ستحتاج إلى المكونات الآتية:

- **لوحة أردوينو أونو R3** (Arduino Uno R3).
- **محرك تيار مستمر** (DC motor).
- **مُستشعر درجة الحرارة** (TMP36).

لإنشاء الدائرة:

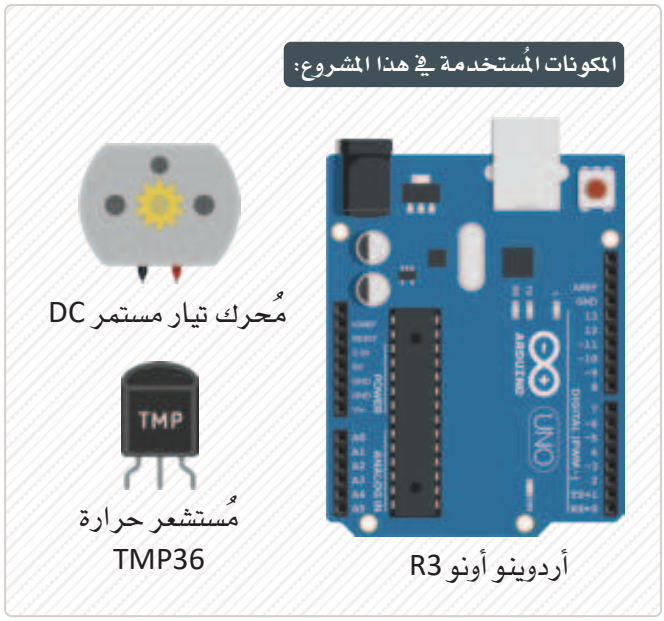
> قُم بتوصيل طرف Power (الطاقة) الخاص بمُستشعر درجة الحرارة بطرف 5V (5 فولت) من لوحة الأردوينو، وغيِّر لون السلك إلى red (الأحمر). 1

> قُم بتوصيل طرف مخرج الجهد (Vout) الخاص بمُستشعر درجة الحرارة بالطرف التناظري (A0) للوحة الأردوينو وغيِّر لون السلك إلى green (الأخضر). 2

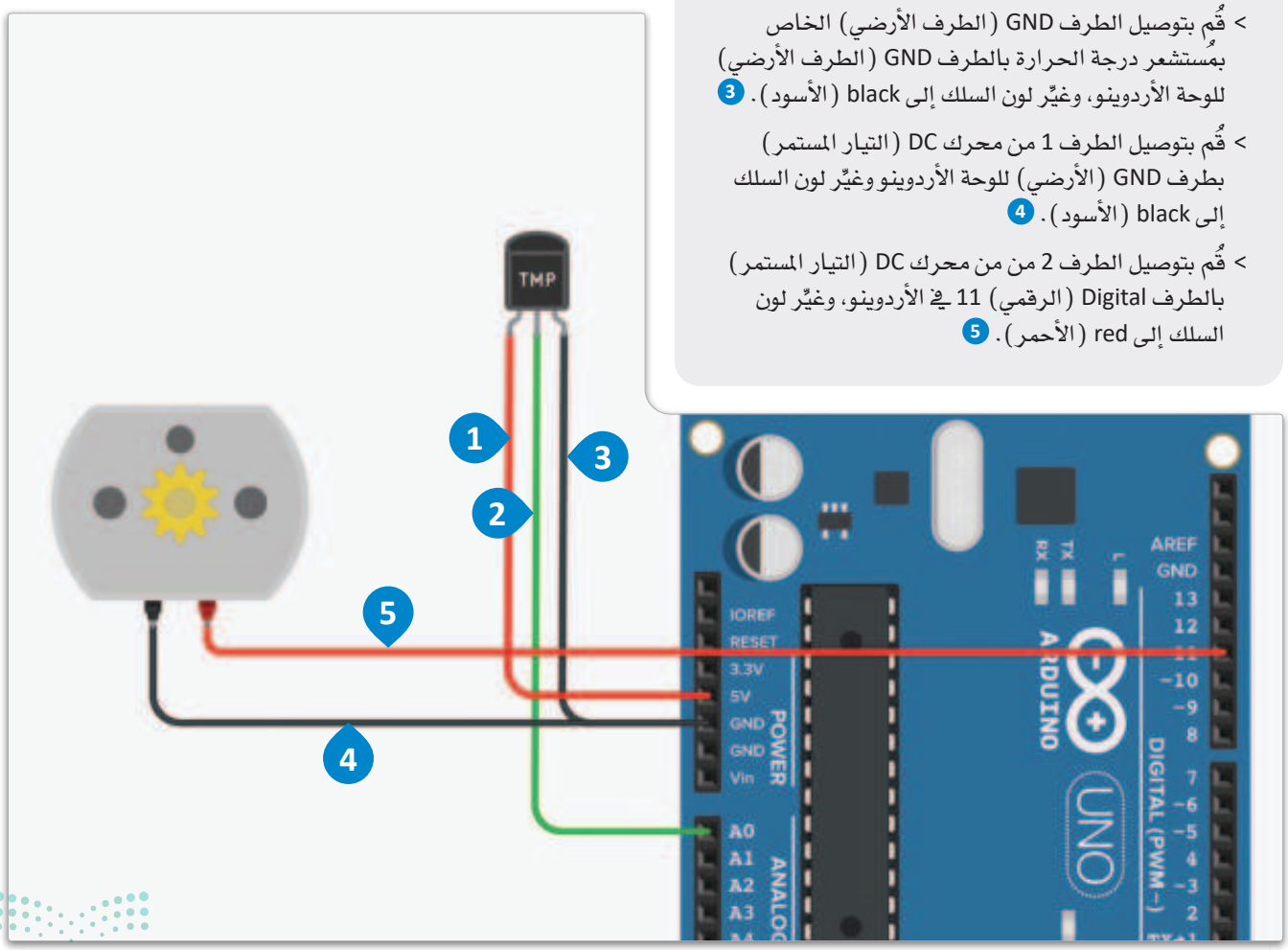
> قُم بتوصيل الطرف GND (الطرف الأرضي) الخاص بمُستشعر درجة الحرارة بالطرف GND (الطرف الأرضي) للوحة الأردوينو، وغيِّر لون السلك إلى black (الأسود). 3

> قُم بتوصيل الطرف 1 من محرك DC (التيار المستمر) بطرف GND (الأرضي) للوحة الأردوينو وغيِّر لون السلك إلى black (الأسود). 4

> قُم بتوصيل الطرف 2 من من محرك DC (التيار المستمر) بالطرف Digital (الرقمي) 11 في الأردوينو، وغيِّر لون السلك إلى red (الأحمر). 5

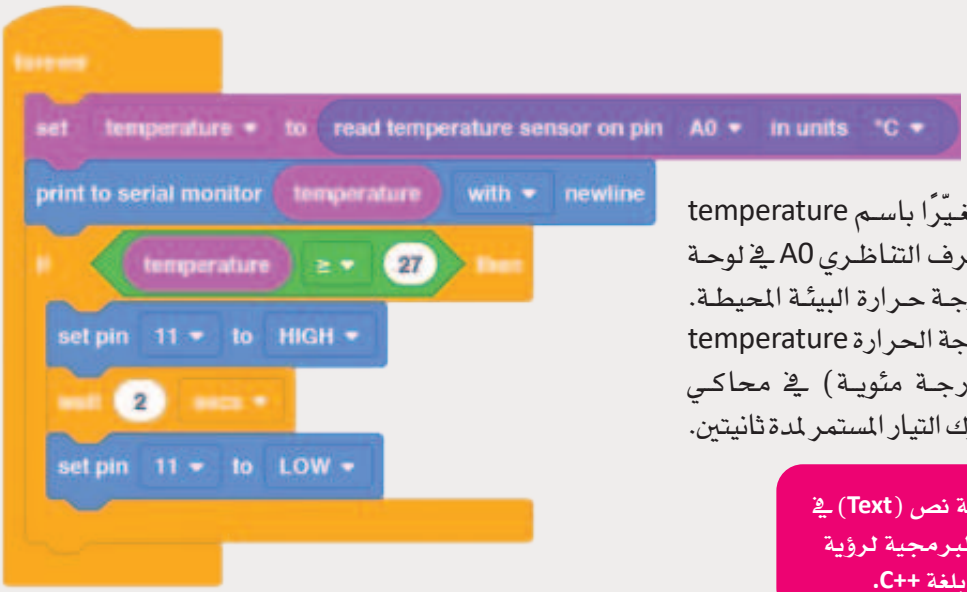


شكل 6.15: مكونات المشروع



شكل 6.16: توصيل الدائرة

لبنات تينكركاد



برمجة الأردوينو

سيُنشئ البرنامج متغيّرًا باسم temperature وسيتم توصيله بالطرف التناظري A0 في لوحة الأردوينو لتسجيل درجة حرارة البيئة المحيطة. عندما يصل متغير درجة الحرارة temperature إلى القيمة 27 (درجة مئوية) في محاكي تينكركاد، يُنشّط محرك التيار المستمر لمدة ثانيتين.

اختر وضع البرمجة نص (Text) في محرر التعليمات البرمجية لرؤية النص الناتج بلغة C++.

C++

```
int temperature = 0;
void setup() {
  pinMode(A0, INPUT);
  Serial.begin(9600);
  pinMode(11, OUTPUT);
}
void loop() {
  temperature = (-40 + 0.488155 * (analogRead(A0) - 20));
  Serial.println(temperature);
  if (temperature >= 27) {
    digitalWrite(11, HIGH);
    delay(2000); // Wait for 2000 millisecond(s)
    digitalWrite(11, LOW);
  }
}
```

يُستخدم الكائن **التسلسلي** (Serial) للطباعة على الشاشة التسلسلية. في دالة ( )setup، تقوم دالة ( )start بتهيئة الشاشة التسلسلية ليُمكن استخدامها لاحقًا. يُمكن للمستخدِم بعد ذلك طباعة القيم والرسائل على الشاشة باستخدام دالة ( )print أو دالة ( )println، مع ملاحظة أن دالة ( )println ستضيف سطرًا جديدًا في نهاية الرسالة.

تمرينات

1 اكتب دالة بلغة C++ تستقبل وسيطين عشريين من نوع بيانات float، وإشارة تناظرية، ومُضاعِفًا. مع مراعاة أن تقوم الدالة بتضخيم (مضاعفة) الإشارة ثم إرجاعها.

2 ارسم مُخطط C++ يقرأ إدخال إشارة تناظرية من طرف يُمثل قراءة درجة الحرارة بالفهرنهايت. ثم أنشئ دالة تُحوِّل هذه القيمة إلى درجات مئوية، وتُرسلها إلى طرف كمُخرج تناظري.

3 ابحث عن خطأ قاعدي (Syntax Error)، وخطأ منطقي (Logical Error) في التعليمات البرمجية الآتية:

```
void loop() {
  for (counter = 0; counter < 5; --counter) {
    digitalWrite(11, HIGH);
    // Wait for 1000 millisecond(s)
    delay("1000");
    digitalWrite(11, LOW);
    // Wait for 1000 millisecond(s)
    delay("1000");
  }
}
```

خطأ منطقي _____

خطأ قاعدي _____

```
void loop() {
  temperature = digitalRead(A0);
  Serial.println(temperature);
    if (temperature >= 270) {
    digitalWrite(11, 1);
    // Wait for 2000 millisecond(s)
    delay(2000);
    digitalWrite(11, 0);
    }
}
```

خطأ منطقي _____

خطأ قاعدي _____

4 ارسم مخطط C++ للأردوينو يستخدِم الدالة في التمرين الأول ويقرأ مُدخل إشارة تناظري. ثم أنشئ تكرار for يستخدِم الدالة في التمرين الأول لتضخيم الإشارة الأصلية 5 مرات. وفي كل مرة تُضخّم الإشارة، تُرسل إلى طرف كمُخرج تناظري.

5 تَوسَّع في المثال السابق الخاص بالدايودات المشعة للضوء الوامضة (LEDs) وقُم بإضافة دايود مشع للضوء آخر بلون مختلف يومض كل مرة يتم فيها إيقاف تشغيل الدايود المشع للضوء الأول.

6 تَوسَّع في المثال السابق الخاص بالإنذار باستخدام مُستشعر الحركة، وقُم بإضافة إنذار بواسطة مُستشعر حركة آخر ودايود مشع للضوء بلونٍ آخر. سيُوصل كل مُستشعر حركة بدايود مشع للضوء يومض بناءً على اكتشاف مُستشعر الحركة لشيء ما.

7 قُم بضبط محرك التيار المستمر في المثال الخاص باستخدام المحرك لإرسال إشارة تناظرية إلى المحرك بناءً على درجة الحرارة التي يكتشفها مُستشعر درجة الحرارة.

رابط الدرس الرقمي

www.ien.edu.sa

الدرس الثالث

برمجة جهاز التحكم الدقيق باستخدام لغة C++

إنشاء قفل باب ذكي Build a Smart Door Lock

ستستخدِم في هذا المشروع المكونات الآتية:

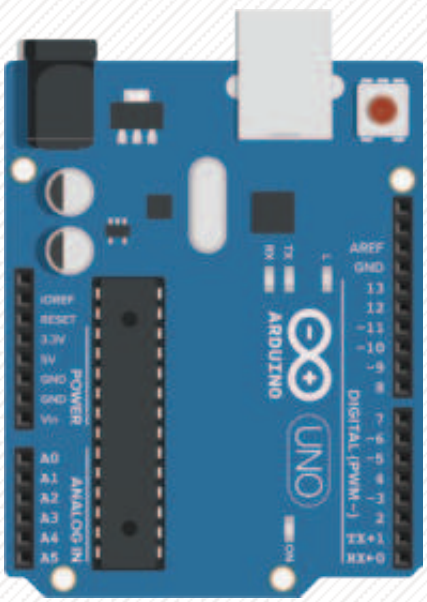
- **لوحة أردوينو أونو R3** (Arduino Uno R3).
- لوحة مفاتيح (مقاس 4x4).
- شاشة LCD مقاس 16x2 (I2C).
- **محرك سيرفو مصغر** (Micro Servo).

المكونات المُستخدمة في هذا المشروع:

شاشة LCD مقاس 16x2 (I2C)

لوحة مفاتيح 4x4

محرك سيرفو مصغر



أردوينو أونو R3

شكل 6.17: مكونات المشروع

ستبدأ بإضافة لوحة المفاتيح من فئة Input (الإدخال) من components (المكونات) وتوصيلها بالأردوينو.

لتوصيل لوحة المفاتيح:

> ابحث عن مكون Keypad 4x4 (لوحة المفاتيح) من فئة Input (الإدخال) في Components (المكونات) واسحبه وأفلته في مساحة العمل. 1

> قم بتوصيل السطر الأول من لوحة المفاتيح بالطرف الرقمي 9 الخاص بالأردوينو. 2

> قم بتوصيل السطر الثاني من لوحة المفاتيح بالطرف الرقمي 8 الخاص بالأردوينو. 3

> قم بتوصيل السطر الثالث من لوحة المفاتيح بالطرف الرقمي 7 الخاص بالأردوينو. 4

> قم بتوصيل السطر الرابع من لوحة المفاتيح بالطرف الرقمي 6 الخاص بالأردوينو. 5

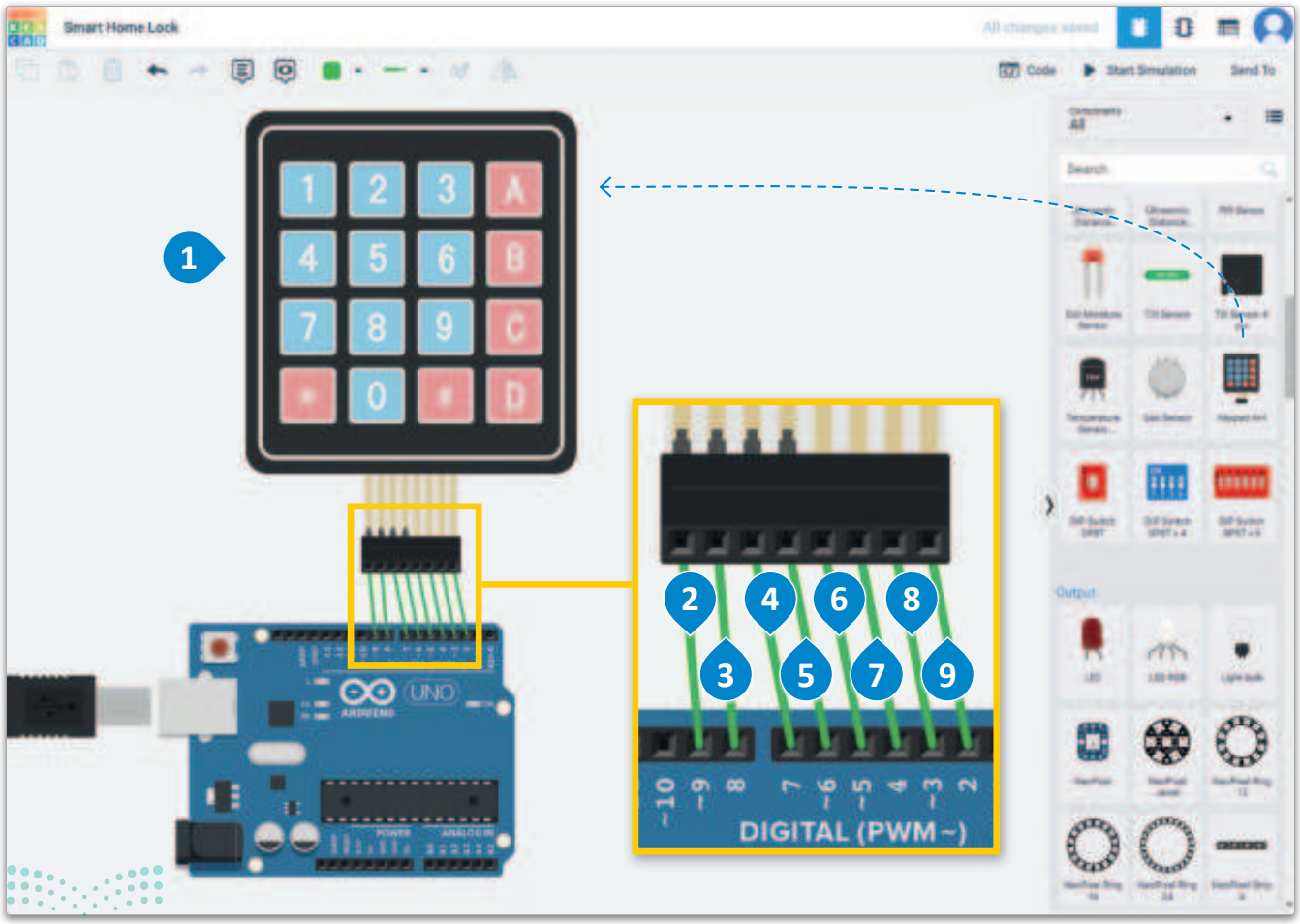
> قم بتوصيل السطر الأول من لوحة المفاتيح بالطرف الرقمي 5 الخاص بالأردوينو. 6

> قم بتوصيل السطر الثاني من لوحة المفاتيح بالطرف الرقمي 4 الخاص بالأردوينو. 7

> قم بتوصيل العمود الثالث من لوحة المفاتيح بالطرف الرقمي 3 الخاص بالأردوينو. 8

> قم بتوصيل العمود الرابع من لوحة المفاتيح بالطرف الرقمي 2 الخاص بالأردوينو. 9

> غيِّر كافة الأسلاك إلى اللون green (الأخضر).



شكل 6.18: توصيل لوحة المفاتيح

ابحث الآن عن شاشة LCD من فئة Output (الإخراج) من Components (المُكونات)، ووصلها بالأردوينو.

لتوصيل شاشة LCD:

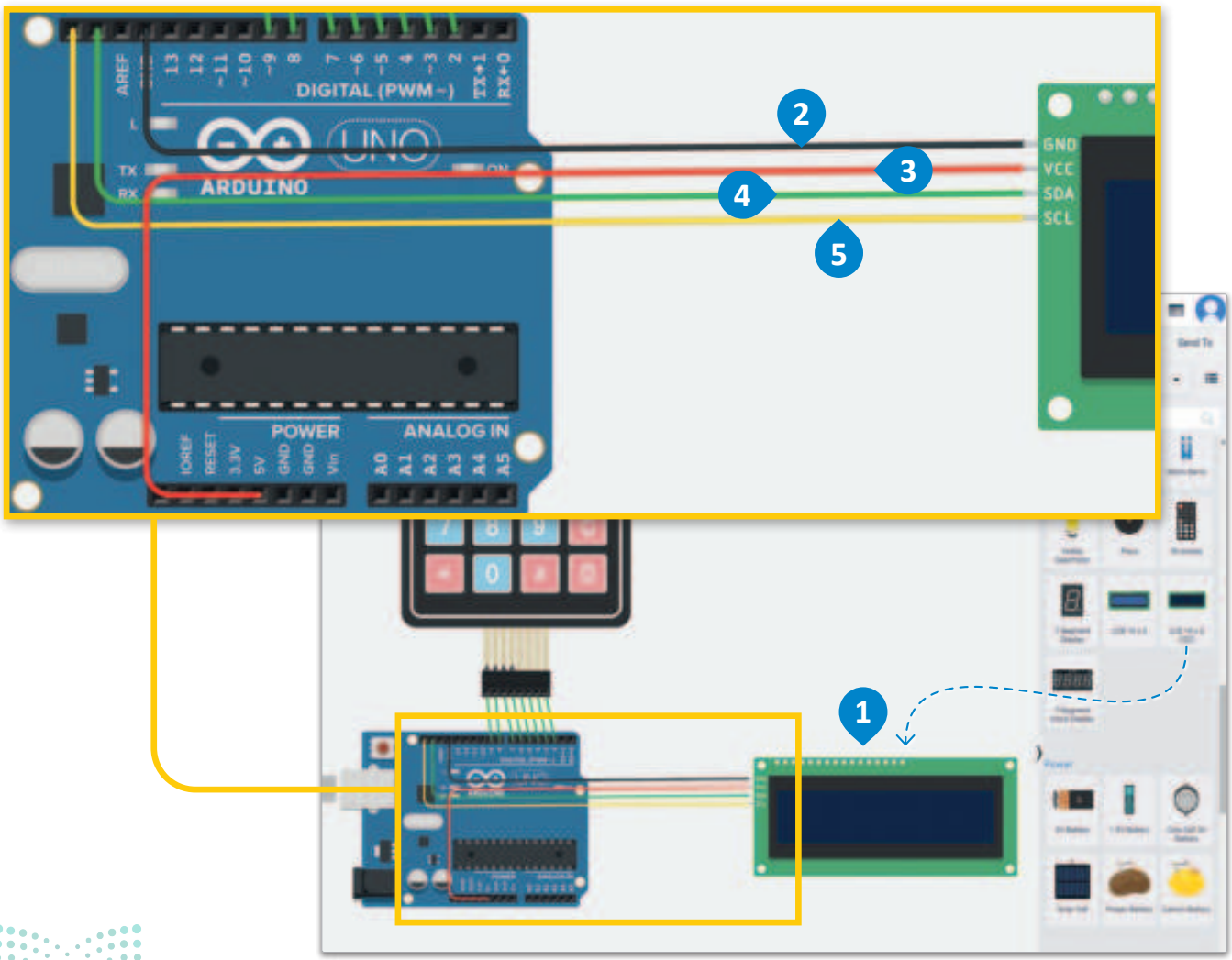
> ابحث عن المكون LCD 16x2 (I2C) من فئة Output (الإخراج) من فئة Components (المُكونات)، واسحبه وأفلته في مساحة العمل. 1

> قم بتوصيل الطرف GND (الأرضي) لشاشة LCD بطرف GND (الأرضي) الخاص بالأردوينو، وغيِّر لون السلك إلى black (الأسود). 2

> قم بتوصيل طرف Power (الطاقة) لشاشة LCD بالطرف 5V (5 فولت) بالأردوينو، وغيِّر لون السلك إلى red (الأحمر). 3

> قم بتوصيل طرف SDA لشاشة LCD بطرف SDA بالأردوينو، وغيِّر لون السلك إلى green (الأخضر). 4

> قم بتوصيل طرف SCL لشاشة LCD بطرف SCL بالأردوينو، وغيِّر لون السلك إلى yellow (الأصفر). 5



شكل 6.19: توصيل شاشة LCD

ختامًا، ستقوم بتوصيل محرك سيرفو.

ابحث عن Micro Servo (محرك سيرفو مصغر) من فئة Output (الإخراج) في Components (المُكونات)، ووصله بالأردوينو.

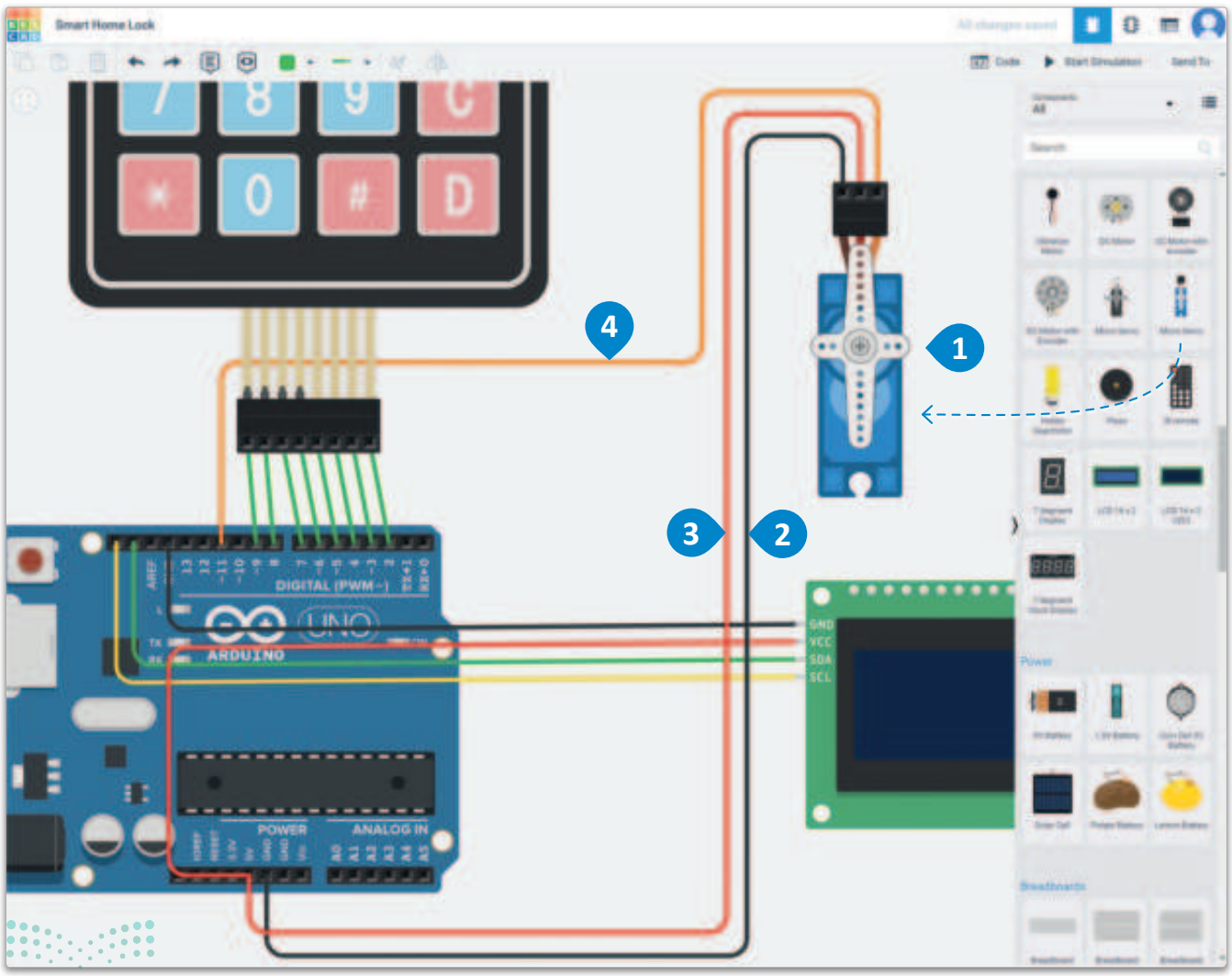
لتوصيل محرك السيرفو المصغر:

> ابحث عن Servo motor (محرك سيرفو مصغر) من فئة Output (الإخراج) في Components (المُكونات)، واسحبه وأفلته في مساحة العمل. 1

> قم بتوصيل الطرف GND (الأرضي) للمحرك بطرف GND (الأرضي) بالأردوينو، وغيِّر لون السلك إلى black (الأسود). 2

> قم بتوصيل طرف Power (الطاقة) للمحرك بالطرف 5V (5 فولت) بالأردوينو، وغيِّر لون السلك إلى red (الأحمر). 3

> قم بتوصيل طرف Signal (الإشارة) للمحرك بالطرف Digital (الرقمي) 11 بالأردوينو، وغيِّر لون السلك إلى orange (البرتقالي). 4



شكل 6.20: توصيل محرك السيرفو المصغر

تضمين المكتبات Include the Libraries

بعيدًا عن وحدة تحكم الأردوينو، ولاستخدام باقي المكونات وبرمجة منطقها بلغة C++، فأنت بحاجة أولًا إلى تضمين مكتباتها في قسم البرمجة في بيئة تينكركاد الأساسية. تُوفِّر هذه المكتبات العديد من الدوال الخاصة بكل مُكوِّن.

ستحتاج إلى كتابة الصيغة الآتية لتضمين مكتبة في C++:

```
#include <library name>
```

بالنسبة للمشروع الحالي، ستحتاج إلى تضمين المكتبات الآتية:

بالنسبة للوحة LCD:

```
#include <Adafruit_LiquidCrystal.h>
```

بالنسبة للوحة المفاتيح:

```
#include <Keypad.h>
```

بالنسبة للمحرك المؤازر:

```
#include <Servo.h>
```

إنشاء الكائنات Create the Objects

بعد تضمينك للمكتبات الضرورية، ستحتاج إلى إنشاء بعض الكائنات وتهيئة بعض الوسيطات.

ستحتاج إلى إنشاء الكائنات الآتية:

- كائن لشاشة LCD.
- كائن **للمحرك المؤازر** (Servo motor).
- كائن للوحة المفاتيح.

عند إنشاء **كائن** (object) أو **عينة** (instance) **لفئة** (class)، تحتاج أحيانًا إلى تزويد بعض الوسيطات إلى **مُنشئ** (constructor) هذا الكائن. المُنشئ هو وظيفة فئة خاصة تُستدعى عند إنشاء كائن، وتتمثل وظيفتها في تهيئة وسيطات الكائن.

كائن محرك السيرفو المصغر

لإنشاء كائن للمحرك سيرفو:

```
Servo servo;
```

تشير دالة "Servo" إلى نوع الكائن، وتشير دالة "servo" إلى الكائن الفعلي المُستخدم في البرنامج، وهنا لا تحتاج إلى إضافة أي مُعامِلات للتهيئة.

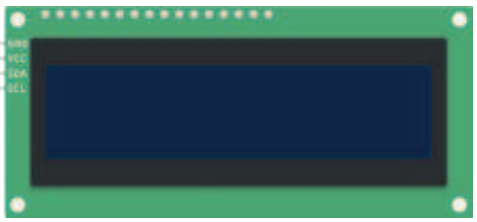
شكل 6.21: محرك السيرفو المصغر (كائن تينكركاد)

كائن شاشة LCD

لإنشاء كائن لشاشة LCD:

```
Adafruit_LiquidCrystal lcd(0);
```

يمكنك باستخدام هذا الأمر تهيئة كائن من نوع Adafruit_LiquidCrystal، وتمرير عنوان الأردوينو الابتدائي الخاص به (وهو 0 افتراضيًا) كوسيط إلى مُنشئ الفئة.



شكل 6.22: شاشة LCD (كائن تينكركاد)

كائن لوحة المفاتيح

تحتاج عملية الإنشاء والتهيئة لكائن لوحة المفاتيح إلى بعض البرمجة لإعداده. ستحتاج في البداية إلى تحديد عدد الصفوف والأعمدة الموجودة في لوحة المفاتيح.

يتم ذلك بهذه الأوامر:

```
const byte numRows = 4; // number of rows on the keypad
const byte numCols = 4; // number of columns on the keypad
```

تُحدد هنا **عدد الصفوف** (numRows) من النوع "const byte" وقيمته هي 4.

وينطبق الشيء ذاته على **عدد الأعمدة** (numCols).

شكل 6.23: لوحة المفاتيح (كائن تينكركاد)

ستحتاج بعد ذلك إلى تحديد المفتاح المضغوط وفقًا للصف والعمود تمامًا كما يظهر على لوحة المفاتيح. طريقة القيام بذلك هي:

```
// keymap defines the key pressed according to the rows and columns just as
   they appear on the keypad


char keymap[numRows][numCols] =
   {
   {'1', '2', '3', 'A'},
   {'4', '5', '6', 'B'},
   {'7', '8', '9', 'C'},
   {'*', '0', '#', 'D'}
   };
```

ستُنشئ هنا مصفوفة keymap باستخدام numRows و numCols التي قُمت بتعريفها سابقًا، وتحديد المفاتيح الموجودة على لوحة المفاتيح بشكل صريح.

ستحتاج بعد ذلك إلى إعداد اتصالات لوحة المفاتيح بأطراف الأردوينو. يمكنك القيام بذلك عن طريق تحديد متغيرين من نوع byte:

```
// Code that shows the the keypad connections to the arduino terminals
byte rowPins[numRows] = {9,8,7,6}; //Rows 0 to 3
byte colPins[numCols] = {5,4,3,2}; //Columns 0 to 3
```

تتمثل الخطوة الأخيرة في تحديد كائن Keypad عن طريق استدعاء مُنشئه، وتوفير وسيطاته اللازمة.

```
// initializes an instance of the Keypad class
Keypad keypad = Keypad(makeKeymap(keymap), rowPins, colPins, numRows, numCols);
```

لاستكمال برنامج الإعداد، ستُعرّف متغيرًا باسم password يقوم بتخزين كلمة مرور قفل الباب، وهو عبارة عن مجموعة من الأحرف بطول 4.

```
char password[4];
```

إيقاف البرنامج Break down the Code

في هذه المرحلة يكون برنامج الإعداد قد اكتمل. وكما تم التوضيح في الدرس الأول، فإن وحدة التحكم في الأردوينو تُنفذ دالة ( )setup مرة واحدة فقط عند تشغيلها، ثم تُنفّذ دالة ( )loop بصورة مستمرة. دعونا الآن نوقف البرنامج.

التعليمات البرمجية لدالة ( )setup هي:

```
void setup()
{
    //servo setup
    servo.attach(11);
    servo.write(0);

    //lcd setup and password set
    lcd.begin(16, 2);
    lcd.setCursor(0, 0);
    lcd.print("Set 4 character");
    lcd.setCursor(0, 1);
    lcd.print("password:");

    for(int i = 0; i < 4; i++) {
       password[i] = keypad.waitForKey();
    }
}
```

تُستخدم دالتي servo من مكتبة Servo كما يلي:

servo.attach(11) تُرفق متغير Servo بالطرف 11.

servo.write(0) تُستخدم لكتابة قيمة إلى servo، في هذه الحالة تُكتب القيمة 0، وتتحكم في عمود الحركة وفقًا لذلك. يحدِّد هذا زاوية العمود في محرك سيرفو القياسي (بالدرجات)، ثم يُحرّك العمود إلى هذا الاتجاه.

تُستخدم بعد ذلك ثلاث دوال لمكتبة Adafruit_LiquidCrystal كالآتي:

lcd.begin(col,row) تهيئ واجهة شاشة LCD، وتحدِّد أبعاد الشاشة (العرض والارتفاع). يجب استدعاء هذه الدالة begin( ) قبل أي أوامر أخرى خاصة بمكتبة LCD. وسيطات هذه الدالة هي:

- cols، وهي عدد الأعمدة الموجودة في الشاشة.
- rows، وهي عدد الصفوف الموجودة في الشاشة؛ ولأن شاشة LCD المُستخدمة16x2، فيمكن إعطاء الدالة الوسيطات col=16 وrow=2 وبالتالي تكون صياغتها lcd.begin (16.2).

الدالة الآتية هي:

تقوم دالة **lcd.setCursor(col,row)** بتحديد الموقع الذي سيُعرض فيه النص المكتوب على شاشة LCD. لعرض عبارة "Set 4 character password" (تعيين كلمة مرور مكونة من 4 أحرف)، فأنت بحاجة إلى كلا الصفين لشاشة LCD. ستُعرض في الصف الأول عبارة "Set 4 character"، وفي الصف الثاني ستُعرض عبارة "password". للقيام بذلك، عليك استدعاء الدالة بالصيغة ;lcd.setCursor(0، 0) قبل عرض العبارة الأولى، ثم استدعاء الدالة مرة أخرى بالصيغة ;lcd.setCursor(0، 1) لعرض العبارة الثانية.

الجزء الأخير من البرمجة في دالة ( )setup هو تكرار for الذي يخزن كلمة مرور تتكون من 4 أحرف يكتبها المستخدِم على لوحة المفاتيح، في متغير password[4]. للقيام بذلك، تُستخدم دالة مكتبة لوحة المفاتيح:

تُستدعى الدالة **keypad.waitForKey( )** والتي ستتعرف على المفتاح الذي تم الضغط عليه، وتخزنه في مصفوفة password (كلمة المرور).

الآن وبالنسبة للمهمة الرئيسة لهذا المشروع، ستُستدعى دالة **التكرار** ( )loop عدة مرات.

أوامر دالة ( )loop هي:

```
void loop()
{
  // clear the screen and display the new message
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Enter password:");

  bool correctPass = true;
  char buttonPressed;

  // this code checks each button pressed against the corresponding password
     digit
  // e.g. it will check the 1st button pressed against the first digit of the
     password and so on
  for (int i = 0; i < 4; i++) {
    buttonPressed = keypad.waitForKey();
    if(password[i] != buttonPressed){
      correctPass = false;
    }
    lcd.setCursor(i, 1);
    lcd.print(buttonPressed);
  }

  delay(1000);

  //this code will be executed if the password is correct
  if (correctPass) {
    // clear the lcd screen
    lcd.clear();
```

```
    // set the cursor to the beginning of the 1st line
    lcd.setCursor(0, 0);
    lcd.print("Correct password!");
    // set the cursor to the beginning of the 2nd line
    lcd.setCursor(0, 1);
    lcd.print("Unlocking...");
    // write the angle by which the servo will rotate
    servo.write(180);
    // wait 5 sec and then rotate the servo to its original angle
    delay(5000);
    servo.write(0);
  }
  else {
    // this code will be executed if the password is wrong
    // clear the lcd screen
    lcd.clear();
    // set the cursor at the 1st col,row
    lcd.setCursor(0, 0);
    // print the message
    lcd.print("Wrong password!");
  }
}
```

إيقاف البرنامج Break this Code down

توجد في البرنامج بعض التعليمات البرمجية لمسح شاشة LCD، ولعرض رسالة تطلب كلمة المرور.

```
  // clear the screen and display the new message
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Enter password:");
```

يتبع ذلك التعليمات البرمجية التي تستقبل كلمة المرور التي يقوم المستخدِم بإدخالها وتتحقق من صحتها. يتم ذلك بالمقارنة بين الأزرار التي يتم الضغط عليها واحدًا تلو الآخر بالتتابع مع رقم كلمة المرور الموجود في نفس الموضع.

على سبيل المثال، لنفترض أن كلمة المرور التي ضُبطت في البداية هي "5456" ويقوم المستخدِم بكتابة كلمة المرور "5453". نظرًا لأن كل مفتاح يضغط عليه المستخدِم سيُقارن بمفتاح كلمة المرور المقابل، فإن ما سيحدث هو:

5 مقارنة مع 5 (نفس الشيء، لا توجد مشكلة)

4 مقارنة مع 4 (نفسها، لا توجد مشكلة)

5 مقارنة مع 5 (لا توجد مشكلة بعد)

3 مقارنة مع 6 (ليستا متطابقتين، لذا فإن كلمة المرور التي تم الضغط عليها غير صحيحة).

عندما يقارن البرنامج بين مفتاحين مختلفين، يجب تحديث المتغير بالمعلومات التي تفيد بأن كلمة المرور غير صحيحة. ولا يُهم ما إذا كان المفتاح الخطأ أول رقم أو آخره أو في أي مكان بينهما، فالنتيجة أن كلمة المرور بأكملها خطأ. لذلك، ولتخزين هذه المعلومات، يمكنك استخدام متغير منطقي يتم تهيئته على أنه **صائب** (true)، وعند الضغط على مفتاح خطأ، تتغير قيمة المتغير إلى **خطأ** (false). وبعد إجراء المقارنة يمكنك التحقق من قيمة هذا المتغير، وإذا كانت هذه القيمة صائبة، فهذا يعني أن المستخدِم كتب كلمة المرور الصحيحة، أما إذا كانت هذه القيمة خطأ، فهذا يعني أن المستخدِم كتب كلمة مرور خطأ.

يتم تنفيذ الدالة التي تم وصفها أعلاه بواسطة هذا الجزء من البرنامج:

```
bool correctPass = true;
char buttonPressed;


// this code checks each button pressed against the corresponding password
   digit
// e.g. it will check the 1st button pressed against the first digit of the
   password and so on
for (int i = 0; i < 4; i++) {
  buttonPressed = keypad.waitForKey();
  if(password[i] != buttonPressed){
    correctPass = false;
  }
  lcd.setCursor(i, 1);
  lcd.print(buttonPressed);
}
```

والآن نصل إلى الجزء المهم من البرنامج وهو عملية فتح الباب (تدوير محرك سيرفو) إذا كانت كلمة المرور المكتوبة صحيحة، وغلق القفل مرة أخرى بعد فترة من الوقت، أو عرض رسالة تفيد بأن كلمة المرور كانت خطأ.

يتم تنفيذ هذه الدالة بواسطة التعليمات البرمجية الآتية:

```
// this code will be executed if the password is correct
if(correctPass){
   // clear the lcd screen
   lcd.clear();
   // set the cursor to the beginning of the 1st line
   lcd.setCursor(0, 0);
   lcd.print("Correct password!");
   // set the cursor to the beginning of the 2nd line
   lcd.setCursor(0, 1);
   lcd.print("Unlocking...");
   // write the angle by which the servo will rotate
   servo.write(180);
   // wait 5 sec and then rotate the servo to its original angle
   delay(5000);
   servo.write(0);
}
else {
   // this code will be executed if the password is wrong
   // clear the lcd screen
   lcd.clear();
   // set the cursor at the 1st col,row
   lcd.setCursor(0, 0);
   // print the message
   lcd.print("Wrong password!");
   }
```

ختامًا، سيبدو البرنامج بأكمله لمشروع القفل الذكي للباب كالآتي:

البرنامج بأكمله Complete Code

اختر وضع البرمجة نص (Text) في محرر التعليمات البرمجية.

```
// C++ code
//
#include <Adafruit_LiquidCrystal.h>
#include <Keypad.h>
#include <Servo.h>

Adafruit_LiquidCrystal lcd(0);
Servo servo;

const byte numRows = 4; //number of rows on the keypad
const byte numCols = 4; //number of columns on the keypad

// keymap defines the key pressed according to the rows and columns just as they
   appear on the //keypad
char keymap[numRows][numCols] =
{
{'1', '2', '3', 'A'},
{'4', '5', '6', 'B'},
{'7', '8', '9', 'C'},
{'*', '0', '#', 'D'}
};

// Code that shows the the keypad connections to the arduino terminals
byte rowPins[numRows] = {9,8,7,6}; //Rows 0 to 3
byte colPins[numCols] = {5,4,3,2}; //Columns 0 to 3

// initializes an instance of the Keypad class
Keypad keypad = Keypad(makeKeymap(keymap), rowPins, colPins, numRows, numCols);
```

```
char password[4];

void setup()
{
  // servo setup
  servo.attach(11);
  servo.write(0);

  // lcd setup
  lcd.begin(16, 2);
  // lcd print 1st line
  lcd.setCursor(0, 0);
  lcd.print("Set 4 character");
  // lcd print 2nd line
  lcd.setCursor(0, 1);
  lcd.print("password:");

  // gets and stores the password
  for(int i = 0; i < 4; i++){
    password[i] = keypad.waitForKey();
  }
}

void loop() {
  // clear the screen and display the new message
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Enter password:");

  bool correctPass = true;
  char buttonPressed;
```

```
  // this code checks each button pressed against the corresponding password digit
  // e.g. it will check the 1st button pressed against the first digit of the
     password and so on
  for(int i = 0; i < 4; i++) {
    buttonPressed = keypad.waitForKey();
    if(password[i] != buttonPressed) {
      correctPass = false;
    }
    lcd.setCursor(i, 1);
    lcd.print(buttonPressed);
  }

  delay(1000);

  //this code will be executed if the password is correct
  if (correctPass){
  // clear the lcd screen
  lcd.clear();
  // set the cursor to the beginning of the 1st line
  lcd.setCursor(0, 0);
  lcd.print("Correct password!");
  // set the cursor to the beginning of the 2nd line
  lcd.setCursor(0, 1);
  lcd.print("Unlocking...");
  // write the angle by which the servo will rotate
  servo.write(180);
  // wait 5 sec and then rotate the servo to its original angle
  delay(5000);
  servo.write(0);
  }
  else {
// this code will be executed if the password is wrong
```

```
  // clear the lcd screen
  lcd.clear();
  // set the cursor at the 1st col,row
  lcd.setCursor(0, 0);
  // print the message
  lcd.print("Wrong password!");
  }
}
```

بعد الانتهاء من كتابة البرنامج، اضغط على زرّ بدء المحاكاة (Start Simulation).

تمرينات

1 أنشئ دائرة في برنامج تينكركاد تتصل بمُستشعر درجة الحرارة وشاشة LCD، ثم قم ببرمجتها باستخدام لغة C++ لعرض درجة الحرارة التي يقوم المُستشعر بقراءتها على شاشة LCD.

2 أنشئ دائرة في برنامج تينكركاد تتصل بلوحة مفاتيح 4x4 وشاشة LCD، ثم قم ببرمجتها باستخدام لغة C++ لعرض الأحرف التي يتم الضغط عليها وذلك على شاشة LCD.

3 أنشئ دائرة في برنامج تينكركاد تتصل بلوحة مفاتيح 4x4 ودايودين مشعين للضوء، أحدهما أحمر والآخر أخضر. سيقوم المستخدِم بتعيين كلمة مرور ثم سيحاول استخدامها. إذا كان الإدخال صائبًا، سيضيئ الدايود المشع للضوء الأخضر، وإذا كان خطأ، فإنّ الدايود المشع للضوء الأحمر سيومض بصورة متكررة.

4 أنشئ دائرة في برنامج تينكركاد تتصل بمُستشعر رطوبة التربة ومحرك سيرفو، ثم قم ببرمجتها باستخدام لغة C++ لتشغيل محرك سيرفو عند وصول درجة رطوبة التربة إلى قيمة معينة.

المشروع

يُعدُّ نظام الحماية الذكية جزءًا واحدًا فقط من نظام المنزل الذكي الكامل لإنترنت الأشياء. توجد العديد من التطبيقات الأخرى لإنترنت الأشياء المنزلية، من أهمها تنظيم درجة الحرارة.

في هذا المشروع ستقوم بتوسعة الدائرة وبرمجة مشروع قفل الباب الذكي من جديد لإضافة المزيد من المكونات الإلكترونية للتحكم في درجة الحرارة المنزلية.

1

القراءات البيئية التي يجب مراقبتها هي درجة الحرارة والساعة الحالية.

ستُراقب درجة الحرارة بواسطة مُستشعر درجة الحرارة، والحصول على الوقت بواسطة **ترانزستور ضوئي** (Phototransistor) يشير إلى مستويات الضوء خارج المنزل.

2

قم بتوصيل محرك التيار المستمر بالدائرة التي تمثل مُنظم الحرارة وشاشة LCD أخرى. ستعرض شاشة LCD درجة الحرارة الحالية بالدرجات المئوية. سيُنشّط محرك التيار المستمر عن طريق إشارة تناظرية اعتمادًا على القراءات من البيئة المحيطة.

3

أنشئ مستويات مختلفة من درجات الحرارة وظروف الإضاءة التي ستُرسل قيمًا تناظرية مختلفة إلى محرك التيار المستمر.

تحتاج البيئات الأكثر برودة إلى المزيد من المُخرجات من منظم الحرارة (محرك التيار المستمر). أنشئ الدائرة وقم ببرمجتها باستخدام لغة C++ لتمثيل التنظيم التلقائي لدرجة الحرارة.

ماذا تعلّمت

- تحديد المزايا والمخاطر لأنظمة الأمان المبنية على إنترنت الأشياء.
- تعيين أمثلة حول أجهزة إنترنت الأشياء المستخدمة في أنظمة الحماية الذكية.
- استخدام الأوامر الأساسية في لغة C++.
- برمجة جهاز تحكم الأردوينو الدقيق باستخدام لغة C++.
- إنشاء دائرة إلكترونية في تينكركاد وبرمجتها باستخدام لغة C++.

المصطلحات الرئيسة

شاشة LCD	LCD display
البرمجة الكائنية	Object-Oriented Programming
كائن	Object
الحماية الذكية	Smart Security

لغة برمجة C++	C++ Programming Language
فئة	Class
لغة برمجة عالية المستوى	High Level Programming Language
لوحة مفاتيح	Keypad

7. الرسائل في إنترنت الأشياء

سيتعرف الطالب في هذه الوحدة على التطبيقات الخاصة بالمدن الذكية، وعلى أساسيات بروتوكول نقل القياس عن بعد في قائمة انتظار الرسائل (Message Queuing Telemetry Transport - MQTT)، كما سينشئ تطبيق إنترنت الأشياء باستخدام متحكم الأردوينو وبروتوكول (MQTT)، وفي الختام سيقوم بإجراء عمليات لتحليل البيانات على التطبيق المدمج.

أهداف التعلُّم

بنهاية هذه الوحدة سيكون الطالب قادرًا على أن:

- يتعرّف على طبقات هيكلية المدن الذكية.
- يُحدِّد أمثلة على المدن الذكية.
- يصف وظيفة بروتوكول (MQTT).
- يُصنِّف جودة الخدمة (QoS) لبروتوكول (MQTT).
- يستخدِم البرمجة النصية في بايثون لنشر الرسائل إلى عميل MQTT X.
- يُنشئ ملف بيانات جسون (JSON) لتخزين التقارير.
- يستخدِم مُفكرة جوبيتر (Jupyter) لإجراء عمليات تحليل البيانات في ملف بيانات (JSON).

الأدوات

- بيئة التطوير المتكاملة للأردوينو (Arduino IDE)
- أداة جيت برينز باي تشارم (JetBrains PyCharm)
- بيئة محاكاة دوائر أوتوديسك تينكركاد (Autodesk Tinkercad Circuits)
- عميل MQTT X

رابط الدرس الرقمي

www.ien.edu.sa

الدرس الأول

المدن الذكية وبروتوكول MQTT

المُدن الذكية Smart Cities

بدأت غالبية المُدن كمراكز حَضرية متواضعة، وذلك دون وجود تخطيط مُسبق يدعم متطلبات الزيادة السكانية المتسارعة. يُؤثر التوسع العُمراني المطّرد للمُدن على بنيتها التّحتية وخدماتها المختلفة، حيث يتم تجاوز الطاقة الاستيعابية القصوى للطرق والجسور وأنظمة الصرف الصحي، مما يُصعِّب من طبيعة الحياة فيها، ويجعل توفير الأساسيات مثل الماء والكهرباء وتقليل الانبعاثات الكربونية يمثل تحديًا مباشرًا في هذه المدن.

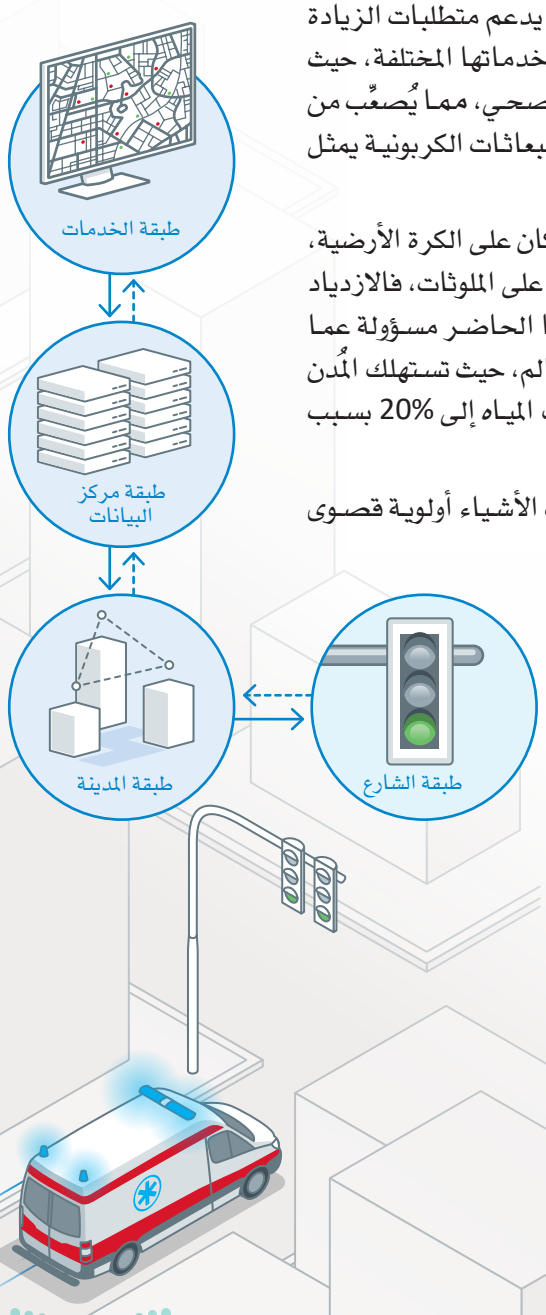
تزداد الانبعاثات الحرارية واستهلاك الطاقة مع الزيادة الكبيرة في عدد السكان على الكرة الأرضية، كما أن تركُّز السكان في مناطق معينة يَحدُّ من قدرة النظام البيئي على التغلب على الملوثات، فالازدياد في الانبعاثات والنفايات يسهم في تسارع التغير المناخي. وتُعدُّ المُدن في وقتنا الحاضر مسؤولة عما يقرب من 80%–60 من انبعاثات الطاقة وغازات الاحتباس الحراري في العالم، حيث تستهلك المُدن 60% من مجموع المياه الصالحة للشرب، بينما تصل نسبة الفاقد من تلك المياه إلى 20% بسبب التسريبات في شبكات المياه.

يُعدُّ تحسين الموارد ومعالجة النفايات والانبعاثات باستخدام تقنيات إنترنت الأشياء أولوية قصوى لجميع السلطات المسؤولة عن المدن في جميع أنحاء العالم.

هيكلية المدينة الذكية باستخدام إنترنت الأشياء
A Smart City IoT Architecture

يتمثل التحدي الرئيس لحلول إنترنت الأشياء الذكية في ربط أنظمة معقدة متعددة في تقنية موحدة، وتوجد العديد من مُخططات المُدن الذكية المقترحة، ومن أبرزها المُخطط المبني على تقسيم شبكة إنترنت الأشياء للمدن الذكية إلى أربع طبقات، وتشمل كل من طبقة **الشارع** (Street)، وطبقة **المدينة** (City)، وطبقة **مركز البيانات** (Data Center)، وطبقة **الخدمات** (Services).

تنتقل البيانات من الأجهزة الموجودة في طبقة الشارع إلى طبقة شبكة المدينة، حيث يتم دمجها وتوحيدها وتخزينها، وتقوم طبقة مركز البيانات بتغذية المعلومات في طبقة الخدمات التي تشتمل على تطبيقات مزود الخدمة للمدينة.

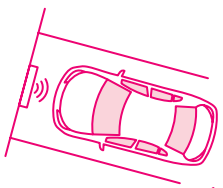
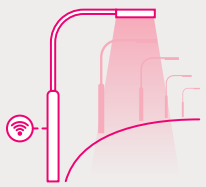
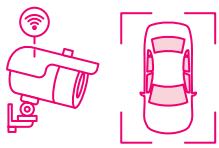
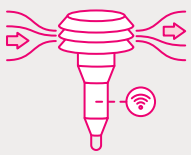
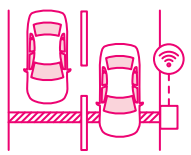


شكل 7.1: مُخطط إنترنت الأشياء في المدينة الذكية

طبقة الشارع Street Layer

تتكون طبقة الشارع من أجهزة ومُستشعرات تجمع البيانات وتعمل وفقًا لمتطلبات النظام المتكامل، وذلك وفقًا لمكونات الشبكات اللازمة لجمع هذه البيانات واختزالها، وتُستخدم في طبقة الشارع مجموعة من الأجهزة ذات الاستخدامات المختلفة في المُدن الذكية، كما يظهر في الجدول أدناه:

جدول 7.1: أجهزة ومُستشعرات طبقة الشارع

النوع	الوصف
 مُستشعر مغناطيسي (Magnetic Sensor)	يكتشف المُستشعر المغناطيسي عملية رَكْنِ المركبات، من خلال مراقبة التغيرات في المجال المغناطيسي له عند اقتراب جسم معدني ثقيل مثل سيارة أو شاحنة.
 مُستشعرات الإضاءة (Lighting Controller)	يمكن لمُستشعر الإضاءة التحكم في الإنارة بناءً على المتغيرات البيئية وعلى الوقت.
 كاميرات المراقبة (Video Cameras)	يمكن لكاميرات المراقبة وتقنيات تحليل الصور التعرف على السيارات والوجوه وحالة المرور في مختلف التطبيقات الخاصة بالمرور والحماية.
 مُستشعر جودة الهواء (Air Quality Sensor)	يمكن لمُستشعر جودة الهواء اكتشاف وتحديد كميات الغازات والجسيمات في الهواء، لمعرفة كمية التلوث بدقة في موقع محدد.
 العدادات (Counters)	تستطيع العدادات المثبتة في الشوارع تسجيل عدد المركبات المُتحركة في الشارع أو المتوقفة في منطقة وقوف عامة، وذلك من أجل توفير التحليلات المرورية وحالة المرور للسائقين.

طبقة المدينة City Layer

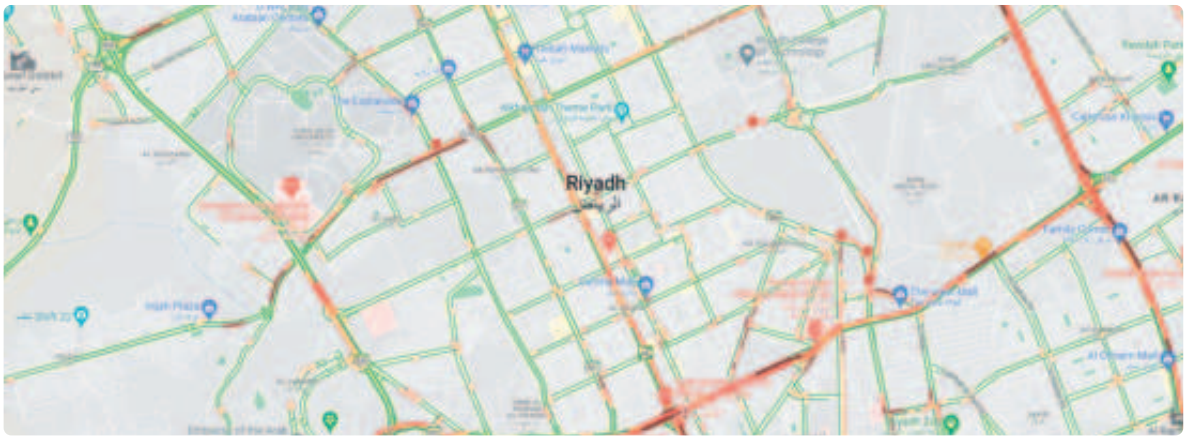
يُمكن النظر إلى طبقة المدينة على أنها طبقة النقل المباشرة بين أجهزة إنترنت الأشياء الطرفية (Edge Devices IoT)، ومراكز البيانات أو الإنترنت. ويجب نشر موجهات الشبكة ومحولاتها في هذه الطبقة في مستوى أعلى من طبقة الشارع لدعم نقل البيانات الضخمة، كما يجب أن تنقل طبقة المدينة البيانات من خلال أنواع عديدة من البروتوكولات لتطبيقات إنترنت الأشياء المختلفة. قد تتسم بعض هذه التطبيقات بالحساسية للتأخر الزمني أو لفقدان الحِزم، حيث يمكن أن يؤدي فقدان حِزمة معينة إلى تشغيل تنبيه أو إنشاء تقرير غير صحيح. لذلك، يجب أن تكون طبقة المدينة مَرنة لضمان وصول حِزمة البيانات المُرسلة من مُستشعر أو بوابة إلى وجهتها دائمًا.

طبقة مركز البيانات Data Center Layer

تُرسل البيانات التي تجمعُها المُستشعرات إلى مركز البيانات للمعالجة والتخزين، وبناءً على عمليات معالجة البيانات هذه، تُحدّد المعلومات والأنماط المهمة، ثم إنشاء الأفكار ودعم القرارات. على سبيل المثال، يمكن لمركز البيانات إعطاء تصور لحركة المرور على مستوى المدينة، ومساعدة السلطات في تحديد أهمية إضافة مركبات نقل جماعي إضافية أو تقليلها. ويُمكن استخدام نفس بيانات حركة المرور لإدارة مدة عمل إشارات المرور في المدينة ومزامنتها تلقائيًا للتحكم بالازدحام المروري. وتُعدُّ الخدمات السحابية وتخزين البيانات سحابيًا ضرورة لتطوير أي حل شامل لإنترنت الأشياء، كما يمكن تخزين هذه البيانات في مراكز البيانات التابعة لسلطات المدينة أو للشركات الخاصة، وذلك حسب التشريعات المحلية.

طبقة الخدمات Services Layer

تكْمُن الأهمية الفعلية لإنترنت الأشياء في الخدمات التي تقدمها للسلطات والمواطنين، ويجب عرض البيانات التي تُعالج وفقًا للمتطلبات الخاصة بكل مُستهلك للبيانات، ووفق متطلبات تجربة المستخدِم وحالات الاستخدام المختلفة. يمكن مثلًا إعادة توجيه الحافلات وأنظمة النقل العام الأخرى لتجنب مواقع الازدحام المروري المتكررة، كما يمكن تسيير المزيد من قطارات الأنفاق بصورة تلقائية وذلك استجابةً لزيادة الازدحام المروري، وتوقع قرارات الركاب باختيار وسائل النقل العام للتنقل بدلًا من السيارات الخاصة في الأيام التي تشهد حركة مرور مُكثفة.



شكل 7.2: التحديث الفوري للحركة المرورية

مثال

تُخطط وزارة الشؤون البلدية والقروية والإسكان لتنفيذ أكثر من 50 مشروعًا لمُدن ذكية متصلة بإنترنت الأشياء بحلول عام 2030. تشمل هذه المشاريع الإدارة الذكية لحركة المرور ومواقف السيارات وأنظمة الحفاظ على البيئة، وكذلك إدارة التخلص من النفايات، والإسكان الذكي، وأنظمة إدارة الأراضي، وذلك لتحقيق الهدف الرئيس وهو تحسين نوعية حياة المواطنين وتحقيق الاستدامة المالية وجودة الخدمة.

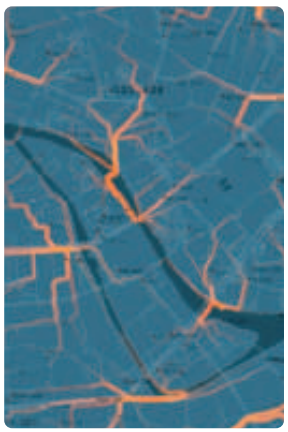
تطبيقات المدينة الذكية Smart City Applications

الإنارة الذكية للشوارع Connected Street Lighting

شكل 7.3: الإنارة الذكية للشوارع

تُعدُّ إنارة الشوارع إحدى أكثر المرافق الحضرية تكلفة، لكونها تمثل ما يصل إلى40% من إجمالي تكلفة الطاقة، وتبحث المُدن عن طرائق لخفض تكاليف الإنارة مع تحسين كفاءة التشغيل وخفض النفقات الأولية. يمكن أن يؤدي تثبيت نظام الإنارة الذكية للشوارع إلى توفير كبير في الطاقة مما يُتيح المجال لتقديم خدمات جديدة. ويُعدُّ الانتقال إلى استخدام تقنيات **الدايودات المشعة للضوء** (LEDs) في طليعة الوسائل المُستخدمة للتحول من الإنارة التقليدية إلى الإنارة الذكية للشوارع، وتتميز الدايودات المشعة للضوء بالاستهلاك المُنخفض للطاقة، مما يجعلها مناسبة بشكل مثالي لتطبيقات الحلول الذكية. ويُمكن أيضًا تعديل ألوان الدايودات المشعة للضوء وشدتها وفقًا للحاجة والظروف المحيطة.

التحكم الذكي في الحركة المرورية Smart Traffic Control



شكل 7.4: التحكم الذكي في الحركة المرورية

يُعدُّ الازدحام المروري من أكثر المشاكل شيوعًا في المُدن الحديثة، حيث يسهم بشكل كبير في التلوث البيئي وفقدان الإنتاجية، ويشمل الحل الذكي لضبط وتنظيم الحركة المرورية في المُدن توفير المعلومات حول عدد السكان، وحركة التنقل، وأعداد المركبات على الطريق، حيث يتم إرسال تلك البيانات إلى المسؤولين عن تخطيط وتنظيم حركة المرور لاتخاذ الإجراءات اللازمة. ومن الممكن تفعيل التطبيقات المرورية من خلال البيانات الواردة من مُستشعرات إنترنت الأشياء وذلك لتخفيف الازدحام والتحكم في الحركة المرورية، ويستطيع مخططو المُدن من خلال تحليل البيانات التي تم جمعها خلال فترات زمنية معينة إنشاء استراتيجيات أكثر فاعلية لتقليل الازدحام المروري. يتسبب الازدحام المروري في ارتفاع حوادث المرور، والتي بدورها تزيد من الازدحام المروري حتى تلك البسيطة منها، ويتمثل أحد الحلول الشائعة في التحكم في تدفق حركة السيارات بناءً على كثافة الحركة المرورية. يمكن للتطبيق الذي يكتشف كثافة الحركة المرورية الفورية تنظيم مدة دورة إشارة المرور لتقييد أو إزالة تأثير الازدحام المروري بالتحكم في عدد المركبات المنضمة إلى حركة المرور على الطرق الرئيسة.

البيئة المتصلة Connected Environment

شكل 7.5: محطة جودة هواء ذكية

تُراقب غالبية المُدن الكبيرة جودة الهواء، ولكن الكثير من محطات مراقبة جودة الهواء تستخدِم معدات مُراقبة قديمة ومُكلفة لجمع هذه البيانات، وعادةً ما تجمع هذه المحطات قراءات دقيقة جدًا، ولكنها تتسم بمحدودية المدى الذي تجمع منه البيانات، وبالتالي يُحتمل ألا يتم تغطية كامل المدينة بالنقاط الكافية، وتؤدي محدودية البيانات التي تُجمع إلى عدم القدرة على تحديد أنماط جودة الهواء بشكل صحيح. إن تكلفة محطات مراقبة جودة الهواء وحجمها يجعلان من الصعب توفير العدد الكافي من هذه المحطات لتوفير معلومات موثوقة على مستوى محلي وتتبع انتقال التلوث في أرجاء المُدن على مدى فترة زمنية معينة.

تنبيهات الأمان الذكية Smart Safety Alerts

شكل 7.6: وحدة الاتصال على جانب الطريق (RSU)

توجد على جانب الطريق وحدة **اتصالات مخصصة للاتصالات قصيرة المدى** (Dedicated Short-Range Communications – DSRC) تعمل كبوابة بين **وحدة التواصل داخل المركبة** (On-Board Unit – OBU) والبنية التحتية للاتصالات، كما تعمل **وحدة الاتصال على جانب الطريق** (Roadside Unit – RSU) بمثابة جهاز اتصال لاسلكي على جانب الطريق وتُوفِّر الاتصال ودعم المعلومات للمركبات المارة بما فيها تحذيرات السلامة والمعلومات المرورية.

تتطلب المدينة الذكية دائمًا معرفة فورية وشاملة بجودة الهواء، ولجمع هذه البيانات تتطلب المُدن الذكية ما يلي:

- أنظمة بيانات مفتوحة تتلقى قياسات جودة الهواء من محطات المراقبة الموجودة.
- مُستشعرات إنترنت الأشياء منخفضة التكلفة وذات مستوىً من الدقة مماثلًا لذلك الذي يمكن الحصول عليه من محطات جودة الهواء.
- إمكانية تمثيل للبيانات البيئية متوافر للسلطات وللمواطنين، وتخزين سجلات بيانات جودة الهواء السابقة لتتبع الانبعاثات زمنيًا وتحديد اتجاهاتها.

مثال

يهدف مشروع ذا لاين (The Line) في مدينة نيوم الكبرى في المملكة العربية السعودية إلى دمج أحدث تقنيات المُدن الذكية المتطورة لتصبح البيئة الحضرية الأكثر تقدمًا من الناحية التقنية.

ستعتمد نيوم بشكل كبير على حلول إنترنت الأشياء للمُدن الذكية للوصول إلى هدفها المتمثل في أن تصبح مدينة خالية من الانبعاثات، دون سيارات أو ازدحام مروري.

بروتوكول نقل القياس عن بُعد في قائمة انتظار الرسائل

Message Queuing Telemetry Transport - MQTT

مقدمة إلى بروتوكول (MQTT)

Introduction to MQTT

طرح المهندسون من شركتي IBM و Arcom في نهاية التسعينيات من القرن الماضي فكرة تطوير بروتوكول غير معقد وموثوق وفعّال، وكذلك منخفض التكلفة، وذلك لمراقبة الأعداد الكبيرة من المُستشعرات، وإدارتها والتعامل مع بياناتها من موقع خادم مركزي، وتحديدًا للاستخدام في قطاعي النفط والغاز، نتج عنه تطوير بروتوكول **نقل القياس عن بُعد في قائمة انتظار الرسائل** (MQTT)، والذي تم توحيده الآن من قِبَل **مؤسسة المعايير الدولية المنظمة** (OASIS). يُستخدم بروتوكول (MQTT) على نطاقٍ أوسع من استخدام بروتوكول (HTTP) في تطبيقات إنترنت الأشياء، وذلك بشكل أساسي لسهولة إنشاء هياكل مُعقدة باستخدام الأجهزة التي تُرسل وتستقبل حزم البيانات.

أساسيات MQTT MQTT Basics

يمكن لعميل (MQTT) أن يكون **ناشرًا** (Publisher) لإرسال البيانات إلى خادم (MQTT) يعمل كخادم رسائل، ويسمى أيضًا **بوسيط الرسائل** (Message Broker). يتلقى خادم (MQTT) الاتصال من شبكة الناشرين ورسائل التطبيق، ويدير هذا الخادم أيضًا عمليات الاشتراك وإلغاء الاشتراك ويقدم بيانات التطبيق لعملاء (MQTT) الذين يعملون **كمُشتركين** (Subscribers). يُمكن للعملاء الاشتراك لاستلام كافة البيانات أو جزء منها من مجموعة معلومات الناشر (MQTT). ويُطلق على عميل (MQTT) في هذه الحالة اسم **المُشترك** (Subscriber). يؤدي تضمين وسيط الرسائل في (MQTT) إلى الفصل بين عملية نقل البيانات بين الناشرين والمُشتركين، فالناشرون والمُشتركون يجهلون بعضهم بعضًا، ويضمن وسيط رسائل (MQTT) إمكانية تأخير المعلومات وتخزينها في حالة فشل الشبكة، وهو ما يُعدُّ ميزة لعملية الفصل بين الناشرين والمُشتركين، ولهذا السبب، لا يُطلب من الناشرين والمُشتركين الاتصال بالإنترنت في آنٍ واحد. وتتكون جلسة (MQTT) لكل عميل وخادم من أربعة مراحل وهي: إنشاء الجلسة، والمصادقة، وتبادل البيانات، وإنهاء الجلسة، وكل عميل يتصل بخادم لديه مُعرف عميل فريد يحدد جلسة (MQTT) بين الطرفين، ويعامل الخادم كل عميل على حدة عند إرسال رسالة تطبيق إلى العديد من العملاء. من عيوب بروتوكول (MQTT) أنه أبطأ في الإرسال من بروتوكول (HTTP)، كما أنه يجب تنفيذ اكتشاف الموارد وخدمات النسخ الاحتياطي من قِبل المستخدِم، ويؤخذ على هذا البروتوكول أيضًا قصوره الأمني في عملية التشفير، وكذلك صعوبة توسيع نطاقه مع زيادة عدد الأجهزة والوسطاء.

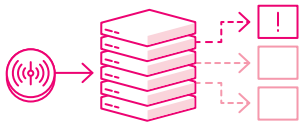
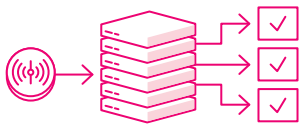
شكل 7.7: وظيفة MQTT

جودة الخدمة Quality of Service - QoS

يوفر بروتوكول (MQTT) ثلاث درجات من **جودة الخدمة** (QoS)، وتُطبّق جودة الخدمة لبروتوكول (MQTT) أثناء تبادل رسائل التطبيق مع الناشرين أو المُشتركين، كما يتعلق بروتوكول التسليم بشكل أساسي بتسليم رسائل التطبيق من مُرسل واحد إلى مُستلم آخر.

يعرض الجدول الآتي مستويات جودة الخدمة الثلاثة لبروتوكول (MQTT):

جدول 7.2: جودة مستويات الخدمة

المستوى	الوصف
 **مستوى جودة الخدمة 0 (مرة واحدة على الأكثر):** • لا يمكنه التعامل مع الفشل. • لا يتكرر أبدًا.	هذه خدمة بيانات غير مؤكدة، وتعرف باسم أفضل جهد أو (مرة واحدة على الأكثر). يُسلم الناشر رسالة واحدة إلى الخادم الذي ينقلها إلى كل مُشترك، ولا يستقبل المُستلم أي إجابة، ولا يحاول المُرسل إرسال البيانات مرة أخرى. يتلقى المُستلم الرسالة إما مرة واحدة أو لا يتلقاها على الإطلاق.
 **مستوى جودة الخدمة 1 (مرة واحدة على الأقل):** • يستطيع التغلب على فقدان الاتصال. • يمكن أن يتكرر.	يضمن مستوى جودة الخدمة هذا إرسال الرسائل مرة واحدة على الأقل بين الناشر والخادم، ثم بين الخادم والمُشتركين. يضمن هذا المستوى التسليم مرة واحدة على الأقل.
 **مستوى جودة الخدمة 2 (مرة واحدة بالضبط):** • يستطيع التغلب على فقدان الاتصال. • لا يمكن أن يتكرر.	يُعدُّ هذا أعلى مستوى لجودة الخدمة، ويُستخدم في الحالات التي لا تسمح بفقدان الرسالة أو تكرارها. يحتوي مستوى جودة الخدمة هذا على تكلفة إضافية نظرًا لأن كل حزمة تتضمن متغيرًا اختياريًا يحتوي على تعريف الحزمة، ويوفر هذا المستوى "خدمة مضمونة" تسمى التسليم "مرة واحدة بالضبط"، ولا يهم عدد مرات إعادة المحاولة طالما تم إرسال الرسالة مرة واحدة بدقة.

مثال

يُمكن أن تتعرض كائنات إنترنت الأشياء في المُدن الذكية للمخاطر بسبب هيكليتها المركزية، حيث لا تتناسب أساليب الحماية التقليدية مع بيئة إنترنت الأشياء المتطورة. ستُطوّر في المملكة العربية السعودية تقنيات سلسلة الكُتل (Blockchain) لإنترنت الأشياء في المُدن الكبرى لتقليل النقاط المركزية لحالة فشل الشبكة التي تعتمد على الهيكلية الموزعة. ستعتمد شبكة مشروع نيوم العملاق على تقنيات سلسلة الكُتل لإنترنت الأشياء لتوفير بُنية تحتية آمنة للشبكة ولمواطنيها.

تمرينات

1

حدِّد الجملة الصحيحة والجملة الخاطئة فيما يلي:	صحيحة	خاطئة
1. تُطوّر تقنيات المُدن الذكية لتحسين الحركة المرورية فقط.	○	○
2. يجب أن تكون موجهات الشبكة في طبقة المدينة مرنة لمواجهة حالات فُقدان البيانات المُحتملة في الحزم.	○	○
3. تُرسل البيانات مُباشرة من طبقة الشارع إلى طبقة مركز البيانات.	○	○
4. يمكن تخزين البيانات الموجودة في طبقة مركز البيانات على الخوادم الخاصة للشركات.	○	○
5. تحتوي طبقة الخدمات على التطبيقات التي يستخدمها سكان المدينة.	○	○
6. ينحصر استخدام أنظمة إنارة الشوارع الذكية على الدايودات المشعة للضوء (LEDs).	○	○
7. لا يُمكن استخدام البيانات التاريخية التي جُمعت على مدى فترات معينة في الماضي لتوقُّع الحركة المرورية المستقبلية.	○	○
8. يمكن استخدام حلول بيئية مبنيّة على إنترنت الأشياء للحد من الانبعاثات الضارة داخل المُدن.	○	○
9. أُنشئ بروتوكول (MQTT) لربط العديد من المُستشعرات من خلال نقطة خدمة واحدة.	○	○
10. في الاتصال الأساسي ببروتوكول (MQTT)، يُدرك الناشر والمُشترك وجود الطرف الآخر.	○	○

2 ما الدافع الأساسي وراء تطوير المُدن الذكية؟ دوِّن أفكارك أدناه.

3 أنشئ مُخططًا يوضح كيفية تدفق البيانات في هيكلية إنترنت الأشياء في المدينة الذكية.

4 اعرض أمثلة حول استخدام المُستشعرات في طبقة الشارع في المدينة الذكية.

5 صِف كيف يُمكن استخدام الأنظمة المتطابقة في طبقة مركز البيانات في تطبيقات متعددة. دوِّن أفكارك أدناه.

6 اعرض مثالين على تطبيقات المُدن الذكية وصِفهما بإيجاز. دوِّن أفكارك أدناه.

7 صِف باختصار آلية عمل بروتوكول (MQTT).

8 صَنِّف درجات جودة الخدمة الثلاث لبروتوكول (MQTT).

9 أنشئ مخططًا لمثال على ثلاثة أجهزة تتصل بواسطة بروتوكول (MQTT)، أحدها كناشر، والجهازان الآخران كمُشتركين.

رابط الدرس الرقمي

www.ien.edu.sa

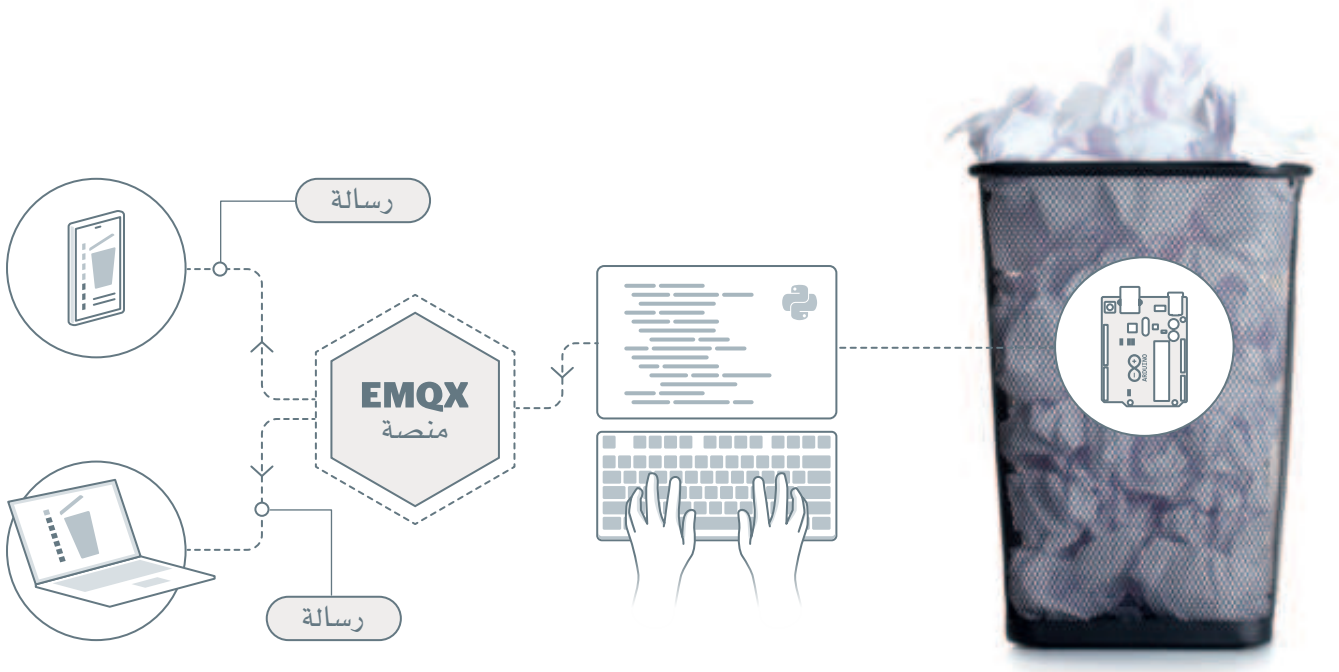
الدرس الثاني

تصميم وبرمجة جهاز ذكي لإنترنت الأشياء

إدارة النفايات الذكية

Smart Waste Management

نظرًا للاكتظاظ السكاني قد لا تُجمع ولا تُعالج كميات كبيرة جدًا من النفايات والمُخلفات بكفاءة، مما يتسبب في زيادة كمية النفايات في عدة أماكن، وتحدث هذه المشكلة بسبب تجاوز سعة حاويات القمامة دون إزالتها في الوقت المناسب. لكن باستخدام حاويات النفايات الذكية، يمكن أن تُرسل رسائل تنبيهية لإعلام مركبات جمع النفايات بهذه الحاويات. كذلك من خلال عمليات تحليل البيانات الملاءمة يُمكن استنباط الأفكار حول كيفية تعبئة حاويات النفايات لتحسين العملية برمتها بشكل أكثر كفاءة.



شكل 7.8: مشروع إدارة النفايات الذكية بالأردوينو وبروتوكول MQTT

ستُنشئ في هذا الدرس نموذجًا أوليًا لحاوية قمامة ذكية تحسب متوسط عدد المرات المطلوبة لتصل إلى سعتها الكاملة. ستُرسل رسالة إلى وسيط (MQTT) كل مرة تُستخدم فيها الحاوية، وعندما تمتلئ الحاوية، تُرسل رسالة أخرى إلى مُتحكم النظام الذي يُنتج التقارير عن الحاوية. ستستخدِم في هذا المشروع مُتحكم أردوينو يُمثل حاوية ذكية، وستقوم ببرمجته باستخدام بروتوكول (Firmata) والبايثون، كما ستستخدِم منصة (MQTT) لتوزيع الرسائل.

EMQX

(EMQX) هو وسيط (MQTT) مفتوح المصدر عالي الأداء مع مُحرك لمعالجة الرسائل بصورة فورية. يُستخدم لدعم تدفق الأحداث بواسطة عدد كبير من أجهزة إنترنت الأشياء.

مكونات وأدوات المشروع Components & Tools for Project

الترانزستور الضوئي Phototransistor

هو مُكون كهربائي يعمل عند تعرضه للضوء، حيث تتدفق كمية متناسبة من التيار العكسي عند سقوط الضوء على المُستشعر، وتُستخدم أجهزة الترانزستورات الضوئية على نطاق واسع لاكتشاف وتحويل نبضات الإضاءة إلى إشارات كهربائية.

ترانزستور ضوئي

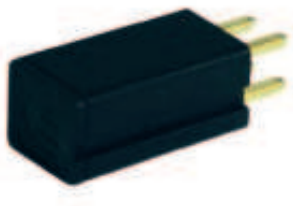
المكوّن في المحاكي

الرمز التخطيطي

شكل 7.9: مُستشعر ضوئي

مُستشعر الإمالة Tilt Sensor

يُستخدم مُستشعر الإمالة لقياس درجة الميل على عدة محاور، وتقوم مُستشعرات الإمالة بتقييم وضع الميل بالنسبة للجاذبية وتُستخدم في تطبيقات مختلفة، حيث تجعل اكتشاف الميل أو الاتّجاه أمرًا ميسورًا.



مُستشعر الإمالة

المكوّن في المحاكي

الرمز التخطيطي

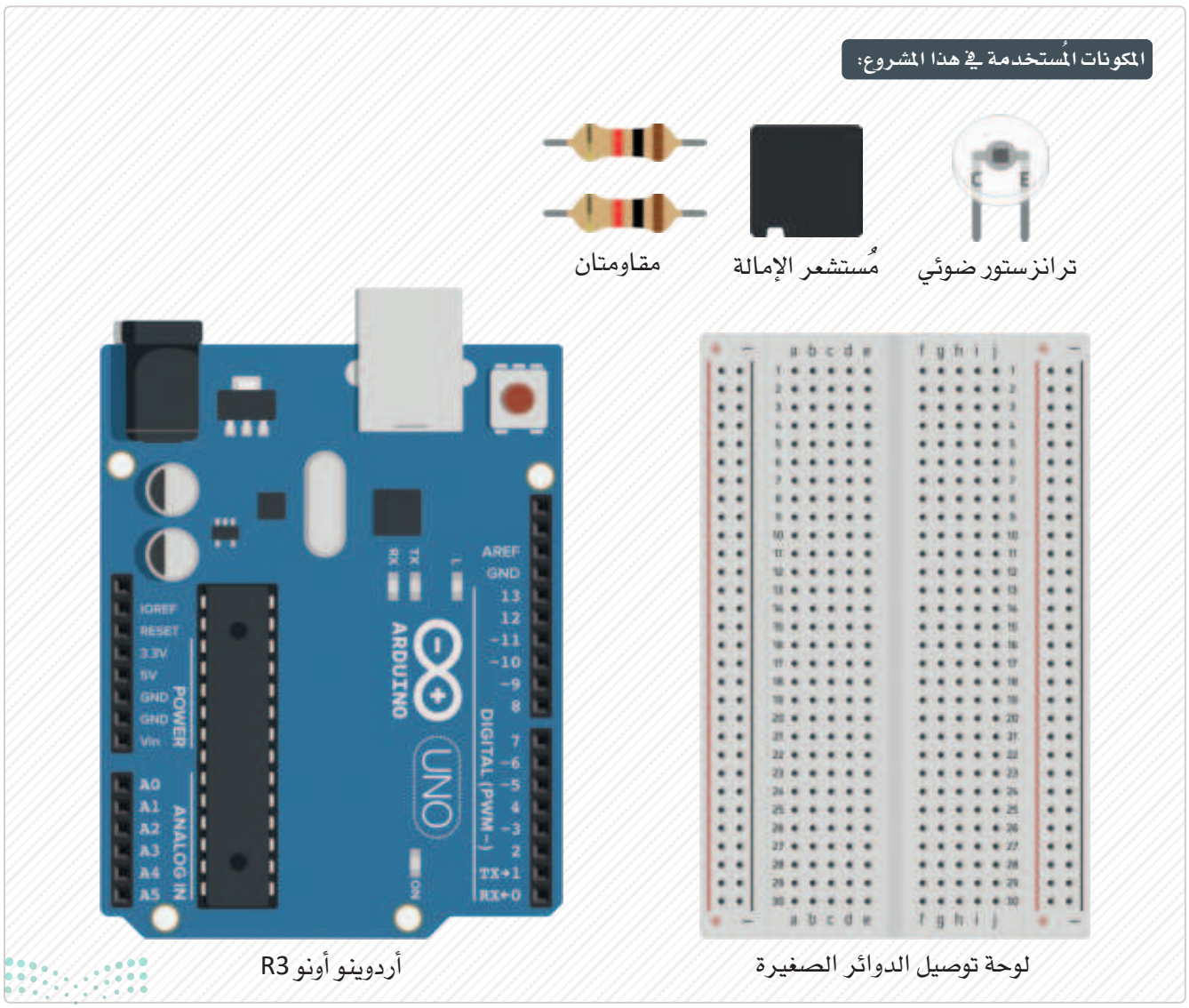
شكل 7.10: مُستشعر الإمالة

النموذج الأولي باستخدام الأردوينو Arduino Prototype

سيراقب جهاز تحكم الأردوينو الدقيق حالة الحاوية، ويجمع بيانات الأحداث المختلفة ويُرسلها من خلال بروتوكول (Firmata). سيُستخدم مُستشعر الإمالة لتسجيل استخدام الحاوية في كل مرة، ومحاكاة حركة غطاء الحاوية، وسيعمل الترانزستور الضوئي كمُستشعر عند الوصول إلى حد مُعين مما يعني أن الحاوية مليئة بالنفايات.

ستحتاج إلى المكونات الآتية:

- **لوحة أردوينو أونو R3** (Arduino Uno R3).
- **لوحة توصيل الدوائر الصغيرة** (Breadboard Small).
- **ترانزستور ضوئي** (Phototransistor).
- **مُستشعر إمالة** (Tilt Sensor).
- **مقاومتان 1 كيلو أوم** (Two resistors 1kΩ).



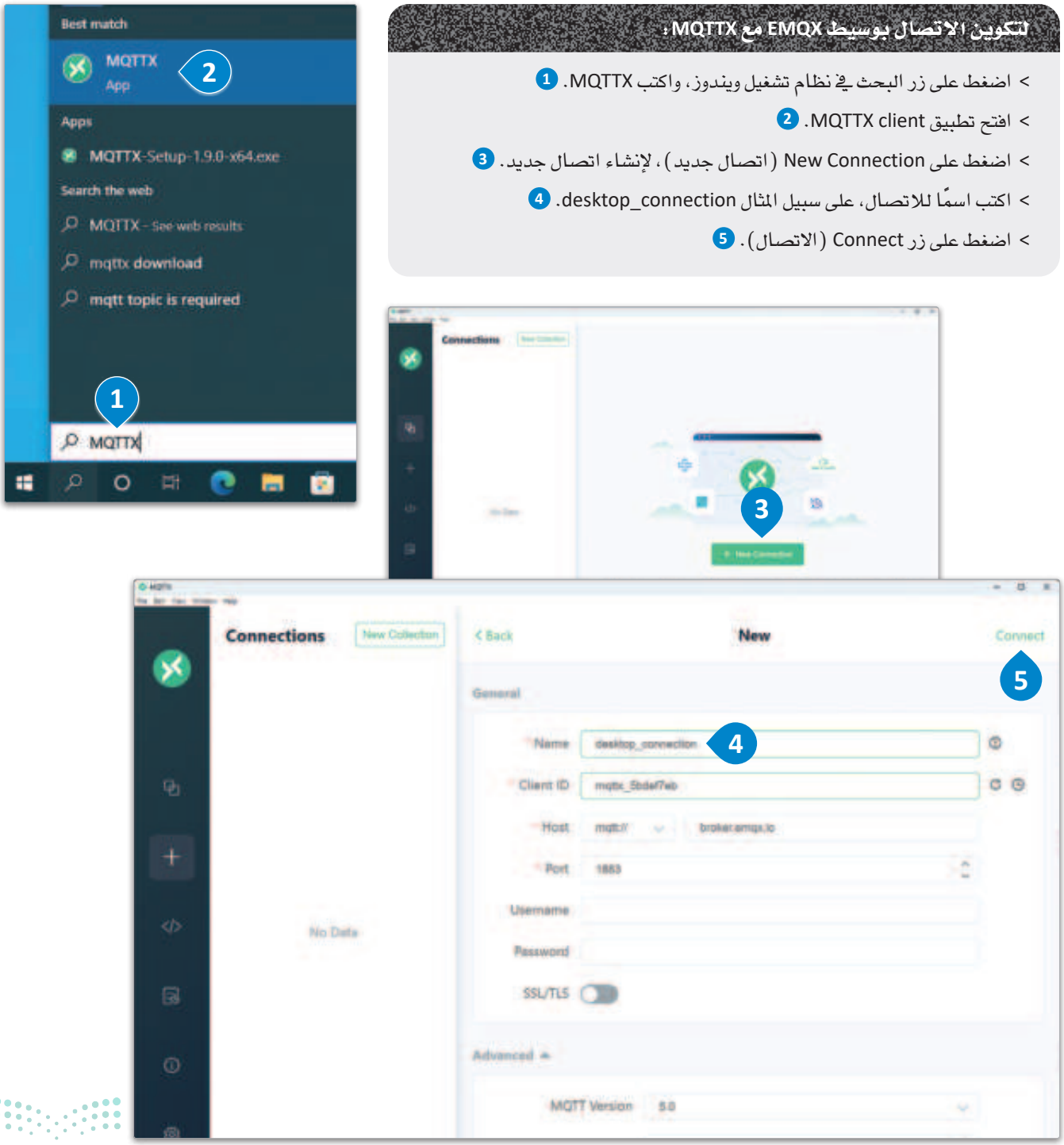
شكل 7.11: مكونات مشروع النفايات الذكية

الاتصال بوسيط EMQX العام Connecting to the EMQX Public Broker

ستحتاج أولًا إلى تثبيت التطبيق المكتبي MQTTX client، ثم اختبار الاتصال مع وسيط (EMQX) العام. لتثبيت تطبيق MQTTX Agent، قم بزيارة موقع الويب: https://mqttx.app/ وقم بتنزيل أحدث إصدار. قم بتشغيل المُثبِّت لإكمال عملية التثبيت. ستقوم الآن بفتح البرنامج العميل، وإنشاء اتصال جديد مع وسيط (EMQX).

لتكوين الاتصال بوسيط EMQX مع MQTTX:

- اضغط على زر البحث في نظام تشغيل ويندوز، واكتب MQTTX. 1
- افتح تطبيق MQTTX client. 2
- اضغط على New Connection (اتصال جديد)، لإنشاء اتصال جديد. 3
- اكتب اسمًا للاتصال، على سبيل المثال desktop_connection. 4
- اضغط على زر Connect (الاتصال). 5



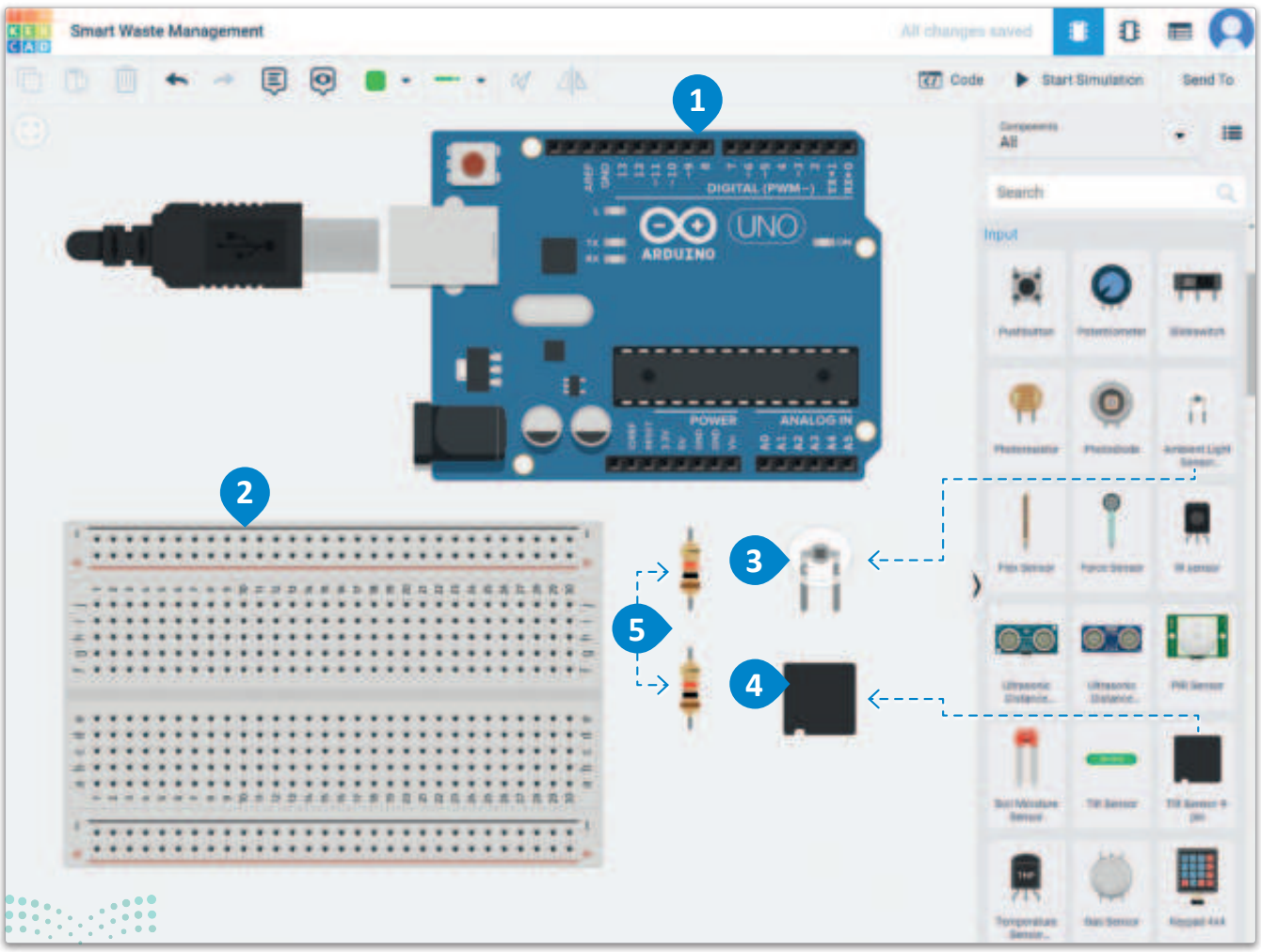
شكل 7.12: تكوين الاتصال بوسيط EMQX مع MQTTX

دائرة الأردوينو Arduino Circuit

ستبدأ في إنشاء دائرة الأردوينو عن طريق إضافة المكونات المطلوبة داخل مساحة عمل دوائر تنكركاد.

لتثبيت المكونات:

- ابحث عن Arduino Uno R3 (لوحة أردوينو أونوR3) من مكتبة Components (المُكونات)، واسحبها وأفلتها في مساحة العمل. 1
- ابحث عن Breadboard Small (لوحة توصيل الدوائر الصغيرة) من مكتبة Components (المُكونات)، واسحبها وأفلتها في مساحة العمل. 2
- ابحث عن Phototransistor (مُستشعر الإضاءة [الترانزستور الضوئي])، من مكتبة Components (المُكونات) واسحبه وأفلته في مساحة العمل. 3
- ابحث عن Tilt Sensor 4-pin (مُستشعر الإمالة بأربعة أطراف) من مكتبة Components (المُكونات)، واسحبه وأفلته في مساحة العمل. 4
- ابحث عن Resistor (المقاومة) من مكتبة Components (المكونات) واسحب اثنتين منها وأفلتهما في مساحة العمل. 5



شكل 7.13: تثبيت مكونات الدائرة

لتوصيل الترانزستور الضوئي:

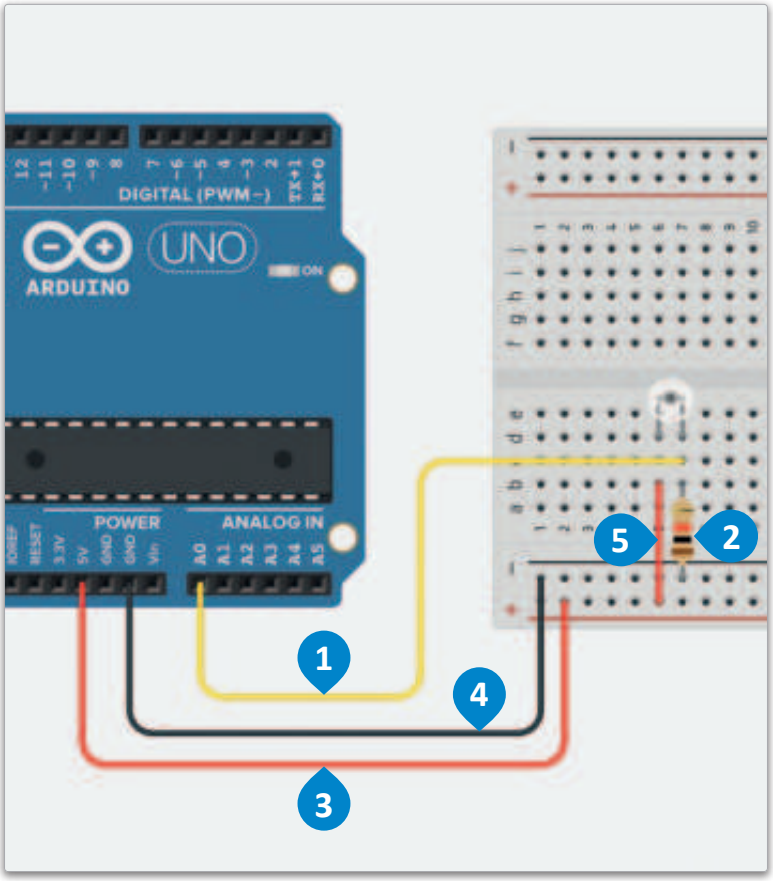
> قم بتوصيل طرف Emitter (الباعث) الخاص بالترانزستور الضوئي بالطرف التناظري A0 في الأردوينو، وغيِّر لون السلك إلى yellow (الأصفر). 1

> قم بتوصيل الطرف الثاني للمقاومة الواحدة مع الصف نفسه الذي تم توصيل باعث الترانزستور الضوئي به، ثم وصّل الطرف الأول من المقاومة بالعمود السالب من لوحة الدوائر الصغيرة. 2

> قم بتوصيل طرف 5v (5 فولت) للوحة الأردوينو أونو R3 بالعمود الموجب من لوحة التجارب، وغيِّر لون السلك إلى Red (الأحمر). 3

> قم بتوصيل GND (الطرف الأرضي) للوحة الأردوينو أونو بالعمود السالب للوحة التجارب، وغيِّر لون السلك إلى Black (الأسود). 4

> قم بتوصيل طرف المُجمع الخاص بالترانزستور الضوئي بالعمود الموجب في لوحة التوصيل المُصغرة. 5



شكل 7.14: توصيل الترانزستور الضوئي

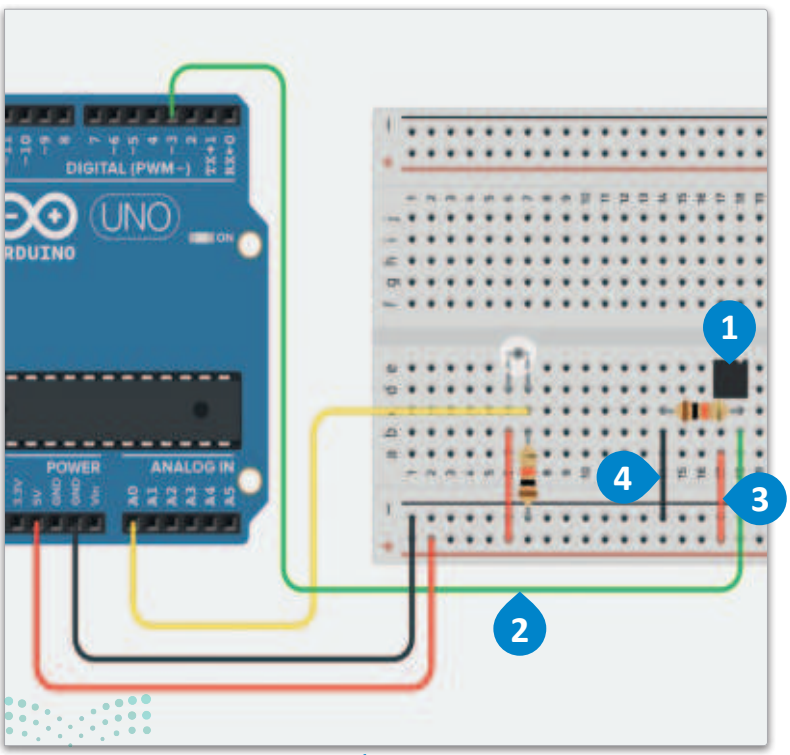
لتوصيل مُستشعر الإمالة:

> قم بتوصيل الطرف الثاني للمقاومة الأخرى بالطرف الثاني لمُستشعر الإمالة. 1

> قم بتوصيل الطرف الثاني من مُستشعر الإمالة بالطرف الرقمي 3 للوحة الأردوينو، وغيِّر لون السلك إلى Green (الأخضر). 2

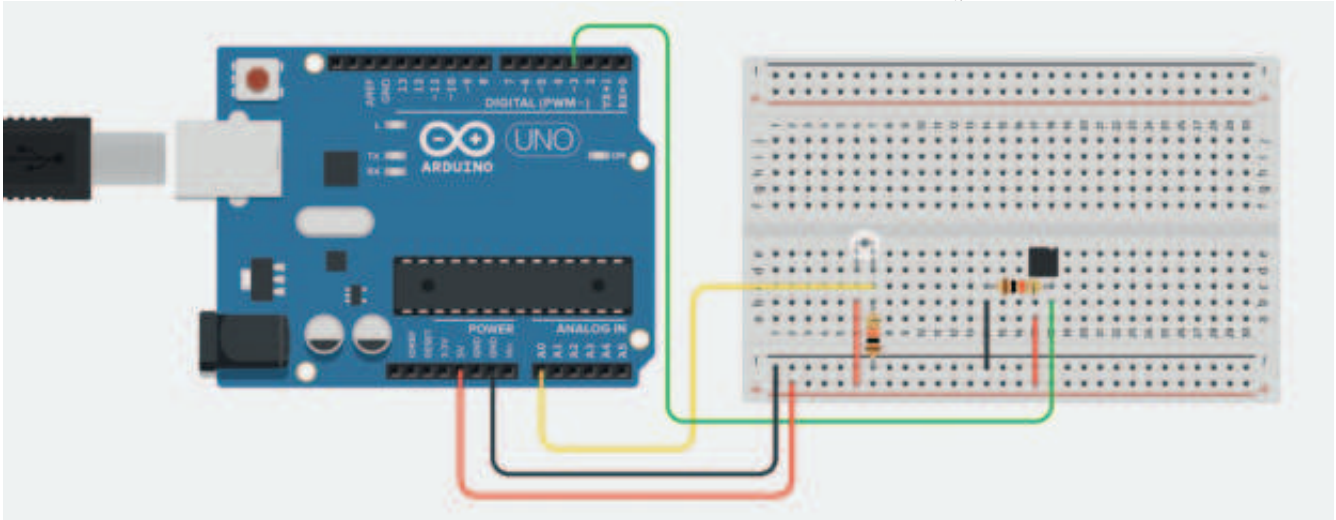
> قم بتوصيل الطرف الرابع من مُستشعر الإمالة بالعمود الموجب من لوحة الدوائر الصغيرة وغيِّر لون السلك إلى Red (الأحمر). 3

> قم بتوصيل الطرف الأول من المقاومة بالعمود السالب من لوحة الدوائر الصغيرة وغيِّر لون السلك إلى Black (الأسود). 4



شكل 7.15: توصيل مُستشعر الإمالة

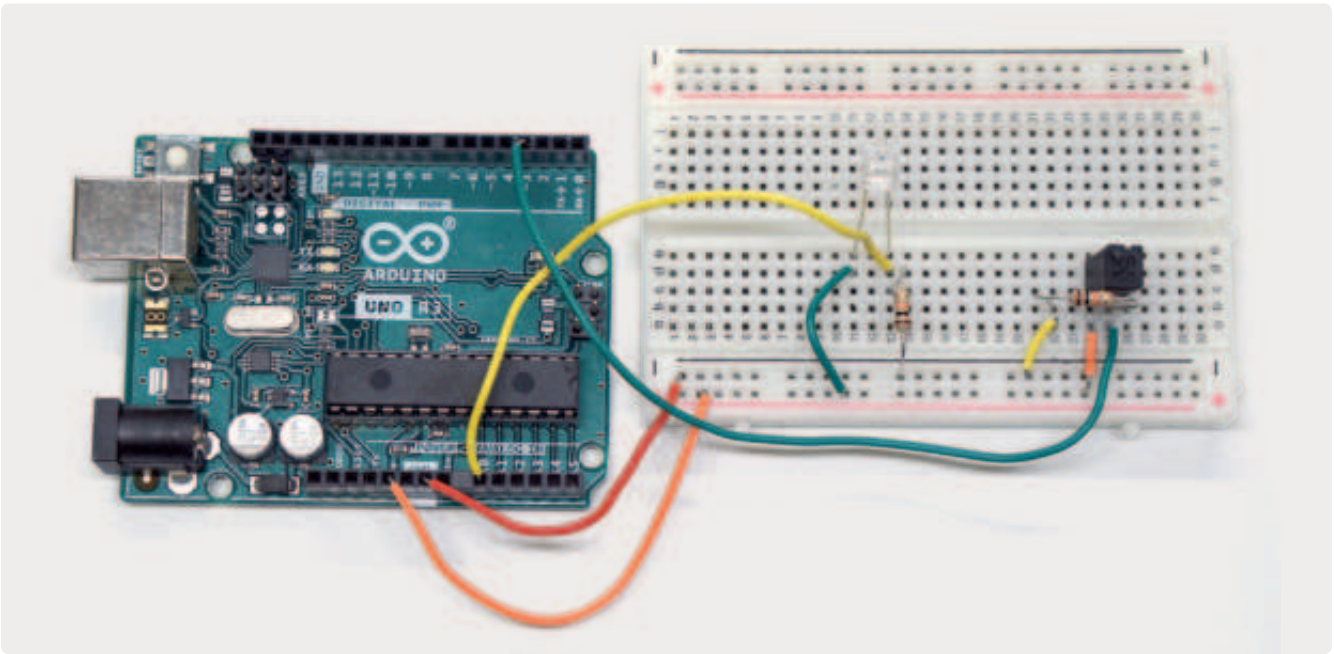
الدائرة بشكلها النهائي Complete Circuit



شكل 7.16: الدائرة بصورتها النهائية في تنكركاد

صورة الدائرة الفعلية Physical Circuit

تمثِّل هذه الصورة الشكل الذي ستبدو عليه الدائرة.



شكل 7.17: صورة للدائرة

تتصل المكونات بالأطراف الآتية:

شكل 7.18: توصيل الأطراف بالمكونات

برمجة الأردوينو Programming the Arduino

ستبدأ بتحميل بروتوكول StandardFirmata من خلال بيئة عمل Arduino IDE لإعداد قناة اتصال بين الأردوينو والبرنامج الذي ستقوم بكتابته بلغة البرمجة **البايثون** (Python).

افتح PyCharm وقم بتثبيت حِزمة paho-mqtt من خلال نظام إدارة الحِزم (pip).

في PyCharm، افتح الواجهة الطرفية (Terminal) في مُجلد عملك، واكتب الأمر الآتي:

```
pip install paho-mqtt
```

قم بإنشاء ملف بايثون جديد باسم mqtt_arduino.py، ثم في بداية مقطعك البرمجي، ثم قم باستيراد الحزم الآتية:

- **datetime**: إنشاء طوابع زمنية للرسائل التي نرسلها.
- **time**: التحكم في سير البرنامج.
- **json**: العمل مع كائنات JSON.
- **pyfirmata**: التواصل مع لوحة الأردوينو من خلال بروتوكول Firmata.
- **paho.mqtt.client**: إنشاء عملاء للتواصل مع وسطاء MQTT.

```
from datetime import datetime
import time
import json
import pyfirmata
import paho.mqtt.client as mq
```

أنشئ المتغيرات الآتية التي ستُستخدم لعميل MQTT الذي ستنشئه. سيكون اسم العميل CLIENT_ID. أما MQTT_BROKER فهو عنوان الوسيط العام الذي توفره EMQX الذي ستتصل به. ويشير TOPIC إلى اسم الموضوع الذي سيشترك فيه العميل. يشير PORT إلى منفذ الخادم الافتراضي للاتصال بالوسيط. وختامًا فإن FLAG_CONNECTED سيُستخدم كمتغير إشارة في دالة معالج الأحداث لاحقًا.

```
# Variables to setup MQTT client
CLIENT_ID = "PUBLISHER_01"      # ID of the client
MQTT_BROKER = "broker.emqx.io"  # Address of the broker
TOPIC = "waste/drops"           # Topic to subscribe to
PORT = 1883                     # Default server port
FLAG_CONNECTED = False          # Connection flag
```

جدول 3.3: متغيرات الاتصال بوسيط MQTT

المُتغير	الوصف
CLIENT___ID	اسم عميل MQTT.
MQTT___BROKER	عنوان وسيط MQTT المستهدف.
TOPIC	الموضوع الذي سيشترك فيه العميل.
PORT	منفذ الخادم المراد الاتصال به.
FLAG___CONNECTED	مُتغير إشارة للتحقق من اتصال الخادم.

أضف الأسطر الآتية، والتي مهمتها تهيئة الاتصال بالأردوينو باستخدام بروتوكول (Firmata) وكذلك تكوين الأطراف الخاصة بمُستشعر الإضاءة ومُستشعر الإمالة المُستخدمين للحصول على البيانات.

```
board = pyfirmata.Arduino('COM4')    # Specify communication port
it = pyfirmata.util.Iterator(board)  # Select the board to connect
it.start()                           # Connect to board

# Selecting the sensor pins
light_sensor_pin = board.get_pin('a:0:i')
tilt_sensor_pin = board.get_pin('d:3:i')
```

أنشئ المتغيرات الآتية بالأسماء الآتية: can_full وهو وَسْمٌ يُحدد ما إذا كانت حاوية القمامة قد مُلئت أم لا، و garbage_drops وهو عدَّاد لتتبع عدد مرات الاستخدام لتعبئة الحاوية بالكامل.

```
can_full = False    # Flag to indicate whether the can is full
garbage_drops = 0   # Counter for the garbage drops
```

قم بإنشاء الدالة الآتية التي تعيد تعيين مُتغيري can_full و garbage_drops في كل مرة تكون فيها الحاوية ممتلئة، وتُرسل رسالة إلى العميل حول هذا الموضوع.

```
def reset_can():
    global garbage_drops  # Access the garbage_drops variable
    global can_full       # Access the can_full variable
    garbage_drops = 0     # Reset the counter to 0
    can_full = False      # Clear the can
```

قم بإنشاء الدالة الآتية لإرسال رسالة للعميل تفيد بأنه تم استخدام الحاوية. ستقوم أولًا بإنشاء متغير باسم timestamp لتسجيل الوقت، وإنشاء كائن قاموس Dictionary بالخصائص timestamp و garbage_drops و can_full. ستقوم بتحويل هذا القاموس إلى كائن JSON، ثم نشره إلى موضوع المُشترك "waste/drops" من خلال العميل.

```
def publish_message():
    global garbage_drops  # Access garbage_drops variable
    global can_full       # Access can_full variable
    # Create a custom format for the timestamp
    timestamp = str(datetime.now().strftime("%H:%M:%S"))
    msg_dictionary = {    # Creating the JSON object
        "timestamp": timestamp,
        "garbage_drops": garbage_drops,
        "can_full": can_full
    }
    msg = json.dumps(msg_dictionary)  # Convert dictionary to JSON
    try:
        result = client.publish(TOPIC, msg)  # Publish message
    except:
        print("There was an error while publishing the message")
    time.sleep(2)
```

إنشاء تنسيق يتناسب مع بيانات الوقت.

إنشاء كائن قاموس JSON لإرسال البيانات.

قم بإنشاء دالة معالج الأحداث الآتية، والتي ستطبع رسالة تأكيد إلى الواجهة الطرفية Terminal حول نجاح الاتصال بالعميل أو فشله. وسيطات الدالة هي وسيطات افتراضية تُستخدم لربط هذه الدالة بمعالج الأحداث المناسب الذي توفره مكتبة paho.mqtt.client.

```
def on_connect(client, userdata, flags, rc):
    global FLAG_CONNECTED  # Access the FLAG_CONNECTED variable

    if rc == 0:            # If rc is 0 the client connected successfully
        FLAG_CONNECTED = True
        print("Connected to MQTT Broker!")
    else:
        print("Failed to connect to MQTT Broker!")
```

هذا المتغير يُرسل من خلال مكتبة paho ليظهر حالة الاتصال.

ستقوم في الجزء الرئيس من البرنامج بتهيئة عميل MQTT، وربط معالج الأحداث on_connect بالدالة المذكورة أعلاه، ثم الاتصال بوسيط MQTT المُحدد، والاشتراك في الموضوع المحدد.

```
client = mq.Client(CLIENT_ID)     # Initialize an MQTT client
client.on_connect = on_connect    # Bind the on_connect event handler
client.connect(MQTT_BROKER, PORT) # Connect to the specified MQTT broker
client.subscribe(TOPIC, 0)        # Subscribe to the specified topic
```

قم بإنشاء التكرار الرئيس للبرنامج. إذا كانت قيمة light_value أقل من 0.200، فستُعدّ الحاوية ممتلئة.

```
while True:
    # Get sensor values
    light_value = light_sensor_pin.read()
    tilt_value = tilt_sensor_pin.read()

    if (light_value is not None) and (tilt_value is not None):
        print("Light levels : " + str(light_value))
        print("Tilt levels : " + str(tilt_value))
        print("Garbage drops : " + str(garbage_drops))

        # If there is a tilt, add 1 to the counter
        if (tilt_value == True):
            garbage_drops += 1
            # If there is a tilt and the can is full
            # publish a message and reset the can
            if (light_value <= 0.200):
                can_full = True
                publish_message()
                reset_can()
            publish_message()
        time.sleep(1)
```

سيُبدأ في نشر البيانات عندما تمتلئ حاوية القمامة التي يُكشف عنها في ظروف الإضاءة المنخفضة.

البرنامج بشكله النهائي Complete Code

```
from datetime import datetime
import time
import json
import pyfirmata
import paho.mqtt.client as mq

# Variables to setup MQTT client
CLIENT_ID = "PUBLISHER_01"      # ID of the client
MQTT_BROKER = "broker.emqx.io"  # Address of the broker
TOPIC = "waste/drops"           # Topic to subscribe to
PORT = 1883                     # Default server port
FLAG_CONNECTED = False          # Connection flag

board = pyfirmata.Arduino('COM4')   # Specify communication port
it = pyfirmata.util.Iterator(board) # Select the board to connect
it.start()                          # Connect to board

# Selecting the sensor pins
light_sensor_pin = board.get_pin('a:0:i')
tilt_sensor_pin = board.get_pin('d:3:i')

can_full = False    # Flag to indicate whether the can is full
garbage_drops = 0   # Counter for the garbage drops

def reset_can():
    global garbage_drops  # Access garbage_drops variable
    global can_full       # Access can_full variable
    garbage_drops = 0     # Reset the counter to 0
    can_full = False      # Clear the can
```

```
def publish_message():
    global garbage_drops   # Access garbage_drops variable
    global can_full        # Access can_full variable

    # Create a custom format for the timestamp
    timestamp = str(datetime.now().strftime("%Y-%m-%d %H:%M:%S"))

    # Creating the dictionary object
    msg_dictionary = {
        "timestamp": timestamp,
        "garbage_drops": garbage_drops,
        "can_full": can_full
    }
    msg = json.dumps(msg_dictionary)  # Convert dictionary to JSON

    try:
        result = client.publish(TOPIC, msg)  # Publish message
    except:
        print("There was an error while publishing the message")

    time.sleep(2)
    print("Message sent to the MQTT broker")

def on_connect(client, userdata, flags, rc):
    global FLAG_CONNECTED  # Access the FLAG_CONNECTED variable

    if rc == 0:            # If rc is 0 the client connected successfully
        FLAG_CONNECTED = True
        print("Connected to MQTT Broker!")
    else:
        print("Failed to connect to MQTT Broker!")
```

```
client = mq.Client(CLIENT_ID)     # Initialize an MQTT client
client.on_connect = on_connect    # Bind the on_connect event handler
client.connect(MQTT_BROKER, PORT) # Connect to the specified MQTT broker
client.subscribe(TOPIC, 0)        # Subscribe to the specified topic

while True:
    # Get sensor values
    light_value = light_sensor_pin.read()
    tilt_value = tilt_sensor_pin.read()

    if (light_value is not None) and (tilt_value is not None):
        print("Light levels : " + str(light_value))
        print("Tilt levels : " + str(tilt_value))
        print("Garbage drops : " + str(garbage_drops))

        # If there is a tilt, add 1 to the counter
        if (tilt_value == True):
            garbage_drops += 1

            # If there is a tilt and the can is filled
            # publish a message and reset the can
            if (light_value <= 0.200):
                can_full = True
                publish_message()
                reset_can()

            publish_message()

        time.sleep(1)
```

اختبار الوسيط Testing the Broker

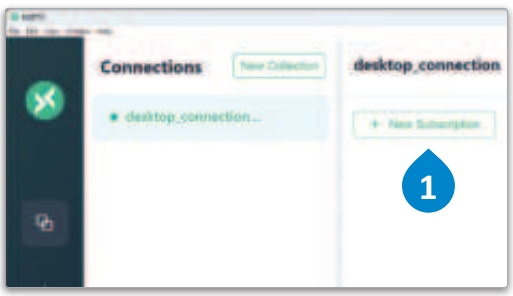
تُعدُّ EMQX وسيط MQTT عام لاختبار وتطوير تطبيقاتMQTT. تساعد EMQX في تطوير النماذج الأولية لتطبيقات إنترنت الأشياء دون الحاجة إلى وضع البنى التحتية وتطوير الوسيط. ستستخدِم عميل MQTTX client للتحقق من نشر الرسائل، ثم ستُنشئ مقطعًا برمجيًا آخر في البايثون يستقبل الرسائل المنشورة، ويُنشئ تقاريرَ عن الحاوية، ويحلِّل البيانات الموجودة في تلك التقارير. بعد تحميل مخطط StandardFirmata إلى الأردوينو، نفِّذ تعليمات البايثون وحرِّك لوحة التجارب لتفعيل مُستشعر الإمالة. في كل مرة يُفعّل بها المُستشعر، ستُزاد عداد النفايات والذي يشير إلى عدد المرات التي تم بها فتح الحاوية افتراضيًا لوضع القمامة. وعند تفعيل مُستشعر الإمالة مع تغطية مُستشعر الإضاءة، سينشر البرنامج رسالة مفادها أن الحاوية مُمتلئة، ويعيد ضبط عداد النفايات. ستقوم في الدرس التالي بتحليل البيانات بناءً على الرسائل المنشورة. لاختبار نشر رسائلك بصورة صحيحة، ستستخدِم العميل المكتبي MQTTX Agent قبل تنفيذ مقاطع البايثون البرمجية، ستستخدِم عميل MQTTX للاشتراك في موضوع "waste/drops". سينتظر العميل الآن استلام الرسائل التي تُنشر من خلال برنامج البايثون وتُوزّع من خلال وسيط EMQX العام.

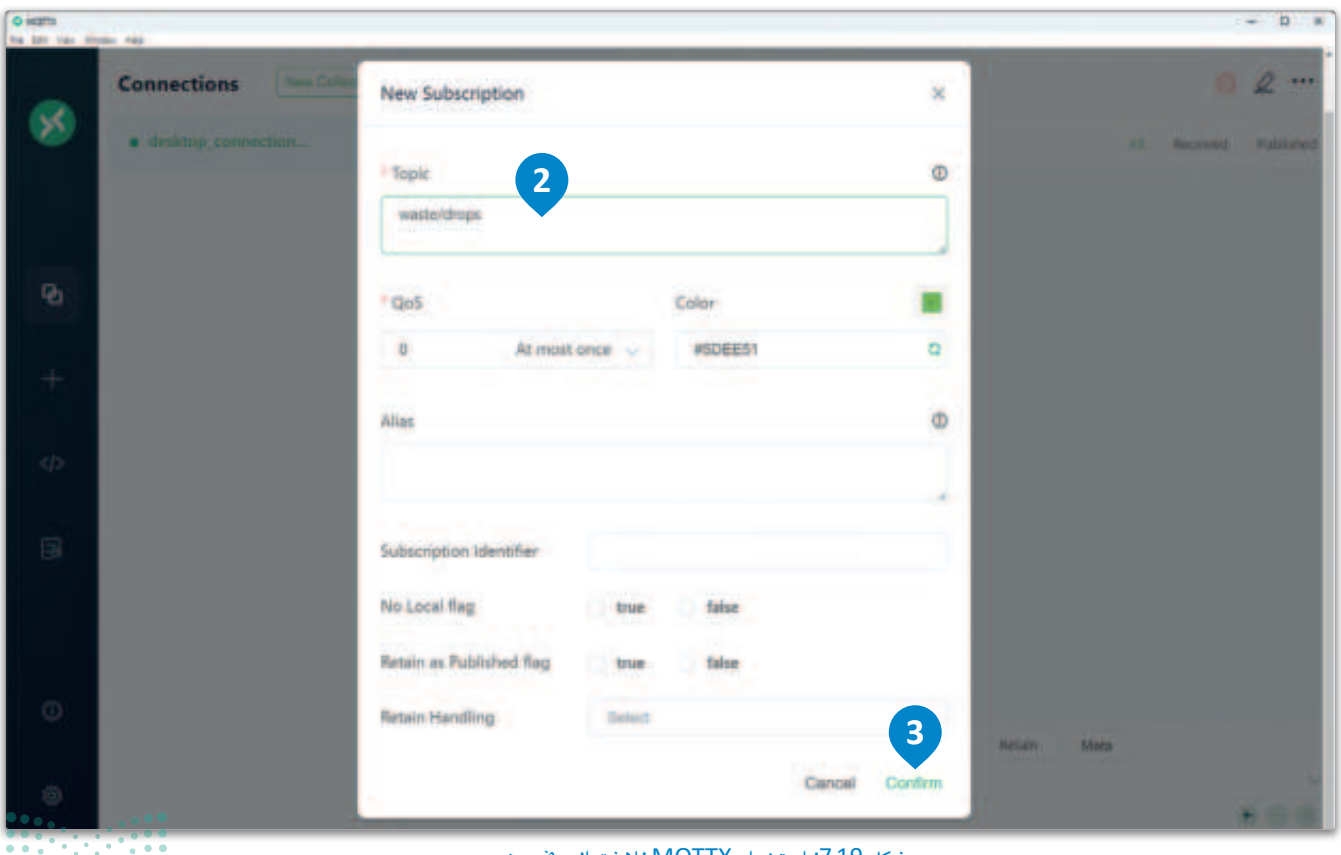
لاستخدام MQTTX للاشتراك في موضوع المحدد:

> في قائمة Connections (اتصالات) في تبويبة desktop_connection (سطح المكتب)، اضغط على زر New Subscription (اشتراك جديد). 1

> في مربع نص Topic (الموضوع)، اكتب "waste/drops". 2

> اضغط على زر Confirm (تأكيد). 3



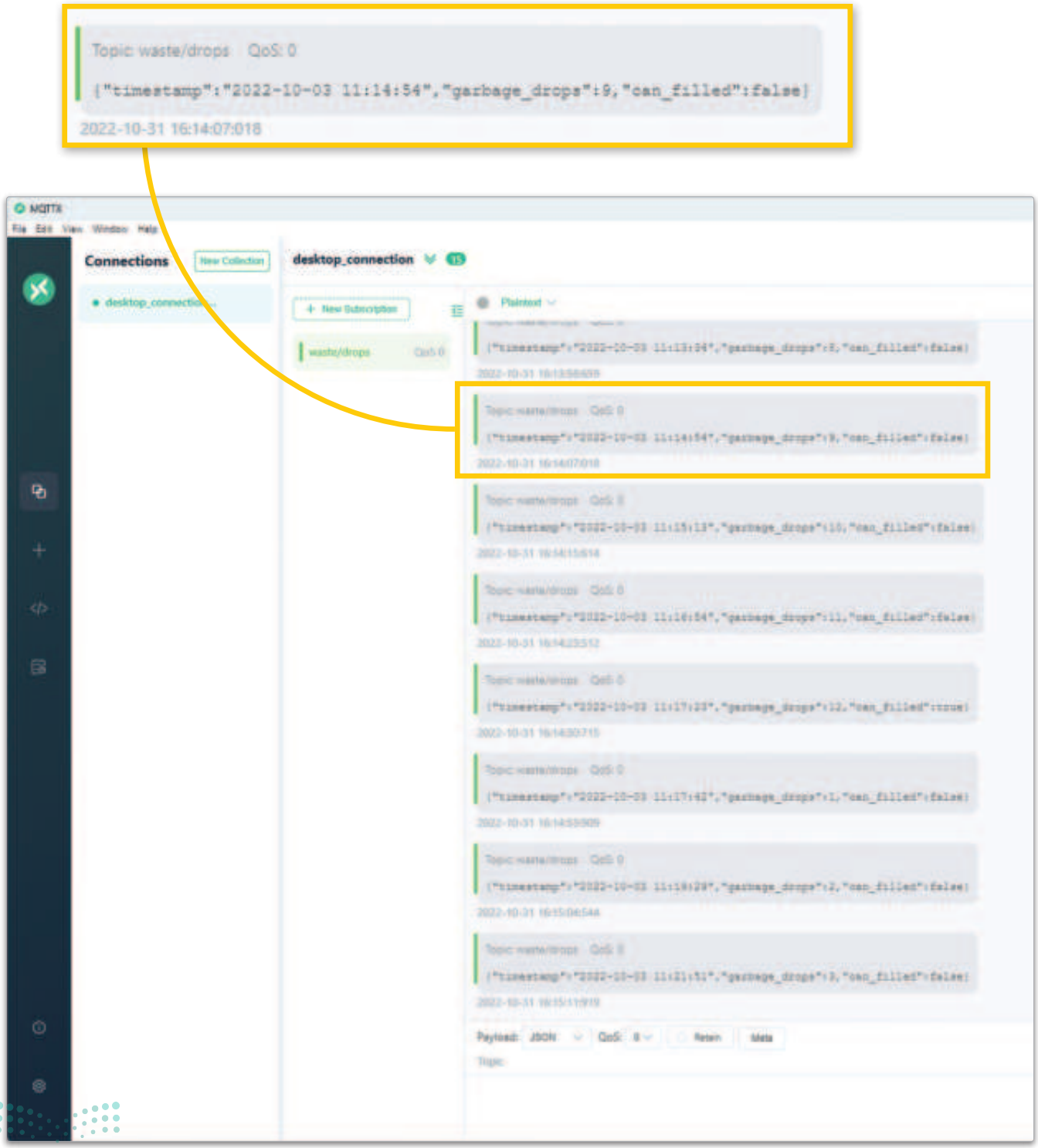


شكل 7.19: استخدام MQTTX للاشتراك في موضوع محدد

عرض الرسائل من خلال عميل MQTTX

Viewing Messages through the MQTTX Client

بعد تنفيذ المقاطع البرمجية في البايثون، وبدء نشر الرسائل، سيتم استقبال الرسائل من خلال العميل المكتبي MQTTX كما يظهر أدناه.



شكل 7.20: عرض الرسائل من خلال عميل MQTTX

تمرينات

1 أنشئ مُخططًا لشبكة MQTT مع لوحة أردوينو واحدة تعمل كناشر، واثنتان تعملان كمُستقبلات.

2 قدِّمْ وصفًا للترانزستور الضوئي ومكونات مُستشعر الإمالة وحالات استخدامهما.

3 قُمْ بتحليل ماهية وسيط EMQX العام ومدى مساهمته في تطوير تطبيقات النماذج الأولية لإنترنت الأشياء.

4 قُمْ بتحديث معالج الحدث لحدث on_connect الذي يطبع معلومات التكوين إلى وحدة التحكم المُقدمة إلى عميل MQTT.

5 قُمْ بتحديث الدالة ( )publish_message لنشر رسالة زمنية timestamp تعرض التاريخ والوقت بأكملهما، وخاصية جديدة تحتوي على مُعرِّف العميل.

6 أنشئ مقطعًا برمجيًا بلغة البايثون يتيح للمستخدِم كتابة الموضوع الذي يريد الاشتراك به، والرسالة التي يريد إرسالها ثم نشرها من خلال وسيط EMQX العام.

اختبر برنامجك باستخدام العميل المكتبي MQTT X.

رابط الدرس الرقمي

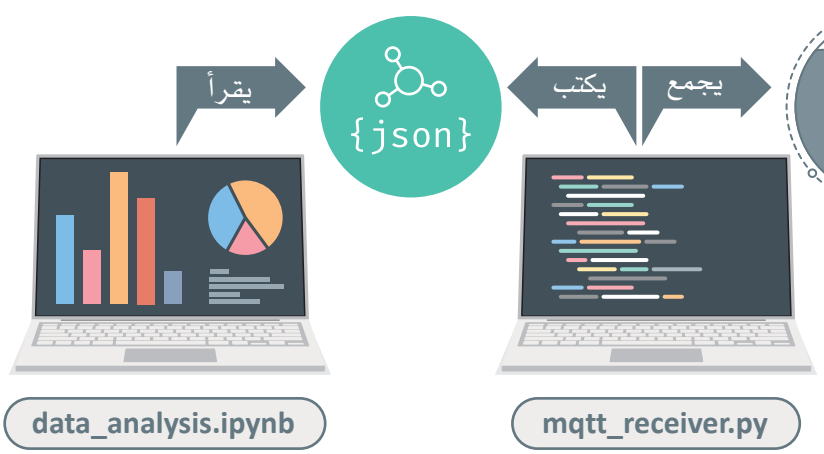
www.ien.edu.sa

الدرس الثالث

إنشاء حل ذكي لإدارة النفايات

إدارة النفايات الذكية وتحليل البيانات
Smart Waste Management and Data Analysis

لقد قُمت في الدرس السابق بإنشاء نموذج أولي ذكي لإدارة حاوية نفايات باستخدام جهاز تحكم أردوينو دقيق يراقب بيئته المحيطة، ثم يُنتج البيانات من المُستشعرات، وينشرها كرسائل إلى موضوع في MQTT. يجب أن يتم جمع البيانات ثم معالجتها لتكوين تصورات مختلفة ووضع الخطط بناءً على هذه البيانات.



شكل 7.21: تحليل بيانات مشروع النفايات الذكية

ستقوم في هذا الدرس بإنشاء مقطع برمجي في البايثون يشترك في نفس الموضوع الخاص بوسيط MQTT الذي نُشرت عليه الرسائل. ستُجمع هذه الرسائل، وفي كل مرة تمتلئ فيها سلة النفايات، سيُنشأ تقرير ويُخزّن في ملف آخر. سيحتوي هذا الملف فقط على البيانات من التقارير التي أُنشئت، ثم ستُنشئ بعد ذلك مفكرة جوبيتر تقوم بفتح الملف وبتحليل البيانات وإنشاء رؤى وتصورات حولها.

سيُسمّى هذان الملفان باسم mqtt_receiver.py وdata_analysis.ipynb على التوالي، سيكتب البرنامج الأول البيانات المحفوظة في ملف JSON، وسيقرأ البرنامج الثاني البيانات من ملف JSON ويقوم بتحليلها.

إنشاء ملف بيانات جسون Creating the JSON Data File

ستقوم بإنشاء ملف بيانات JSON بواسطة مصفوفة فارغة، وسيقوم البرنامج mqtt_receiver.py بإلحاق كل تقرير أُنشئ ككائن JSON بالمصفوفة، ثم سيفتح ملف data_analysis.py ملف JSON هذا، وسيقرأ محتويات مصفوفة JSON السابقة ويقوم بعمليات تحليل البيانات.

افتح PyCharm ثم أنشئ ملفًا جديدًا في مجلدك الخاص باسم data.json، ثم أنشئ كائنًا لمصفوفة فارغة داخل هذا الملف كما هو موضح أدناه. سيُلحق mqtt_receiver.py كائنات JSON الخاصة بالتقارير المُنشأة بالمصفوفة المعروضة أدناه. احفظ الملف data.json ثم أغلقه.

```
[]
```

قم بإنشاء ملف بايثون جديد باسم mqtt_receiver.py، وفي بداية المقطع البرمجي، قم باستيراد الحزم الآتية:

- **datetime**: تُنشئ طابعًا زمنيًا للرسائل المُرسلة.
- **json**: للتعامل مع كائنات JSON.
- **paho.mqtt.client**: لإنشاء عملاء للتواصل مع وسطاء MQTT.
- **os**: للتعامل مع الملفات الموجودة على حاسبك الخاص.

```
from datetime import datetime
import json
import paho.mqtt.client as mq
from os import path
```

قم بإنشاء المتغيرات الآتية data_file و data_file_objects والتي ستتفاعل مع ملف بيانات JSON.

```
data_file = "your_file_path"  # Absolute path to the JSON data file
data_file_objects = []  # This contains the objects from the JSON data file
```

تأكد من إدخال امتداد ملف البيانات الصحيح.

قم بإنشاء المتغيرات الآتية التي ستُستخدم لعميل MQTT الذي ستُنشئه باسم CLIENT_ID. بينما يشير MQTT_BROKER إلى عنوان الوسيط العام الذي توفره EMQX الذي سيتم الاتصال به، و TOPIC إلى اسم الموضوع الذي سيشترك فيه العميل. و PORT إلى مَنفذ الخادم الافتراضي للاتصال بالوسيط، وFLAG_CONNECTED الذي سيُستخدم كمتغير إشارة في دالة معالج الأحداث لاحقًا.

```
# Variables to setup MQTT client
CLIENT_ID = "RECEIVER_01"       # ID of the client
MQTT_BROKER = "broker.emqx.io" # Address of the broker
TOPIC = "waste/drops"           # Topic to subscribe to
PORT = 1883                     # Default server port
FLAG_CONNECTED = False          # Connection flag
```

قم بإنشاء المتغيرات الآتية messages_stack و reports والتي ستُستخدم لتخزين المعلومات من الرسائل المنشورة.

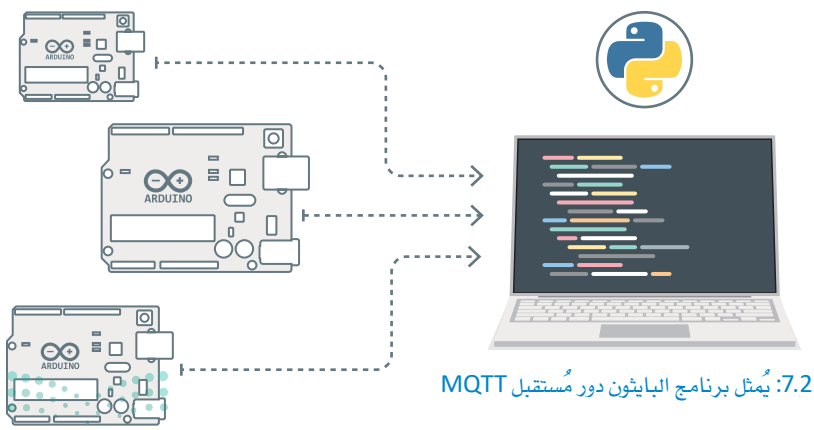
```
messages_stack = []  # The array with the messages per can filling
reports = []  # The array with all the generated report objects
```

قم بإنشاء دالة معالج الأحداث الآتية والتي تطبع رسالة تأكيد إلى الواجهة الطرفية Terminal حول نجاح الاتصال بالعميل من عدمه. وسيطات الدالة هي وسيطات افتراضية يجب استخدامها لربط هذه الدالة بمعالج الأحداث المناسب الذي توفره المكتبة paho.mqtt.client.

```
def on_connect(client, userdata, flags, rc):
    global FLAG_CONNECTED  # Access the FLAG_CONNECTED variable

    if rc == 0:            # If rc is 0 the client connected successfully
        FLAG_CONNECTED = True
        print("Connected to MQTT Broker!")
    else:
        print("Failed to connect to MQTT Broker!")
```

```
def on_message(client, userdata, msg):
    global messages_stack  # Access the messages_stack variable

    # Decode the message payload
    payload = str(msg.payload.decode())

    # Convert the payload to a JSON object and append it
    # to the messages stack
    payload_object = json.loads(payload)
    messages_stack.append(payload_object)

    # When you receive a message, print it to the terminal
    print("||---- MESSAGE RECEIVED ----||\n")
    print("Payload: " + str(payload_object))

    # If the payload object has the can_filled flag set to True
    # generate a report for the filled can
    if payload_object["can_filled"] == True:
        generate_report()
```

إنشاء بيانات التقرير فقط عند امتلاء حاوية القمامة.



شكل 7.22: يُمثل برنامج البايثون دور مُستقبل MQTT

يمكن لبرنامج البايثون هذا أن يعمل كمُستقبل MQTT لجمع رسائل من عدة ناشري أردوينو في نفس الوقت. يمكن توسعة هذا البرنامج ليتمكن من معالجة بيانات JSON مع إضافة المزيد من الحقول التي تحتوي على معلومات حول الناشرين أيضًا.

قم بإنشاء دالة generate_report الآتية والتي ستنشئ كائن JSON للتقرير وتلحقه بملف البيانات في كل مرة تُستلم فيها رسالة تشير إلى امتلاء حاوية النفايات.

```
def generate_report():
    global data_file_objects  # Access data_file_objects variable
    global messages_stack     # Access messages_stack variable
    global reports            # Access reports variable

    # Getting the first and last objects from the messages_stack
    first_msg = messages_stack[0]
    last_msg = messages_stack[len(messages_stack) - 1]

    # Converting the string attributes of the messages objects to datetime
    time_filled_timestamp = last_msg["timestamp"]
    first_timestamp = datetime.strptime(first_msg["timestamp"],"%H:%M:%S")
    last_timestamp = datetime.strptime(last_msg["timestamp"], "%H:%M:%S")

    garbage_drops = last_msg["garbage_drops"]

    # Calculating the time_to_fill by comparing the timestamps
    # of the first and last fillings
    time_delta = last_timestamp - first_timestamp
    time_to_fill = time_delta.total_seconds() / 60
    report_id = len(reports)  # This will be used for object indexing
    # The JSON object that will be appended to the JSON data file
    report = {
        "id": report_id,
        "timestamp": time_filled_timestamp,
        "garbage_drops": garbage_drops,
        "time_to_fill": time_to_fill
    }
```

ستُستخدم الرسالتان الأولى والأخيرة لحساب فرق الوقت.

ستُستخدم عملية الطابع الزمني لحساب فرق الوقت.

```
    # Append the new report to the objects of the data file
    # and write the data_file_objects array to the data file
    data_file_objects.append(report)
    with open(data_file, 'w') as file:
        json.dump(data_file_objects, file, indent=4, separators=(',',': '))

    # Append the report object to the reports array and to the JSON data file
    # and clear the messages stack
    reports.append(report)
    messages_stack = []
```

تحديث محتويات مصفوفة البيانات وملف البيانات.

ستتحقق في الجزء الرئيس من البرنامج مما إذا كانت البيانات موجودة وتقوم بفتحها، ثم ستقوم بتهيئة عميل MQTT، وتربط معالجات الأحداث on_connect و on_message بالدوال المذكورة أعلاه، ثم تقوم بالاتصال بوسيط MQTT، والاشتراك في الموضوع المحدد والاستماع إلى الرسائل الواردة.

```
#  Check if the data file exists
if path.isfile(data_file) is False:
    raise Exception("Data file not found")
# Read the contents of the JSON data file
with open(data_file) as fp:
    data_file_objects = json.load(fp)

client = mq.Client(CLIENT_ID)      # Initialize an MQTT client
client.on_connect = on_connect     # Bind the on_connect event handler
client.on_message = on_message     # Bind the on_message event handler
client.connect(MQTT_BROKER, PORT)  # Connect on the specified MQTT broker
client.subscribe(TOPIC, 0)         # Subscribe to the specified topic
client.loop_forever()              # Listen continuously for messages
```

التأكد من وجود ملف البيانات لتجنب الأخطاء.

البرنامج بشكله النهائي Complete Code

```
from datetime import datetime
import json
import paho.mqtt.client as mq
from os import path

data_file = "your_file_path"  # Absolute path to the JSON data file
data_file_objects = []  # This contains the objects from the JSON data file

# Variables to setup MQTT client
CLIENT_ID = "RECEIVER_01"      # ID of the client
MQTT_BROKER = "broker.emqx.io" # Address of the broker
TOPIC = "waste/drops"          # Topic to subscribe to
PORT = 1883                    # Default server port
FLAG_CONNECTED = False         # Connection flag

messages_stack = []  # The array with the messages per can filling
reports = []  # The array with all the generated report objects

def on_connect(client, userdata, flags, rc):
    global FLAG_CONNECTED  # Access the FLAG_CONNECTED variable

    if rc == 0:            # If rc is 0 the client connected successfully
        FLAG_CONNECTED = True
        print("Connected to MQTT Broker!")
    else:
        print("Failed to connect to MQTT Broker!")

def on_message(client, userdata, msg):
    global messages_stack  # Access the messages_stack variable
```

```
    # Decode the message payload
    payload = str(msg.payload.decode())

    # Convert the payload to a JSON object and append it
    # to the messages stack
    payload_object = json.loads(payload)
    messages_stack.append(payload_object)

    # When you receive a message, print it to the terminal
    print("||---- MESSAGE RECEIVED ----||\n")
    print("Payload: " + str(payload_object))

    # If the payload object has the can_filled flag set to True
    # generate a report for the filled can
    if payload_object["can_filled"] == True:
        generate_report()
def generate_report():
    global data_file_objects  # Access data_file_objects variable
    global messages_stack     # Access messages_stack variable
    global reports            # Access reports variable

    # Getting the first and last objects from the messages_stack
    first_msg = messages_stack[0]
    last_msg = messages_stack[len(messages_stack) - 1]

    # Converting the string datetimes to datetime objects
    first_timestamp = datetime.strptime(first_msg["timestamp"], "%H:%M:%S")
    last_timestamp = datetime.strptime(last_msg["timestamp"], "%H:%M:%S")
    garbage_drops = last_msg["garbage_drops"]
    # Calculating the time_to_fill by comparing the timestamps
    # of the first and last fillings
    time_delta = last_timestamp - first_timestamp
```

```
    time_to_fill = time_delta.total_seconds() / 60
    report_id = len(reports)  # This will be used for object indexing

    # The JSON object that will be appended to the JSON data file
    report = {
        "id": report_id,
        "garbage_drops": garbage_drops,
        "time_to_fill": time_to_fill
    }
    # Append the new report to the objects of the data file
    # and write the data_file_objects array to the data file
    data_file_objects.append(report)
    with open(data_file, 'w') as file:
        json.dump(data_file_objects, file, indent=4, separators=(',',': '))

    # Append the report object to the reports array and to the JSON data file
    # and clear the messages stack
    reports.append(report)
    messages_stack = []
#  Check if the data file exists
if path.isfile(data_file) is False:
    raise Exception("Data file not found")
# Read the contents of the JSON data file
with open(data_file) as fp:
    data_file_objects = json.load(fp)

client = mq.Client(CLIENT_ID)      # Initialize an MQTT client
client.on_connect = on_connect     # Bind the on_connect event handler
client.on_message = on_message     # Bind the on_message event handler
client.connect(MQTT_BROKER, PORT)  # Connect on the specified MQTT broker
client.subscribe(TOPIC, 0)         # Subscribe to the specified topic
client.loop_forever()              # Listen continuously for messages
```

تحليل البيانات في جوبيتر Data Analysis in Jupyter Notebook

ستستخدِم الآن مُفكرة جوبيتر لإجراء عمليات تحليل البيانات على ملف بيانات JSON. ونظرًا لأن جمع البيانات اللازمة وتحليلها يستغرق وقتًا طويلًا، فقد تم توفير مجموعة بيانات JSON جاهزة لكي تستخدمها. تحاكي مجموعة البيانات هذه ترك نموذج الأردوينو الأولي قيد التشغيل لفترة طويلة من الوقت.

ملف JSON متاح للتنزيل من هنا:

http://binary-academy.com/dnld/KSA/IOT2/U3_L3_DATA.json

ستقوم أولًا باستيراد المكتبات المطلوبة وقراءة بيانات JSON من الملف.

```
import os
import pandas as pd                    # library used for data manipulation
import matplotlib.pyplot as plt        # library used for plotting data

# The data that will be used, extracted from the JSON dataset
data = pd.read_json('U3_L3_DATA.json','records',convert_dates=['timestamp'])
```

ستقوم بعد ذلك بوصف مجموعة البيانات لاستخراج الخصائص الإحصائية.

```
data.describe().round(0)
```

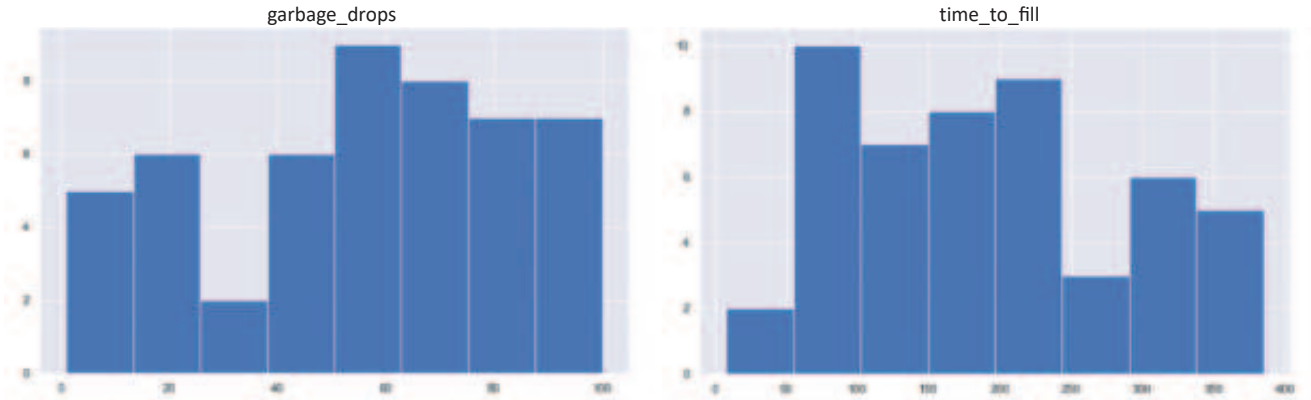
هذا هو ملف البيانات الذي تم تنزيله والذي يجب وضعه في نفس مجلد المشروع.

	id	garbage_drops	time_to_fill
count	50.0	50.0	50.0
mean	24.0	54.0	152.0
std	15.0	30.0	100.0
min	0.0	2.0	5.0
25%	12.0	30.0	60.0
50%	24.0	55.0	147.0
75%	37.0	78.0	235.0
max	49.0	100.0	376.0

شكل 7.23: وصف البيانات

ستقوم بإنشاء مُخططين بيانيين مُجمّعين حسب خصائص garbage_drops و time_to_fill.

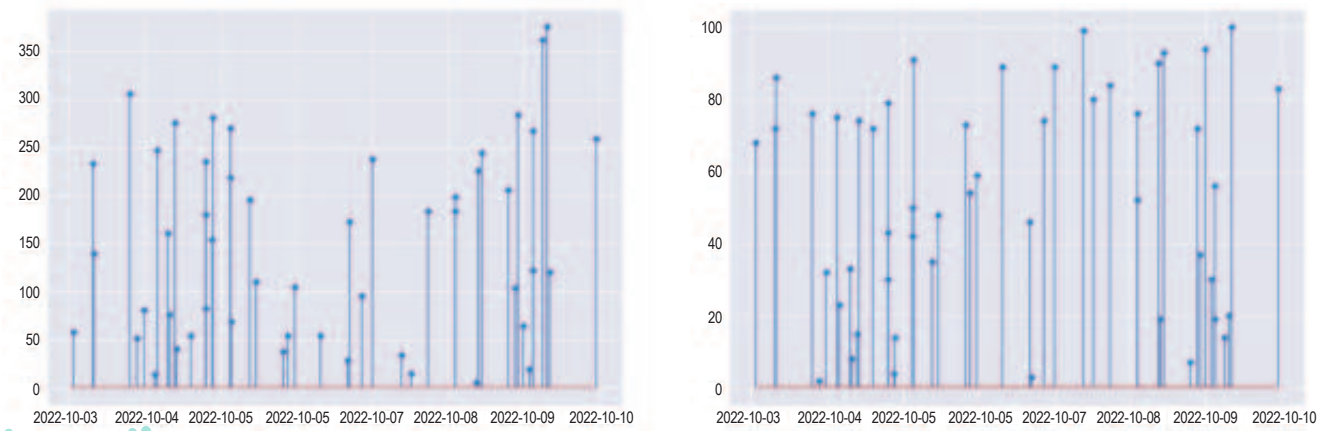
```
# Create histograms for the data using 8 groupings
hist = data.hist(['garbage_drops'],figsize=(10,6),bins=8)
hist = data.hist(['time_to_fill'],figsize=(10,6),bins=8)
```



شكل 7.24: المُخططات البيانية

ستقوم بعد ذلك بإنشاء مُخططين من نوع stem plots لعرض garbage_drops و time_to_fill في كل فترة زمنية.

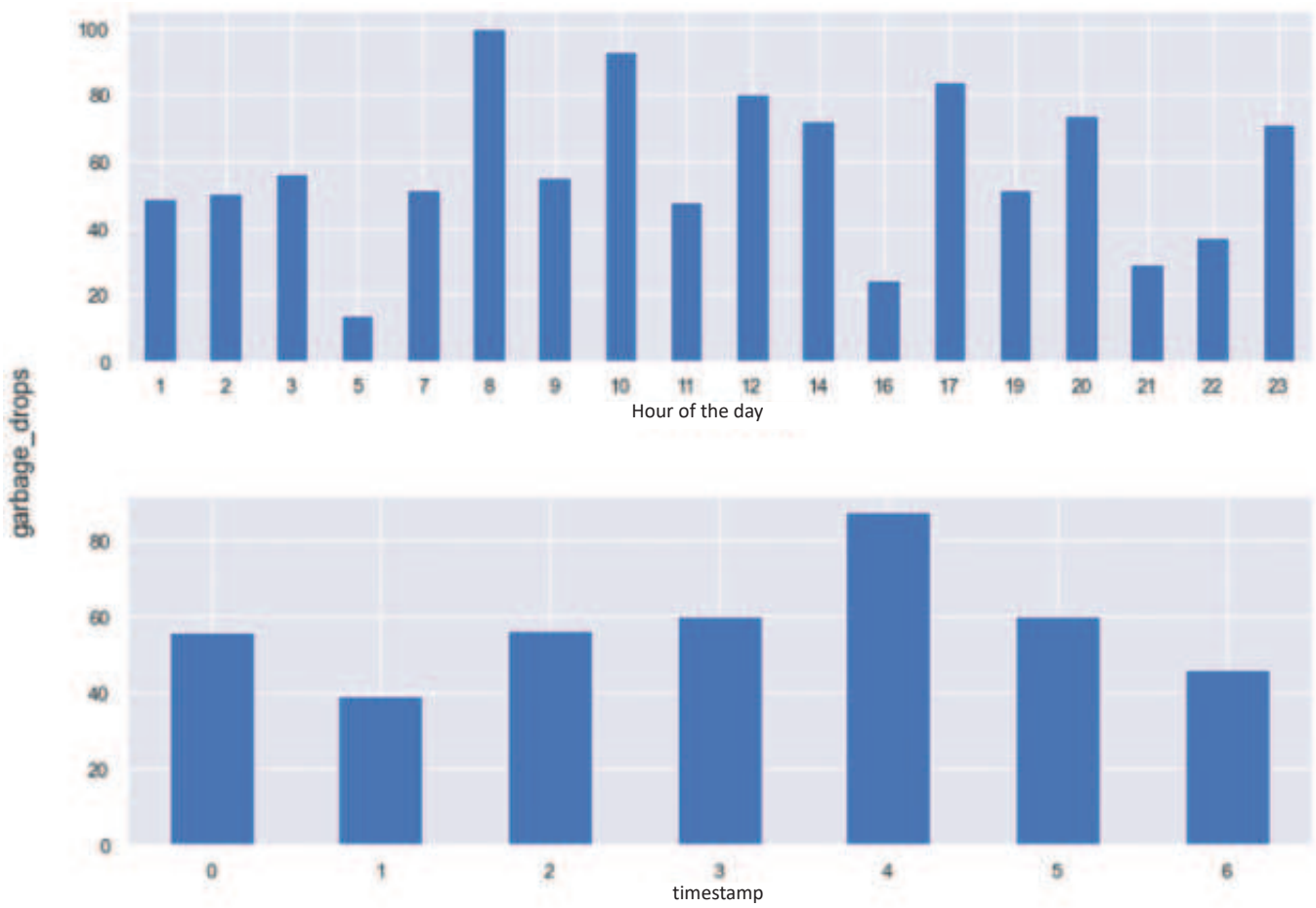
```
# Create stem plots for the data with diamond-shaped ('D') markers
plt.stem(data['timestamp'], data['time_to_fill'], markerfmt='D');
plt.stem(data['timestamp'], data['garbage_drops'], markerfmt='D');
```



شكل 7.25: مخططات stem plots البيانية

ستنشئ في الختام مُخططين يجمعان عدد garbage_drops لكل ساعة من اليوم، ولكل ساعة خلال الأسبوع.

```
# Create 2 plots, one that groups mean garbage amount by hour and one by day
fig, (ax1,ax2) = plt.subplots(2,figsize=(12, 8))
fig.supylabel('garbage_drops')
data.groupby(data["timestamp"].dt.hour)["garbage_drops"].mean().plot(kind='bar',
rot=0, xlabel='Hour of the day', ax=ax1);
# Monday = 0, Sunday = 6
data.groupby(data["timestamp"].dt.day_of_week)["garbage_drops"].mean().
plot(kind='bar', rot=0, ax=ax2);
```



شكل 7.26: مُخططات تجميعية

تمرينات

1 قم بإنشاء مُخطط يوضِّح العلاقة بين ملفي البايثون وملف JSON المحتفظ بالبيانات.

2 قُمْ بإنشاء ملف بلغة البايثون يتصل بثلاثة موضوعات ويكتب معالج أحداث **on_connect** لطباعة معلومات التكوين والموضوعات التي اشترك فيها العميل إلى الواجهة الطرفية Terminal.

3 قُمْ بتحديث الكائن on_message لطباعة المعلومات إلى الواجهة الطرفية حول العميل الذي نشر البيانات، وكذلك الموضوع الذي استُلمت البيانات منه.

4 قُمْ بإنشاء ملف JSON جديد يحتوي على جميع القيم من قائمة الرسائل، واستخدم دالة create_report( ) لإلحاق قيم messages_stack بملف JSON الجديد.

5 في مُفكرة جوبيتر قم بإنشاء مُخطط مبعثر (Scatter Diagram) جديد لنفس البيانات التي قمت بمعالجتها في الدرس.

6 أضف مقطعًا برمجيًا آخر بلغة البايثون يستقبل الرسائل التي نشرتها من البرنامج في التمرين السادس من الدرس الثاني. عندما تستقبل رسالة، اطبع المعلومات المتعلقة بالناشر والمُستقبل والموضوع المشترك على الواجهة الطرفية Terminal.

المشروع

تُستخدم الاتصالات مع بروتوكول (MQTT) على نطاقٍ واسع في المشاريع المختلفة على أرض الواقع. يُمكن تطبيق نفس الهيكليات الخاصة بالاتصال من خلال الناشرين والوسطاء والمُستلمين على مجالات أخرى مختلفة. ستقوم بإنشاء حل متكامل لحديقة ذكية متصلة تُراقب بواسطة بروتوكول (MQTT)، ويُمكن بعد ذلك تعميم مثل هذه الهيكليات على تطبيقات صناعية لحدائق ذكية أكبر حجمًا.

1. قم بإنشاء دائرة جديدة باستخدام لوحة أردوينو ومُستشعر درجة الحرارة ومُستشعر رطوبة التربة وترانزستور ضوئي.

2. قم بإنشاء برنامج بايثون آخر بمثابة مُستقبل للبيانات التي جُمعت بواسطة الأردوينو. سيُطلب من المستخدِم اختيار الموضوع الذي سيتلقى حوله البيانات، ثم يقوم بإنشاء عميل للاشتراك في هذا الموضوع. ستُخزن الرسائل ويُعرض تنبيه إذا وجد ارتفاعًا في القيم المحدثة.

3. قم بإنشاء برنامج بايثون يشترك في موضوع بكافة القراءات ويحفظها في ملف JSON. سيُطلب من المستخدِم اختيار ما إذا كان يريد الاستماع إلى الوسيط وجمع البيانات، أو إنشاء تمثيل للقراءات المُخزنة بالفعل.

4. قم بتشغيل مقاطع البايثون البرمجية الثلاثة في آنٍ واحد، واضبط بيئة الأردوينو لتحديث قراءات البيانات ومراقبة النتائج.

ماذا تعلّمت

- تحليل الطبقات الهيكلية للمُدن الذكية.
- نشر الرسائل باستخدام بروتوكول MQTT.
- إنشاء برنامج البايثون لنشر الرسائل إلى عميل MQTTX Client.
- تخزين التقارير في ملف بيانات JSON.
- إجراء عمليات تحليل البيانات على ملف بيانات JSON باستخدام مُفكرة جوبيتر.

المصطلحات الرئيسة

ناشر	Publisher
جودة الخدمة	Quality of Service
مُستقبل	Receiver
طبقة الخدمات	Services Layer
طبقة الشارع	Street Layer
مُشترك	Subscriber
مُستشعر الإمالة	Tilt Sensor

طبقة المدينة	City Layer
عميل	Client
طبقة مركز البيانات	Data Center Layer
وسيط الرسائل	Message Broker
خادم MQTT	MQTT Server
الترانزستور الضوئي	Phototransistor
نموذج أولي	Prototype

8. محاكاة شبكة مُستشعرات إنترنت الأشياء اللاسلكية

سيتعرف الطالب في هذه الوحدة على تقنيات إنترنت الأشياء المستخدمة في الصناعات الذكية. وسيستخدم بيئة كاب كاربون (CupCarbon) لإنشاء شبكات من المُستشعرات ومحاكاتها. وفي الختام سيُنشئ نموذجًا أوّليًّا لنظامٍ للإنذار ومراقبة الحرائق، ونموذجًا آخر أوّليًّا خاصًّا بالصناعة الذكية والأتمتة.

أهداف التعلُّم

بنهاية هذه الوحدة سيكون الطالب قادرًا على أن:

- يتعرّف على تقنيات إنترنت الأشياء المُستخدمة في الأغراض الصناعية.
- يتعرّف على استخدام الأتمتة الصناعية وأنظمة التحكم في المصانع المُتصلة بإنترنت الأشياء.
- يُنشئ ويُمثِّل شبكات إنترنت الأشياء باستخدام كاب كاربون (CupCarbon).
- يتعرّف على محاكاة شبكة إنترنت الأشياء باستخدام كاب كاربون (CupCarbon).
- يُنشئ مقاطع برمجية بلغة البايثون لبرمجة عُقد الشبكة.
- يُنشئ نموذجًا أوّليًّا لمراقبة الحرائق والإنذار بإنترنت الأشياء.
- يُنشئ نموذجًا أوّليًّا للصناعة الذكية والأتمتة بإنترنت الأشياء.

الأدوات

- كاب كاربون (CupCarbon)

رابط الدرس الرقمي

www.ien.edu.sa

الدرس الأول

مقدمة إلى كاب كاربون

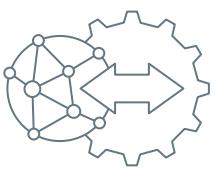
الصناعة الذكية Smart Industry

أصبح لتقنيات إنترنت الأشياء تأثير كبير على جميع مجالات الحياة، بما فيها المجالات الصناعية، وذلك من أجل تقليل التكلفة وتحسين الكفاءة. أدى تطور النماذج الصناعية وازدياد المنافسة إلى تحول التركيز إلى الابتكار وتحسين نماذج الأعمال. قامت الشركات على مدى عقود طويلة بمحاولات لخفض التكلفة الإجمالية لمنتجاتها من خلال خفض تكلفة عمليات التصنيع وسلاسل التوريد، ولكنها أدركت أن المحاولات المستمرة لخفض التكلفة يؤثر بشكلٍ سلبي على خدمة العملاء وجودة الإنتاج. أدت بعض تقنيات إنترنت الأشياء إلى إحداث تغييرات ذات أثر كبير في عمليات التصنيع ومن ذلك:

التصنيع القائم على البيانات Data-Driven Manufacturing

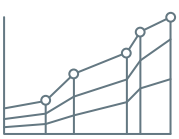
تعمل البيانات الضخمة على تغيير عالم الصناعة، فقد أصبح بإمكان المُصنِّعين الوصول إلى جميع البيانات التي تُنشأ وتُجمع بواسطة الآلات وذلك بهدف المراقبة الفورية للجودة، وتحسين **كفاءة الآلات الكُلية** (Overall Equipment Effectiveness – OEE)، وتقليل وقت الإنتاج. يُعدّ مؤشر **كفاءة الآلات الكلية** (OEE) بمثابة المعيار العالمي لحساب الكفاءة الفعلية لعمليات الإنتاج الصناعية. يبحث المصنِّعون عن طرائق لاستخدام البيانات الضخمة للاستجابةِ لتحولات السوق والتغيرات في حاجات المستهلكين، وذلك من خلال إدخال تقنيات وأدوات صناعية جديدة.

تقارب تقنية التشغيل وتقنية المعلومات
Operational Technology (OT) and Information Technology (IT) Convergence



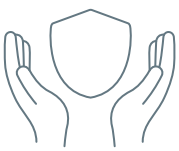
في سياق **إنترنت الأشياء** (Internet of Things – IoT) تضم التقنية التشغيلية في بيئة التصنيع **وحدات تحكم منطقية قابلة للبرمجة** (Programmable Logic Controllers – PLCs)، وأجهزة الحاسب، وغيرها من التقنيات التي تشبه لحد ما تقنية المعلومات، ولكنها تخضع للأعمال التجارية خارج نطاق إدارات تقنية المعلومات. تتيح الشبكات المبنية على بروتوكول IP تكاملًا أعمق بين الآلات وعمليات التصنيع، وتزيل الفجوة بين شبكات الصناعة والأعمال التجارية. يبحث المُصنِّعون عن طرائق لدمج عملياتهم في إطار بُنية تحتية موحدة للشبكات تتجاوز طرائق التخزين التقليدية.

تقنية أفضل وتكلفة أقل Improved Technology with Lower Costs



أصبحت إمكانية الاتصال والمراقبة وتحسين الأجهزة قابلة للتطوير والأتمتة، وقائمة على بيئات تشغيلية متطورة نتيجة ظهور تقنيات جديدة. في ظل هذا التقدم التقني الكبير، يمكن اعتبار الآلات جزءًا من نظام شبكة متصل متكامل بدلًا عن كونها نظامًا مستقلًا بذاته عن باقي عملية التصنيع، كما أدى التقارب في الحوسبة والشبكات والحماية إلى تقليل تكلفة توصيل الأجهزة في النظام المتكامل.

تعزيز الكفاءة والسلامة Enhanced Efficiency and Safety



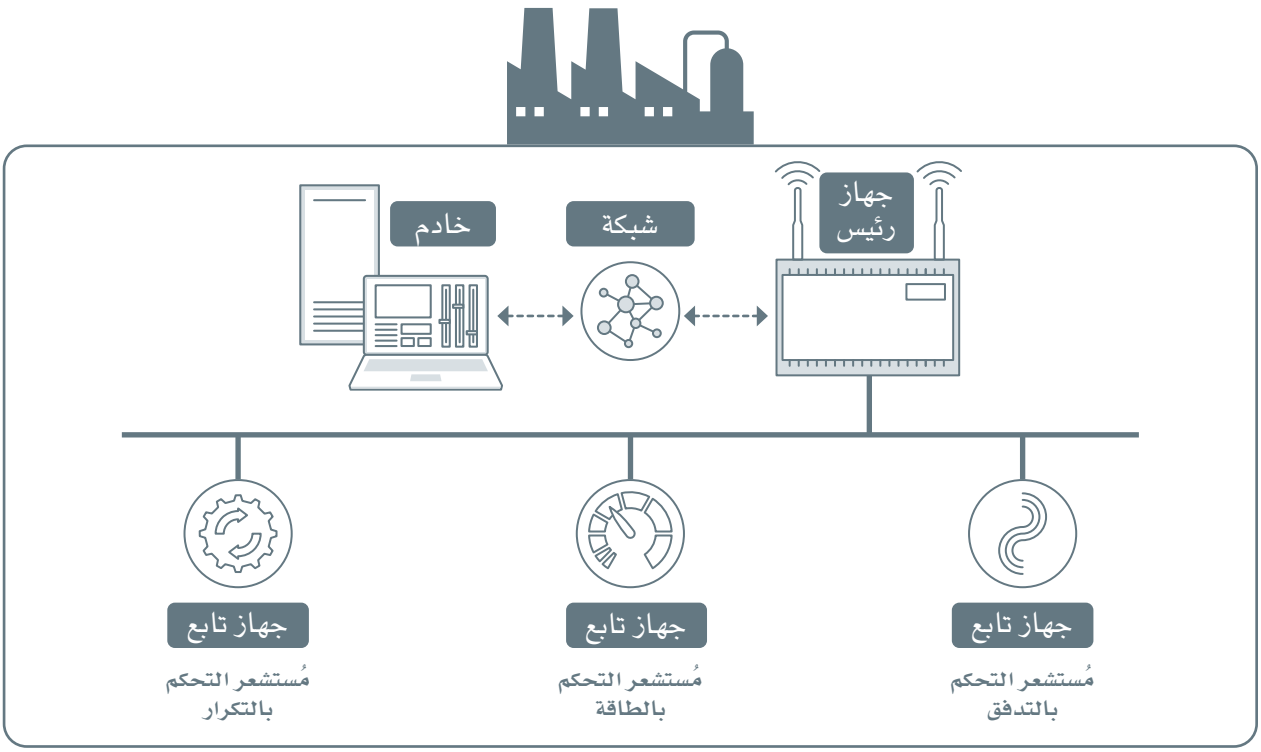
تسعى المصانع، لاسيما في قطاعات الأغذية والمشروبات، إلى الوصول إلى التحكم الآلي والأتمتة والتصنيع دون التدخل البشري لمهمات التصنيع المختلفة. يُمكن توظيف إنترنت الأشياء إضافةً إلى استخدام الروبوتات ومعالجة الصور لتمكين المصانع الحديثة من تحسين الكفاءة والسلامة.

هيكلية المصنع المُتصل An Architecture for the Connected Factory

بدأت الشركات بالدمج بين **الأتمتة الصناعية وأنظمة التحكم** (Industrial Automation and Control Systems – IACS) مع تطبيقات تقنية المعلومات وأدوات التحليلات لتوفير إمكانات تحكم وتحليلات تشغيلية مفيدة للأعمال. يُستخدم هذا الدمج للتحكم في العمليات الأساسية، أو مراقبة استخدام تدابير السلامة عند حدوث أي طارئ. يهدف دمج **الأتمتة الصناعية وأنظمة التحكم** (IACS) إلى تحقيق الجودة والكفاءة في الإنتاج مع الحفاظ على مستوى عالٍ من التكامل والموثوقية.

بروتوكول التحكم في الإرسال وبروتوكول مودبس
Modbus / Transmission Control Protocol - TCP

يُستخدم بروتوكول **مودبس** (Modbus) في القطاع الصناعي بشكل شائع لإدارة الأجهزة الرئيسة والفرعية. تم تحويل **مودبس** (Modbus) إلى بروتوكولات الاتصالات المستخدمة على نطاقٍ واسع مثل Ethernet وTCP/IP، وعلى غرار تقنيات التحكم الأخرى في الأتمتة، يُستخدم مودبس كبروتوكول معياري مفتوح مُوثق ومُثبت في جميع أنحاء العالم. تُعدُّ آلية بروتوكول مودبس في إدارة الأجهزة الرئيسة والتابعة مناسبة تمامًا لطبيعة بروتوكول **التحكم في الإرسال** (TCP) الخاص بالاتصالات، ولكن بشكلٍ أقل تنوعًا.



شكل 8.1: بروتوكول الشبكة مودبس(Modbus)

تحديات المصنع المتصل Connected Factory Challenges

أصبح القطاع الصناعي أحد أبرز أهداف القرصنة الإلكترونية ومهاجمي الإنترنت. تسبب التقارب الحاصل بين الشبكات في المصانع والأعمال التجارية بظهور ثغرات أمنية لعمليات التصنيع، والتي كانت تجري تقليديًا بمعزل عن العمليات الأخرى.

يُعدّ الفصل بين شبكة المصنع الأساسية وشبكة تقنية المعلومات أبسط حل للتغلب على هذه الهجمات في الكثير من الأحيان. رغم أن هذا الحل يُعدّ فعّالًا وعمليًا، إلا أنه سيمنع التواصل مع عمليات الطبقة العليا وسيحدُّ من القدرات الممكنة لتحسين الأعمال المُدعَّمة بإنترنت الأشياء، وقد تظهر المزيد من المخاطر المُحتملة من أجهزة الحاسب المحمولة والأجهزة الحاسوبية الأخرى المتاحة في المصانع للعاملين الذين يتمتعون بوصول غير مُقيَّد للأجهزة.

الحوسبة الطرفية في المصنع المتصل Edge Computing in the Connected Factory

يمكن للآلات الموجودة في المصنع إنتاج كميات هائلة من البيانات، وبالتالي تبرز مشكلة تخزين تلك البيانات، وقد عالجت العديد من المصانع هذه المشكلة من خلال نشر الحواسيب لتخزين هذه البيانات. أدى جمع البيانات من أجهزة الحاسب الموجودة في المصنع إلى ظهور مشكلات عديدة تتعلق بالصيانة والأمان، فكما هو معروف، يتطلب كل حاسب تصحيحات وترقيات لنظام تشغيله، كما تزداد أعطال الأجهزة بشكلٍ ملحوظ في المصانع، إذ إنّ معظم تلك الأجهزة لا تُصمّم لتحمُّل الظروف المختلفة فيها، وتشكِّل هذه المشكلات عائقًا أمام عمليات التصنيع في جمع البيانات ومعالجتها والاستجابة لها بكفاءة. يُمثّل هذا النهج عائقًا كبيرًا أمام تطوير الأفكار والفوائد التجارية المُحتملة التي قد توفرها تحليلات البيانات الصناعية. تساعد التطورات الجديدة في إضافة القدرات المحوسبة في الشبكات الطرفية على حل هذه المشكلات. بدأ المُصنِّعون بإدراك ميزات توصيل الآلات بخدمات الحوسبة المتطورة مع أجهزة الحوسبة الطرفية المُدمجة بالآلة القريبة من الحافة، والتي تتضمن قدرات التبديل والتوجيه والأمن معًا بشكلٍ دائم.

صناعة النفط والغاز Oil and Gas Industry

يُعدّ كل من النفط والغاز من أهم الموارد التي يستخدمها المجتمع الحديث، وذلك بدءًا بالبُنى التحتية للمواصلات، إلى تصنيع المواد البلاستيكية. يعتمد كل عنصر من عناصر الحياة الحديثة تقريبًا على توفُّر السلع التي تعتمد على هذه الصناعة. تهتم شركات النفط والغاز بشكل أساسي بخفض التكاليف، وزيادة الكفاءة والسرعة، وزيادة عوائد الاستثمارات. يُعدُّ التحكم في تكاليف الإنتاج وتعزيز الصحة والسلامة العامة وخاصة في الصناعات الخطرة من بين **مؤشرات الأداء الرئيسة** (Key Performance Indicators – KPIs) الأكثر أهمية في القطاع الصناعي.

شكل 8.2: مراقبة إنتاج النفط

على غرار القطاعات الأخرى، تستخدِم شركات النفط والغاز إنترنت الأشياء للعديد من الأغراض بما فيها ما يلي:

- مراقبة حالة المُعدات الصناعية أو سلوكها للرؤية والتحكم.
- تحقيق أقصى قدر من الكفاءة للعمليات والموارد.
- تحسين عملية اتخاذ قرارات الأعمال التجارية.

تحديات الصناعة الرئيسة كمُحركات للتحول إلى الرقمنة Industry Key Challenges As Digitization Drivers

إن **إنترنت الأشياء** (IoT) والرقمنة – وهي عملية الاستفادة من التقدم الكبير في تقنية المعلومات لتطوير حلول وتقنيات جديدة للأعمال وإجراءاتها – تُمهد الطريق لتحقيق مكاسب تحسين الكفاءة التي كانت مستحيلة سابقًا وكذلك نماذج الأعمال الجديدة.

- النمذجة والتحليلات المتقدمة.
- البيانات الضخمة.
- تقارب **تقنية المعلومات** (IT) / **التقنية التشغيلية** (OT).
- الآلات الذكية.
- التنقل والتخزين السحابي.
- إدارة أداء الأصول.

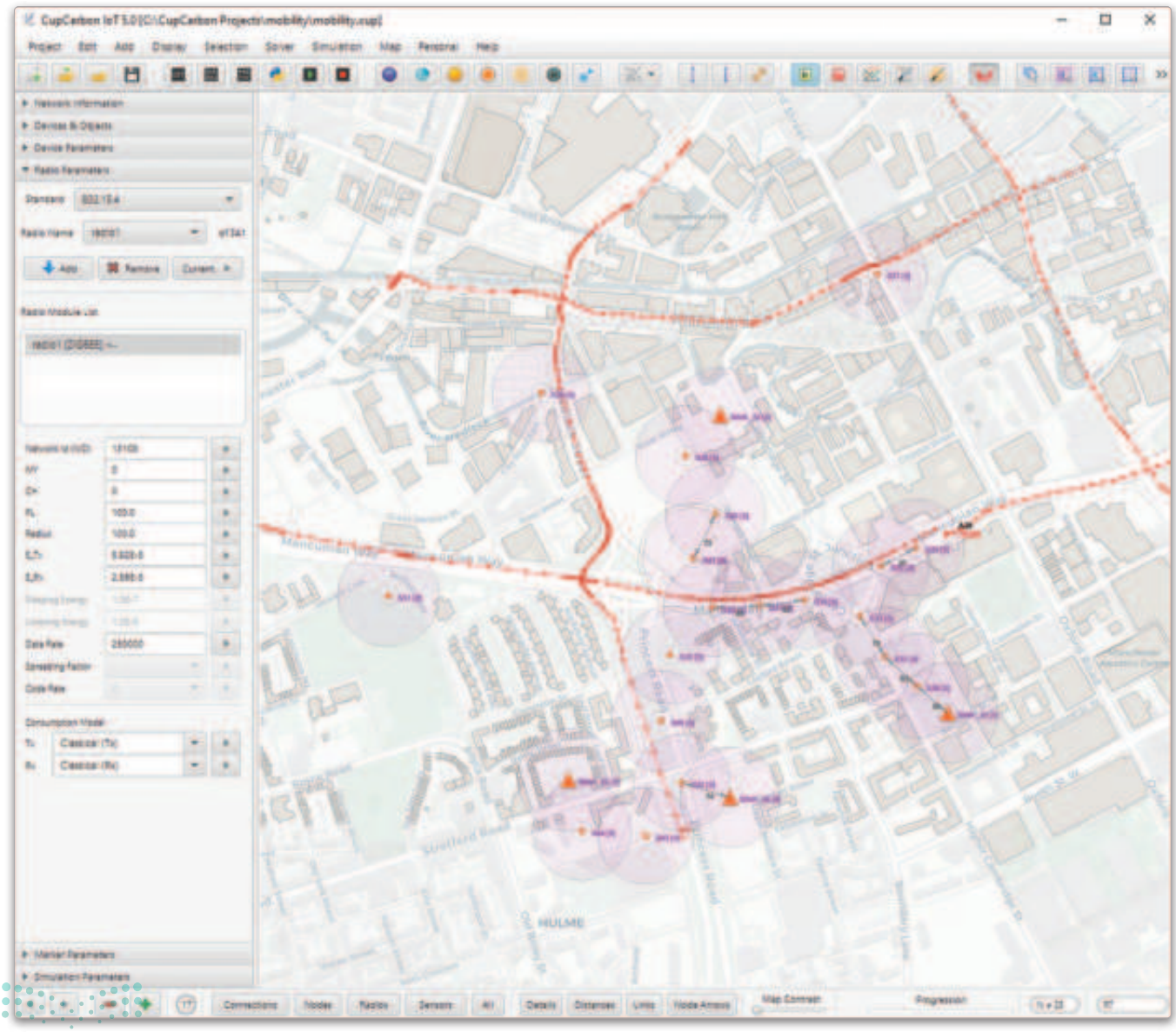
مثال

يُعدّ معمل الغاز في العثمانية أحد أكبر المصانع لمعالجة الغاز في المملكة العربية السعودية. تعمل حلول إنترنت الأشياء والذكاء الاصطناعي على تحسين الإنتاجية وموثوقية هذه المنشأة. فتُستخدم الطائرات دون طيار، والكائنات الذكية لمراقبة مُعدات مصفاة خطوط أنابيب الغاز وتُستخدم طرائق تحليل البيانات لتحسين الاستخدام. يوجد الآلاف من مُستشعرات إنترنت الأشياء تراقب حقل خريص النفطي.

ما برنامج كاب كاربون؟ What CupCarbon is

برنامج **كاب كاربون** (CupCarbon) هو مدينة ذكية افتراضية، وبيئة محاكاة لشبكة مُستشعرات لاسلكية بإنترنت الأشياء. يُمكن استخدامه لتصميم وإنشاء وتمثيل شبكات إنترنت الأشياء المُكونة من عُقد وأجهزة وأحداث وأمور أخرى. يوفِّر هذا المحاكي عددًا كبيرًا من الأدوات لتكوين الشبكات التي أُنشئت لاختبارها وتحسينها. كما تم به تضمين البروتوكولات الشائعة التي عُرضت سابقًا مثل **زيجبي** (Zigbee) و**واي فاي** (Wi-Fi) و**لورا** (LoRa)، بالإضافة إلى استخدام **خريطة الشارع المفتوح** (OpenStreetMap) كواجهة للتمكُّن من إجراء عمليات المحاكاة من أي مكان.

بعد إنشائك للشبكة المطلوبة بنجاح، يُمكنك استخدام برنامج كاب كاربون لبرمجة وحدة تحكم **أردوينو** (Auruino) يُمكن تجريبها على أجهزة حقيقية. تتواصل العُقد مع بعضها باستخدام البرمجة النصية. يستخدِم المحاكي لغته البرمجية الخاصة المعروفة باسم **سينسكريبت** (SenScript)، كما أنه يدعم لغة البايثون. ستقوم في هذه الوحدة ببرمجة العُقد باستخدام لغة البايثون.



شكل 8.3: مشروع في كاب كاربون (CupCarbon)

لتنزيل كاب كاربون (CupCarbon) وتشغيله:

> افتح متصفحك وقم بتنزيل الملف من الرابط:
http://binary-academy.com/dnld/KSA/IOT2/BinaryCupCarbon.zip (1)

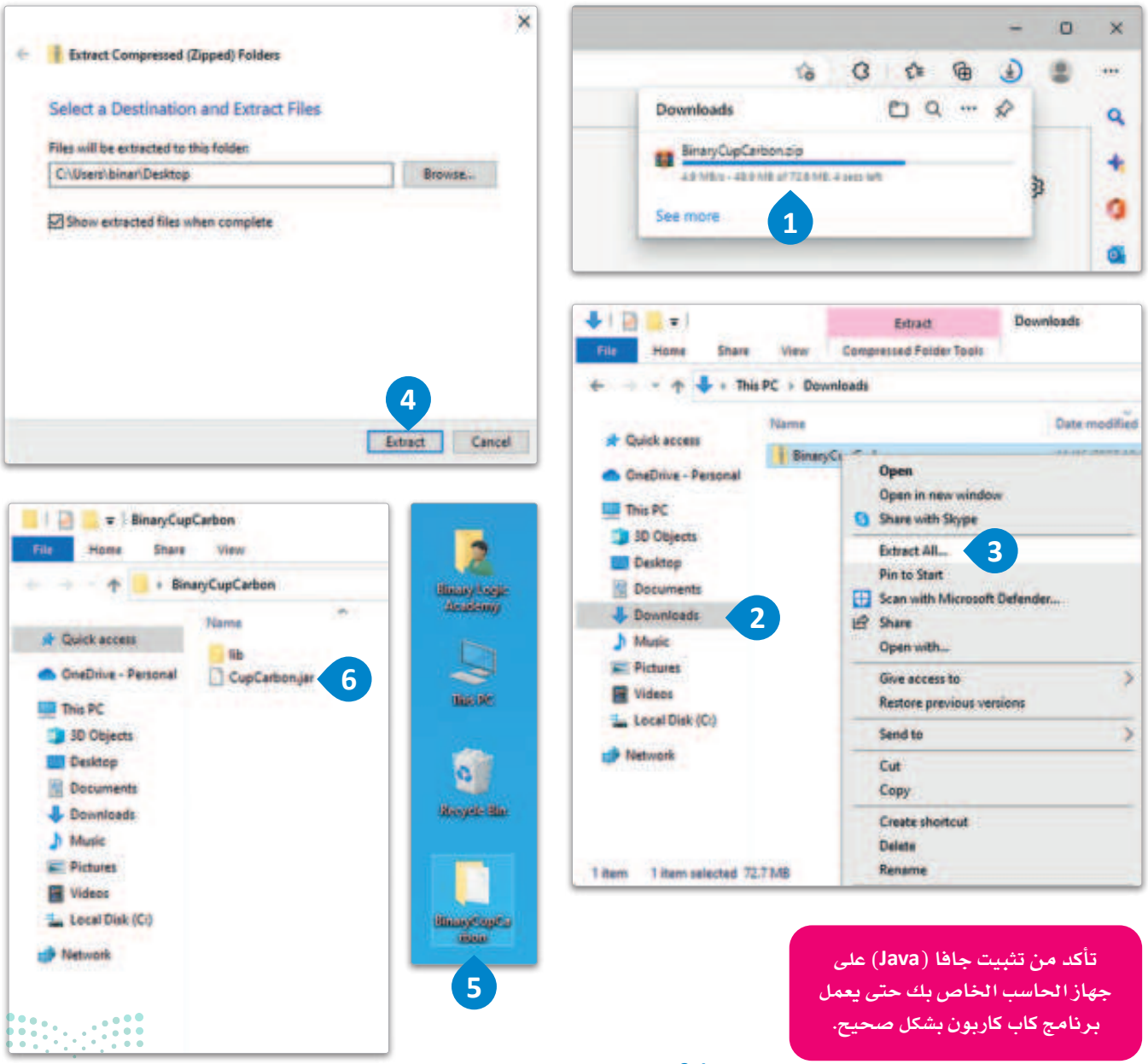
> افتح مستكشف الملفات، وابحث عن الملف الذي تم تنزيله في مُجلد Downloads (التنزيلات). (2)

> اضغط بزر الفأرة الأيمن على الملف واختر Extract All (استخراج الكل). (3)

> اختر سطح المكتب الخاص بك كوجهة للاستخراج، واضغط على Extract (استخراج). (4)

> ابحث عن المجلد المُستخرج في سطح المكتب وافتحه. (5)

> اضغط ضغطًا مزدوجًا على CupCarbon.jar لبدء برنامج CupCarbon (كاب كاربون). (6)



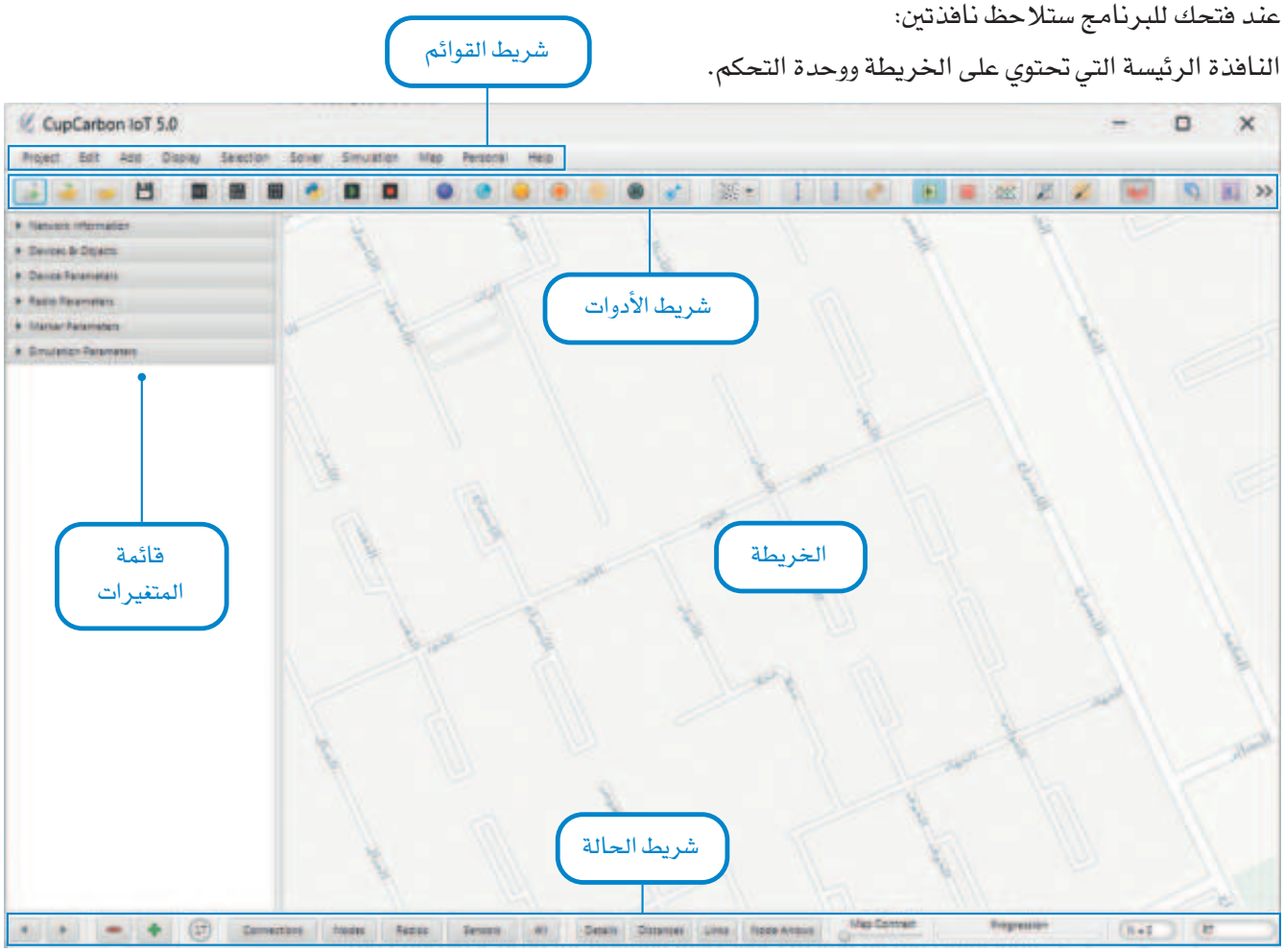
شكل 8.4: خطوات تنزيل كاب كاربون

تأكد من تثبيت جافا (Java) على جهاز الحاسب الخاص بك حتى يعمل برنامج كاب كاربون بشكل صحيح.

نافذة برنامج كاب كاربون The CupCarbon Windows

عند فتحك للبرنامج ستلاحظ نافذتين:

النافذة الرئيسة التي تحتوي على الخريطة ووحدة التحكم.



شكل 8.5: النافذة الرئيسة لبرنامج كاب كاربون

تُستخدم وحدة التحكم لطباعة الرسائل التي أُنشئت بواسطة المحاكاة، ولإخراج رسائل الأخطاء لمساعدة المستخدِم على تصحيح الخطأ في البرامج النصية.

شكل 8.6:وحدة تحكم كاب كاربون

بدء الاستخدام Getting Started

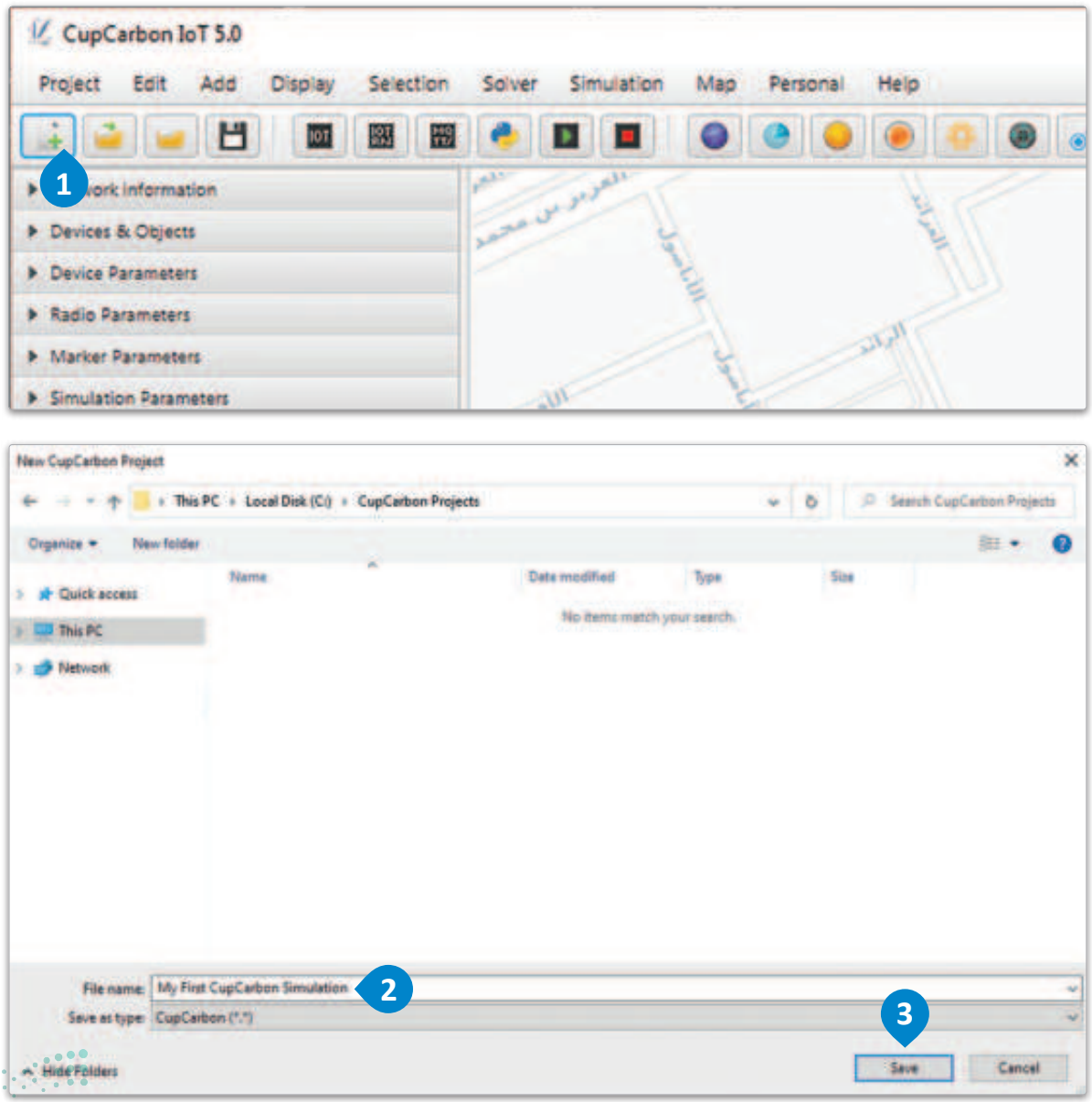
ستُنشئ في هذا الدرس محاكاة بسيطة لعُقدة إنترنت أشياء تطبع رسائل من أجل أن تعتاد على استخدام بيئة كاب كاربون.

في البداية ستنشئ مشروعًا جديدًا:

لإنشاء مشروع جديد:

> اضغط على New Project (مشروع جديد) من شريط الأدوات. 1

> اختر الموقع الذي تريده لحفظ المشروع، ثم اكتب "My First CupCarbon Simulation" في حقل File name (اسم الملف)، 2 ثم اضغط Save (حفظ). 3



شكل 8.7: إنشاء مشروع جديد

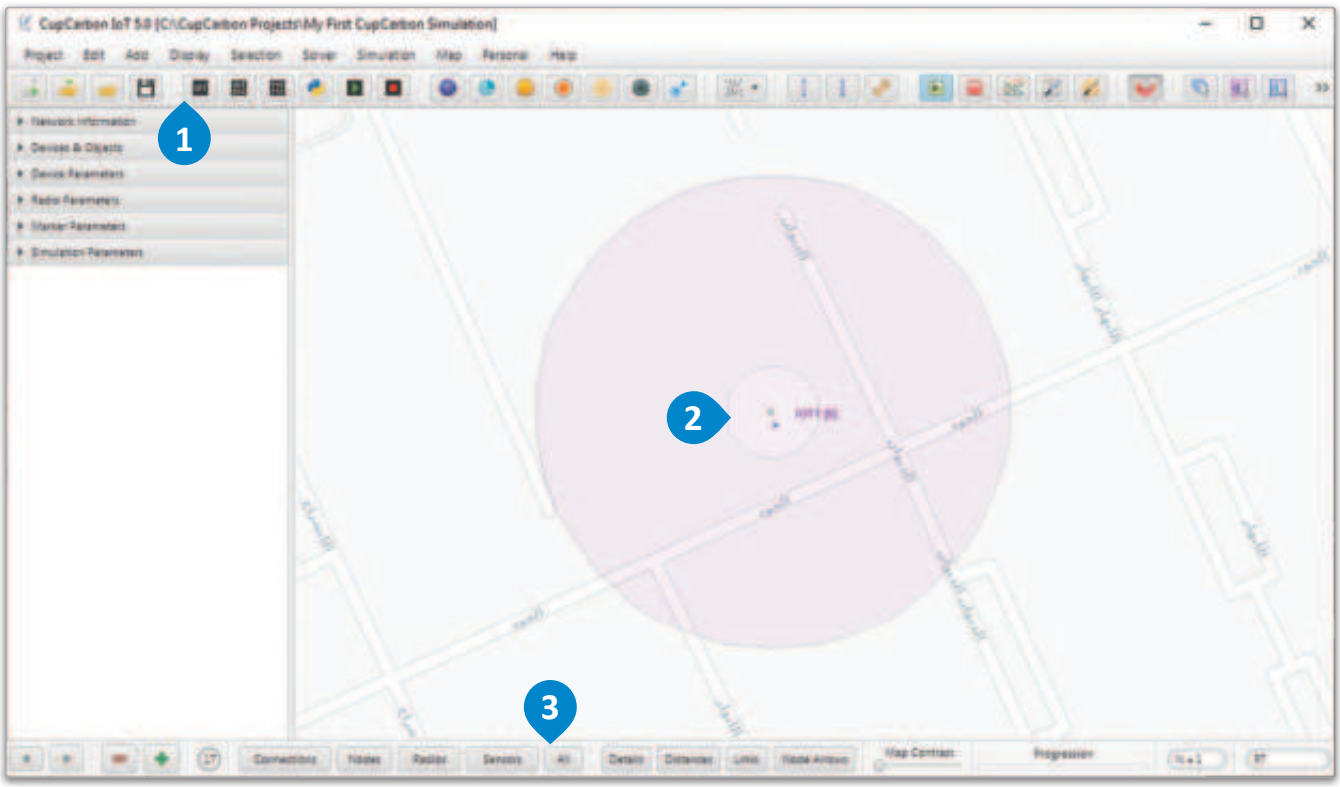
إضافة عُقدة Placing a Node

يمكنك في شريط الأدوات العثور على الكائنات المختلفة التي ستستخدمها في مشاريعك، والتي ستُنتج إما إشارات وتتواصل مع بعضها، أو ستنفذ إجراءات معينة. من هذه الكائنات الكائن IoT Node (عُقدة إنترنت أشياء) والذي يُمكن وضعه على الخريطة، ويمكن إعطاؤه مقطعًا برمجيًّا لتشغيله.

العُقد هي اللبِنات الأساسية لتكوين كاب كاربون. يُعرض في العُقدة المُعرِّف الخاص بها مع دائرتين حولها، ودائرة داخلية تشير إلى نصف قطر المُستشعر المُستخدَم للكشف عن المُستشعرات، ودائرة خارجية تكشف عن الأجهزة اللاسلكية مثل العُقد الأخرى.

لإضافة عُقدة:

- اضغط على IoT Node (عُقدة إنترنت أشياء) من شريط الأدوات. 1
- اضغط على الخريطة لإضافة العُقدة. 2
- اضغط على All (الكل) من شريط State (الحالة). 3
- اضغط على Esc في لوحة المفاتيح.



شكل 8.8: إنشاء مشروع جديد

إنشاء المقطع البرمجي Creating a Script

ستقوم الآن بإنشاء مقطع برمجي بسيط يطبع رسالتين ذاتيتين بالتناوب على العُقدة. المقطع البرمجي المُستخدَم هو كما يلي:

```
import time
while node.loop():
    node.print("hello")
    time.sleep(1)
    node.print("world")
    time.sleep(1)
```

تأكد من استخدام المسافة البادئة المناسبة داخل التكرار (Loop) حتى يعمل المقطع البرمجي بشكل صحيح.

يجب عليك أولًا تضمين مكتبة time في البايثون، ستُستخدم دالة sleep المُدمجة لتأخير عملية الطباعة. يجب تضمين التعليمات البرمجية للعُقدة داخل التكرار while node.loop( ). يمكن للعُقدة الطباعة الذاتية باستخدام node.print( )، ويمكنها "السكون" -أي ألا تفعل شيئًا- باستخدام time.sleep( ). تأخذ دالة print( ) كمُعامل الرسالة المراد طباعتها على شكل نص، على سبيل المثال node.print ("hello world")، وتأخذ دالة sleep كمُعامل عدد موجب يشير لعدد الثواني التي تريدها للعُقدة ليتم التأخير الزمني، على سبيل المثال مع time.sleep( ) سيُنفّذ سكون للعُقدة لمدة 3 ثوان.

ستطبع العُقدة في برنامجك كلمة "hello"، ثم ستنتظر لمدة ثانية واحدة وتطبع "world"، وتنتظر مرةً أخرى لمدة ثانية واحدة، ثم تبدأ مرة أخرى من البداية بلا توقف ما لم يتم إنهاء المحاكاة.

لإنشاء المقطع البرمجي:

> اضغط على Python (البايثون) من شريط الأدوات. 1
> اكتب التعليمات البرمجية بلغة بايثون في المُحرر النصي. 2
> اكتب "hello" في حقل File name (اسم الملف). 3
> اضغط على Save (حفظ). 4
> أغلق نافذة محرر البايثون. 5

شكل 8.9: مُحرر البايثون

لإدراج المقطع البرمجي وتشغيل المحاكاة:

> اضغط على العُقدة. 1

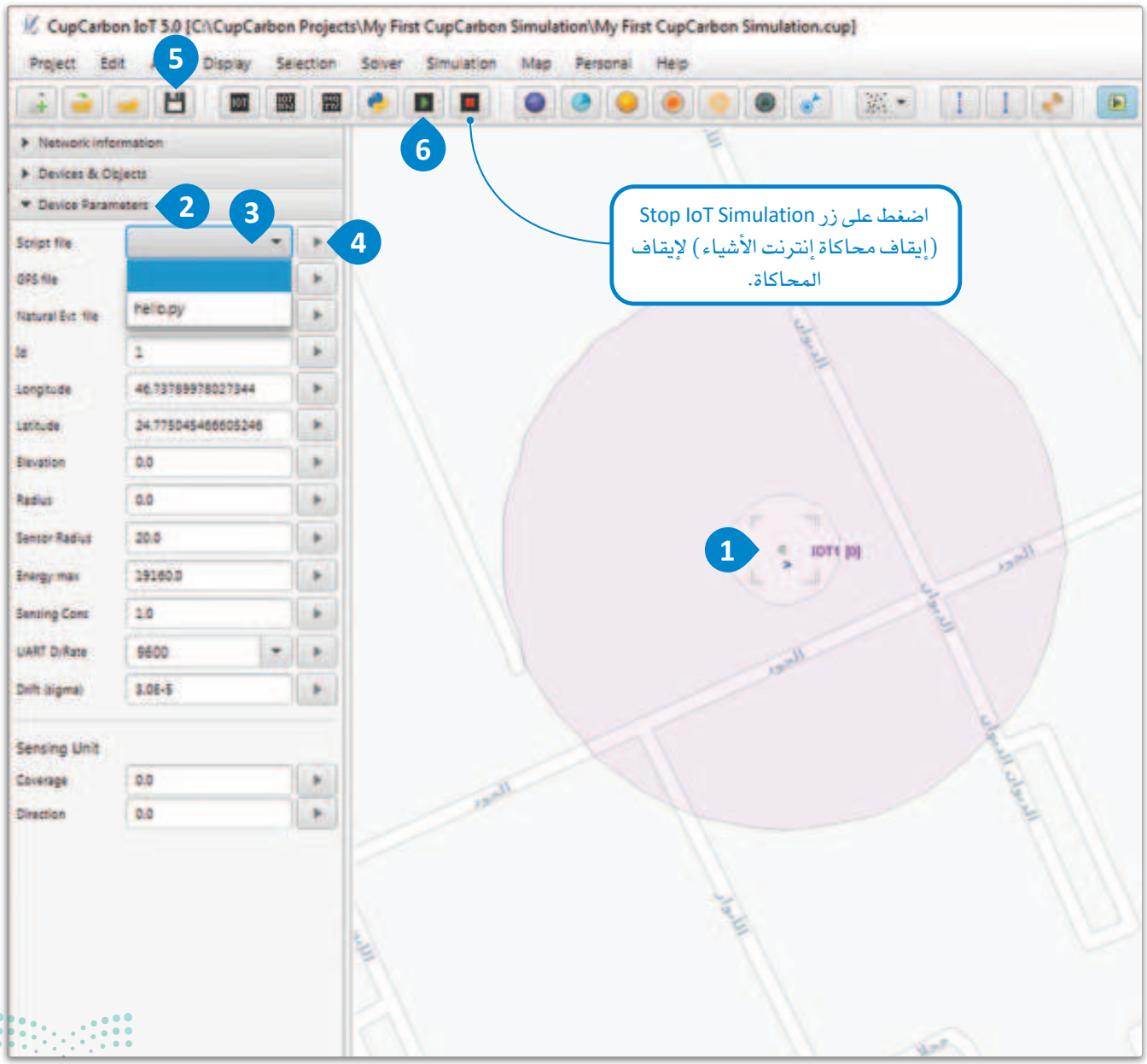
> اضغط على علامة تبويب Device Parameter (مُعامِل الجهاز) في قائمة Parameter (مُعامِل). 2

> اضغط على صندوق Script file (ملف البرنامج). 3

> من القائمة المنسدلة، اختر المقطع البرمجي hello.py، واضغط على الزر الموجود على اليمين لإدراج المقطع البرمجي في العُقدة. 4

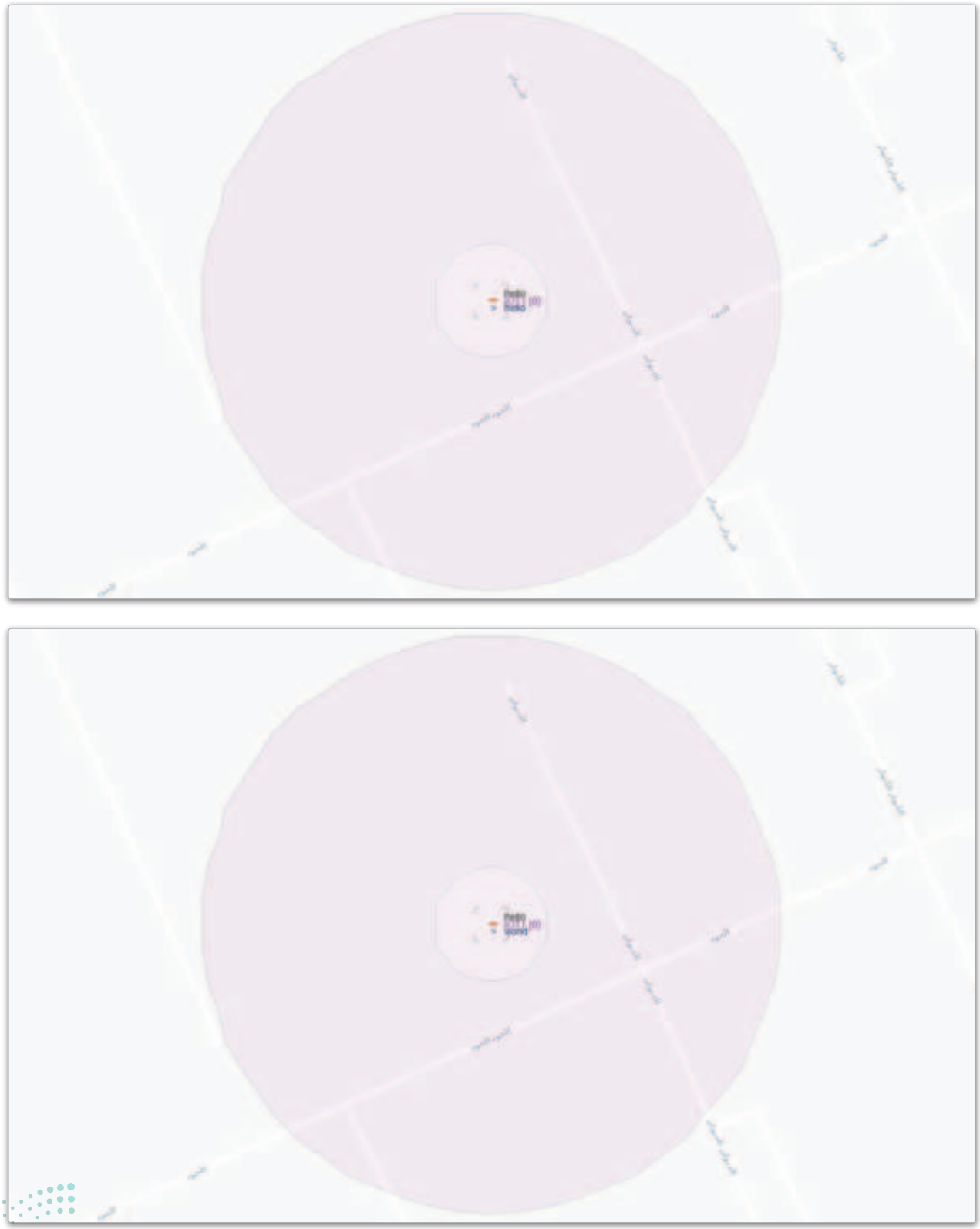
> اضغط على Save project (حفظ المشروع) من شريط الأدوات. 5

> من شريط الأدوات، اضغط على Run IoT Simulation (تشغيل محاكاة إنترنت الأشياء). 6



شكل 8.10: إدراج المقطع البرمجي وتشغيل المحاكاة

كما هو متوقع، فإن العقدة ستتناوب في طباعة النصين "hello" و"world" لمدة ثانية واحدة لكل منهما.



شكل 8.11: حالات المحاكاة

تمرينات

1

حدِّد الجملة الصحيحة والجملة الخاطئة فيما يلي:	صحيحة	خاطئة
1. لا يُمكن استخدام مراقبة البيانات لزيادة كفاءة تحسين المعدات بشكل عام.	○	○
2. يُمكن لأقسام التقنية التشغيلية (OT) وتقنية المعلومات (IT) الدمج بين جميع قطاعات التصنيع في نطاق شبكي واحد.	○	○
3. يسهم توصيل أجهزة المصنع بشبكة واحدة في تقليل التكاليف.	○	○
4. يمكن للعمليات الأوتوماتيكية التي لا تعمل باللمس في مصنع الأطعمة والمشروبات تحسين جودة المُنتج النهائي.	○	○
5. لا يُمكن أن تتعرض الحواسيب الداخلية في المصانع إلى مخاطر أمنية.	○	○
6. قد تفقد أجهزة المصنع غير الموصولة بالشبكة الطرفية بيانات قيمة في حالة تعطلها.	○	○
7. يُمكن لأنظمة إنترنت الأشياء في صناعات النفط والغاز الحد من تعرض العمال للخطر.	○	○
8. يُمكن في برنامج كاب كاربون (CupCarbon) محاكاة بروتوكول زيجبي (ZigBee) الخاص بالأشياء الذكية.	○	○
9. يُمكن برمجة عُقد كاب كاربون بواسطة لغة البايثون فقط.	○	○
10. يُمكن في برنامج كاب كاربون إنتاج مخططات لوحدات التحكم الدقيقة مثل الأردوينو.	○	○

2 صنِّف تقنيات إنترنت الأشياء الرئيسة التي ستُغيِّر عمليات التصنيع التقليدية.

3 قَدِّم تحليلًا لكيفية تعرض المصانع المتصلة بأنظمة إنترنت الأشياء للهجمات الإلكترونية.

4 صِفْ مدى مساهمة الحوسبة الطرفية في المصانع المتصلة في تحسين كفاءتها وقدرتها الإنتاجية.

5 اذكر كيفية استخدام حلول إنترنت الأشياء الصناعية الذكية لتحسين العمليات في صناعة النفط والغاز.

6 أنشئ عُقدة في برنامج كاب كاربون واكتب برمجتها بحيث يكون هناك تكرار يجعل العُقدة تطبع الرسالة "A" لمدة ثانية واحدة، و"B" لمدة ثانيتين، و"C" لمدة 3 ثوانٍ. قم ببرمجة العُقدة وتشغيل المحاكاة.

7 أنشئ عُقدتين في برنامج كاب كاربون واكتب برمجتهما بواسطة البايثون. سيعرض كل مقطع برمجي رسالة "blink" متكررة. ستُرسل كل عُقدة في كل مرة رسالة عندما تكون الأخرى غير نشطة.

قم ببرمجة العُقدتين وتشغيل المحاكاة.

رابط الدرس الرقمي

www.ien.edu.sa

الدرس الثاني

الاتصال في شبكة إنترنت الأشياء

الاتصال بين الأجهزة Communication between Devices

تتكون شبكة إنترنت الأشياء من العديد من الأجهزة التي تُرسل وتستقبل البيانات بين بعضها. تتمايز هذه الأجهزة في إمكانياتها المختلفة مثل النطاق، وعرض النطاق الترددي للبيانات، واستهلاكها للطاقة، وبالتالي تقوم بتشغيل مجموعات مختلفة من الأوامر لتوفير العديد من الوظائف. ستُنشئ في المشروع الآتي شبكة مثل هذه، وستستكشف اللبنات البرمجية الأساسية المكونة لها.

مراقبة الحريق والتحذيرات Fire Surveillance and Notification

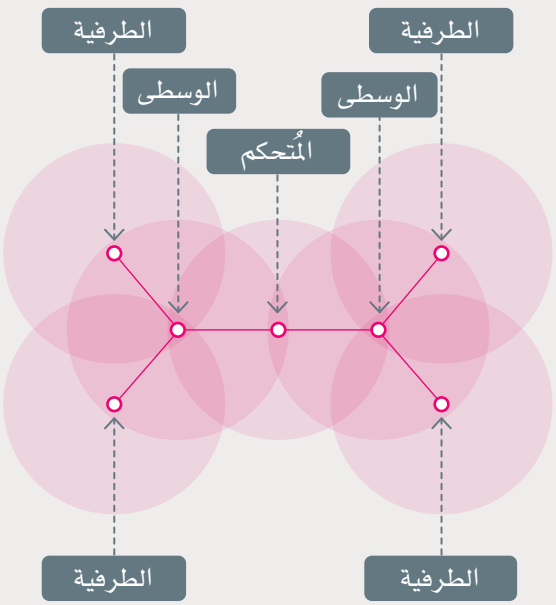
ستُنشئ في هذا الدرس مشروعًا يحاكي نظام مراقبة الحرائق في المصانع.

سيتم في هذه المحاكاة إنشاء حرائق عشوائية في مصنع، وسيبلغ النظام وحدة التحكم الرئيسة بالمصنع عن القطاع الذي يوجد فيه الحريق. سيُنفّذ ذلك باستخدام مجموعة متنوعة من العُقد ذات الوظائف المختلفة التي ستتواصل مع بعضها لتمرير الرسالة بدءًا من العُقد **الطرفية** (Edges) مرورًا بالعُقد **الوسطى** (Proxies) لتصل أخيرًا إلى العُقدة الداخلية: **وحدة المُتحكم الرئيسة** (Main Controller).

العُقد ووظائفها
The Nodes and their Functions

ستستخدِم ثلاثة أنواع من العُقد في مشروعك لتمثيل قطاعات المصنع المختلفة:

- قد يحدث حريق في **الأطراف** (Edges)، وسيتم تحديده بواسطة مولد أرقام عشوائي. في مثل هذه الحالات، ستطبع العُقدة رسالة ذاتية وتُرسل رسالة مختلفة تحتوي على رقم القطاع إلى عُقدة **الوسطى** (Proxy) المجاورة لها، ثم ستسكُن لمدة زمنية محددة.
- سيقرأ الوكلاء أي رسائل قد تلقوها ويعيدونها إلى **وحدة المُتحكم الرئيسة** (Main Controller). كما قد يقومون بطباعة رسالة ذاتية لإبلاغ المستخدِم بما يقومون به.
- تقوم وحدة المُتحكم الرئيسة أيضًا بقراءة أي رسائل قد تكون تلقتها، وتطبع الرسائل التي أُنشئت بواسطة العُقد الطرفية التي تخبر المستخدِم بالحريق.



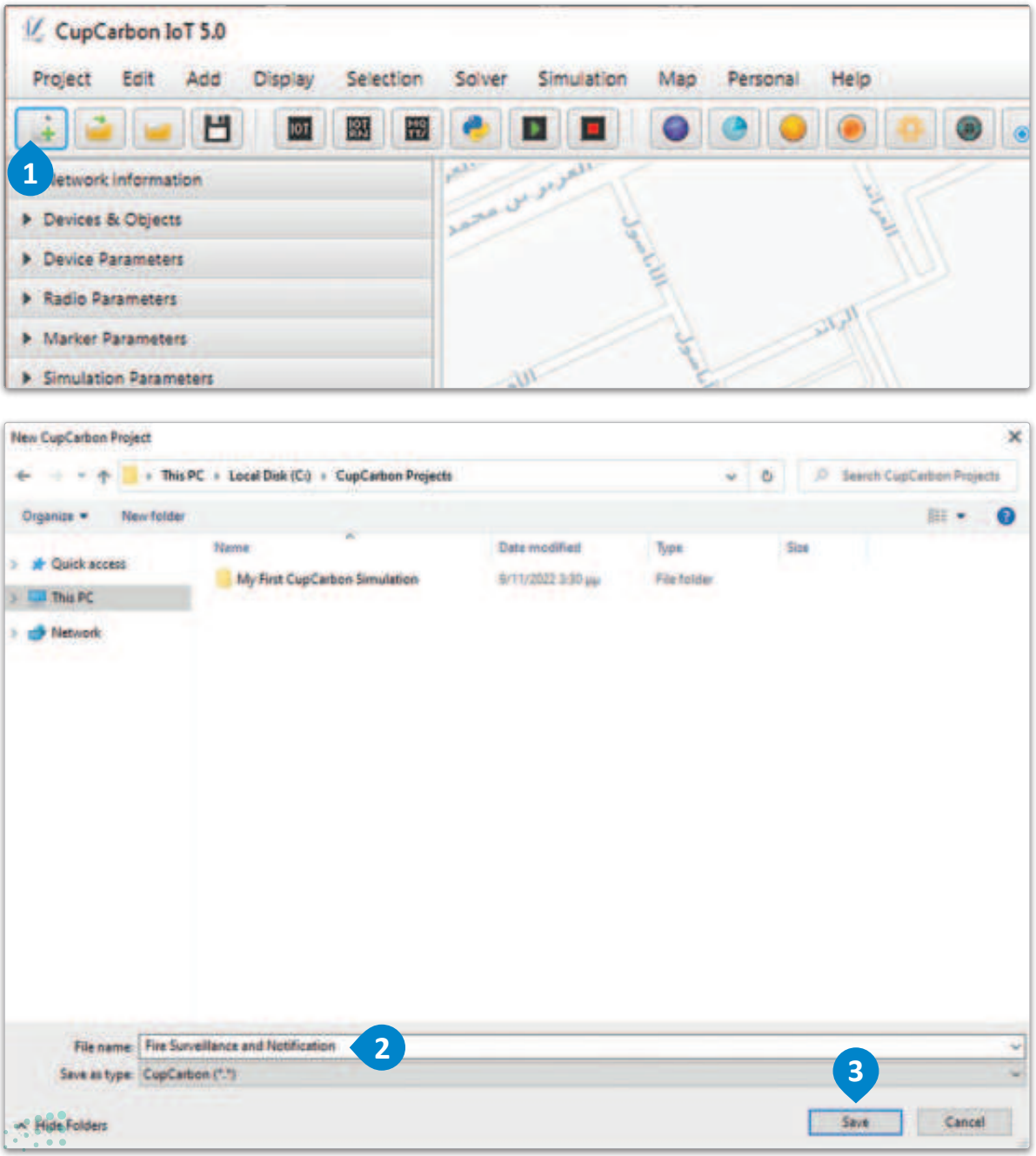
شكل 8.12: العُقد ووظائفها

لنبدأ بإنشاء مشروع جديد:

لإنشاء مشروع جديد:

> اضغط على New Project (مشروع جديد) من شريط الأدوات. 1

> اختر الموقع الذي تريد تخزين المشروع فيه، ثم اكتب "Fire Surveillance and Notification" في حقل File name (اسم الملف)، 2 واضغط على Save (حفظ). 3



شكل 8.13: إنشاء مشروع جديد

ابدأ بإنشاء شبكة العُقدة بإضافة وحدة المُتحكم الرئيسة والوسطى:

لإدراج وحدة المُتحكم وعُقد الوسطى:

> اضغط على IoT Node (عُقدة إنترنت أشياء) من شريط Toolbar (الأدوات). 1

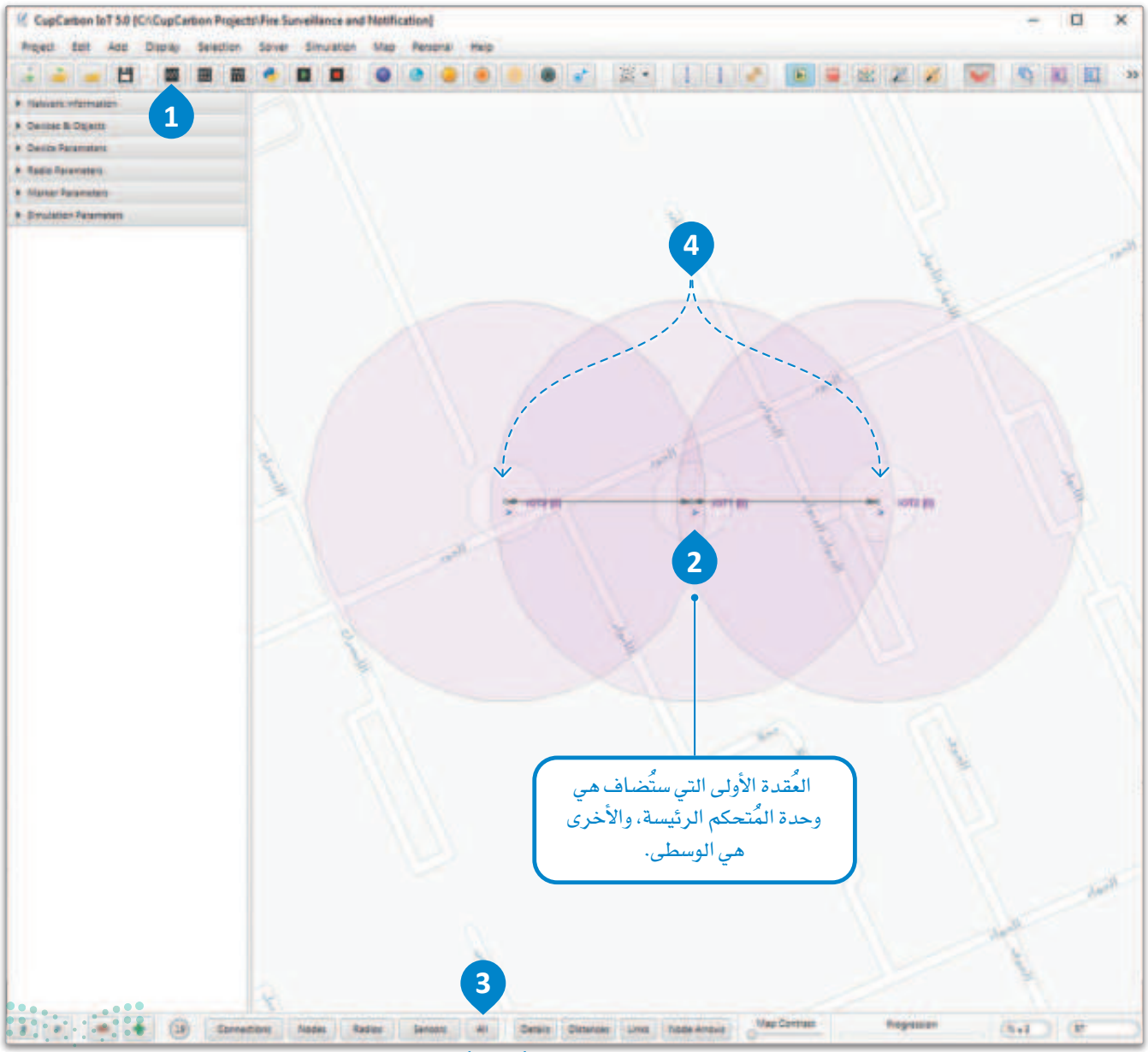
> اضغط على الخريطة لإضافة العُقدة. 2

> اضغط على All (الكل) من شريط State (الحالة). 3

> ضع عُقدتين أخريين على يسار ويمين العُقدة الأولى، وداخل دائرتها الخارجية. 4

> اضغط على Esc في لوحة المفاتيح.

إذا لم يتم وضع العُقد داخل نصف قطر وحدة المُتحكم، فلن تتمكن من الاتصال. وللتغلب على ذلك اسحبها وأفلتها بالقرب من وحدة المُتحكم حتى يظهر سهم ثنائي الاتجاه يربط بين العُقدتين.



شكل 8.14: إدراج وحدة المُتحكم وعُقد الوسطى

تابع بإضافة العُقد الطرفية:

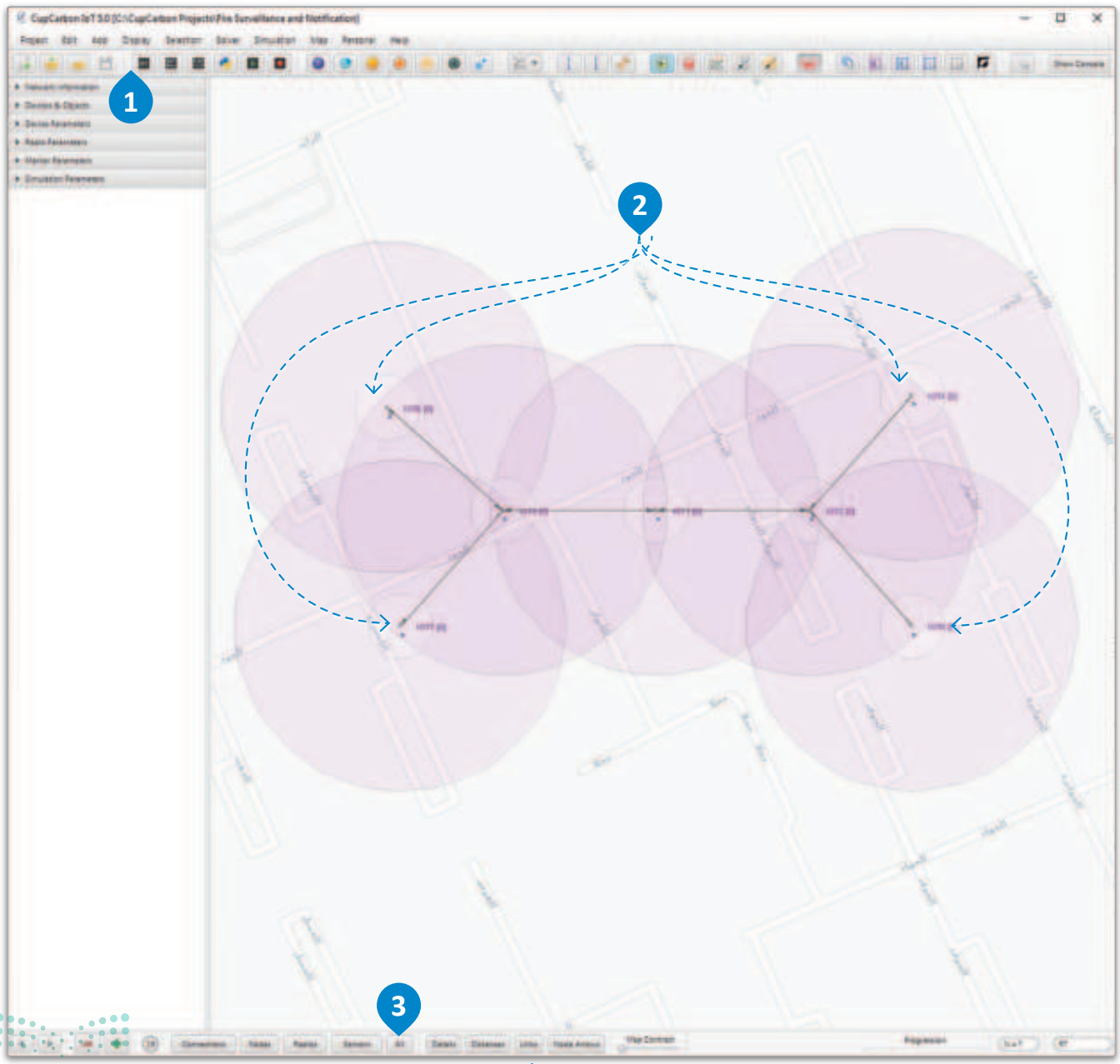
لإدراج العُقد الطرفية:

> اضغط على IoT Node (عُقدة إنترنت أشياء) من شريط Toolbar (الأدوات). 1

> ضع عُقدتين على كل proxy node (عُقدة الوسطى)، وذلك داخل دائرتها الخارجية، ولكن خارج نطاق أي عُقدة أخرى. 2

> اضغط على All (الكل) من شريط الحالة. 3

> اضغط على Esc في لوحة المفاتيح.



شكل 8.15: إدراج العُقد الطرفية

إنشاء المقاطع البرمجية Creating the Scripts

ستتعرف الآن على المقاطع البرمجية التي ستقوم بتشغيل العُقد. لنبدأ ببرمجة العُقد الطرفية.

في البداية، أضف المكتبات اللازمة.

```
import time
import random
```

تأخذ دالة توليد الأرقام العشرية ( )randint عددين صحيحين كوسيطين، وتعيد عددًا صحيحًا عشوائيًا داخل نطاق هذين العددين. على سبيل المثال، في الحالة السابقة ستُنشئ (1،6)randint عددًا صحيحًا بشكل عشوائي بقيمة بين 1 إلى 6. سيُستخدم الرقم ليمثل القطاع الذي قد يندلع فيه الحريق في كل فترة زمنية. وسيتم تخزين العدد الصحيح في متغير fire.

```
while node.loop():
  fire = random.randint(1, 6)
```

إذا افترضنا أن دالة ( )randint ستُرجع الرقم 1، فستكون النيران قد اندلعت افتراضيًا في هذا القطاع. سيتحقق البرنامج مما إذا كانت قيمة المتغير fire تساوي 1، وإذا كان الأمر كذلك، فسيتم تشغيل سلسلة من الأوامر بما فيها طباعة الرسالة "FIRE!" (حريق) على العُقدة نفسها. وإرسال رسالة تحتوي على مُعرف القطاع الخاص بها إلى العُقد الوسطى (proxy node) المجاورة لها.

إن مُعرِّف القطاع هو نفسه رقم مُعرِّف العُقدة، وهو عدد صحيح فريد. إذا كان مُعرِّف القطاع 5، فستكون الرسالة المُرسلة "FIRE IN SECTOR 5" (حريق في قطاع 5). يمكن إرجاع مُعرِّف العُقدة وبالتالي القطاع بواسطة الدالة ( )id. يتم إرجاع المُعرِّف كرقم، لذلك يجب "تحويله" إلى نوع نص قبل أن يُربط بالرسالة المتبقية.

```
if fire == 1:
  node.print("FIRE!")
  message = "FIRE IN SECTOR " + str(node.id())
```

يمكن للعُقد إرسال البيانات لبعضها باستخدام دالة ( )send. تستخدِم الدالة وسيطًا واحدًا فقط، وهو نص الرسالة الذي تقوم ببثه إلى جميع العُقد داخل نطاقها.

```
  node.send(message)
```

إذا أنتجت دالة توليد الأرقام العشوائية أي عددٍ صحيح آخر (في حالتنا أي رقم من 2 إلى 6)، فلا يوجد حريق في القطاع، ويتعين على العُقدة ببساطة طباعة نص ذاتي فارغ لمسح أي نص مطبوع سابقًا.

```
  else:
    node.print("")
```

في الختام، ستسكن العُقدة لفترة زمنية عشوائية، وذلك لمحاكاة عشوائية الأحداث في الحياة الواقعية. سيتحقق ذلك باستخدام الدالة ( )uniform التي تعمل مثل دالة ( )randint، ولكنها تُنتج أعدادًا حقيقية وليس فقط أعدادًا صحيحة. ستتراوح فترة السكون في مشروعك بين 1–4 ثوانٍ.

```
  time.sleep(random.uniform(1, 4))
```

المقطع البرمجي النهائي (edge.py)
Complete Code (edge.py)

```
import time
import random

while node.loop():

  fire = random.randint(1, 6)

  if fire == 1:
    node.print("FIRE!")
    message = "FIRE IN SECTOR " + str(node.id())
    node.send(message)
  else:
    node.print("")

  time.sleep(random.uniform(1, 4))
```

التالي هو المقطع البرمجي الخاص بالعُقد الوسطى.

عند استقبال العُقدة للبيانات، تُخزّن في **المخزن المؤقت** (Buffer) الخاص بها حتى قراءتها، ولذلك يجب التحقق من حجم المخزن المؤقت في البداية حيث يجب أن تكون قيمته أكبر من صفر (غير فارغ). يُمكن إرجاع حجم المخزن المؤقت بواسطة الدالة bufferSize().

```
import time

while node.loop():

  if node.bufferSize() > 0:
```

يُمكنك بعد ذلك قراءة البيانات المُستقبَلة باستخدام الدالة read(). بعد قراءة الرسالة، تُخزّن في رسالة المُتغير. تقوم العُقدة أيضًا بطباعة رسالة "FORWARDING..." لتوضح أنها تعيد توجيه الرسالة إلى وحدة المُتحكم الرئيسة.

```
    message = node.read()
    node.print("FORWARDING…")
```

كما هو الحال في البرنامج السابق، ستُرسل الرسالة المُخزنة في المتغير إلى وحدة المُتحكم الرئيسة باستخدام الدالة send(). وفي هذه المرة وبصرف النظر عن الرسالة، سيستلزم الأمر وسيطة إضافية وهي مُعرف العُقدة المُستقبلة (Node ID). نظرًا لكون الرسالة خاصة بعُقدة واحدة وبمُعرِّف مُحدد، فلا يلزم بث الرسالة، بل يمكن بدلًا من ذلك أن تكون أحادية الإرسال (أي تُرسل إلى عُقدة واحدة فقط). في الحالة السابقة، أُضيفت وحدة المُتحكم الرئيسة أولًا، وبالتالي يكون لها مُعرِّف مساويًا 1.

```
    node.send(message, 1)
```

بعد ذلك، سوف تسكن العُقدة لمدة ثانية واحدة لمنح المُستخدِم وقتًا كافيًا لقراءة الرسالة التوضيحية المطبوعة على العُقدة، ثم ستقوم العُقدة بمسح الرسالة وذلك بطباعة نص فارغ.

```
    time.sleep(1)
    node.print("")
```

ينتهي المقطع البرمجي بسكون العُقدة لفترة زمنية صغيرة جدًا (جزء من مائة من الثانية)، مما يمنحها القدرة على الاستجابة في حالة تلقيها الكثير من البيانات.

```
  time.sleep(0.01)
```

المقطع البرمجي النهائي (proxy.py)
Complete code (proxy.py)

```
import time
while node.loop():

  if node.bufferSize() > 0:
    message = node.read()
    node.print("FORWARDING…")
    node.send(message, 1)
    time.sleep(1)
    node.print("")

  time.sleep(0.01)
```

مُعامِل الرقم في دالة send( ) هو رقم مُعرِّف عُقدة وحدة المُتحكم.

يتشابه المقطع البرمجي لوحدة المُتحكم نوعًا ما مع برامج العُقد الوسطى، فهو يفحص المخزن المؤقت، ويقرأ الرسالة المُستلمة، ولكن الرسالة الذاتية المطبوعة هي نفس الرسالة التي أُنشئت في الأصل بواسطة **عُقدة الطرفية** (Edge Node).

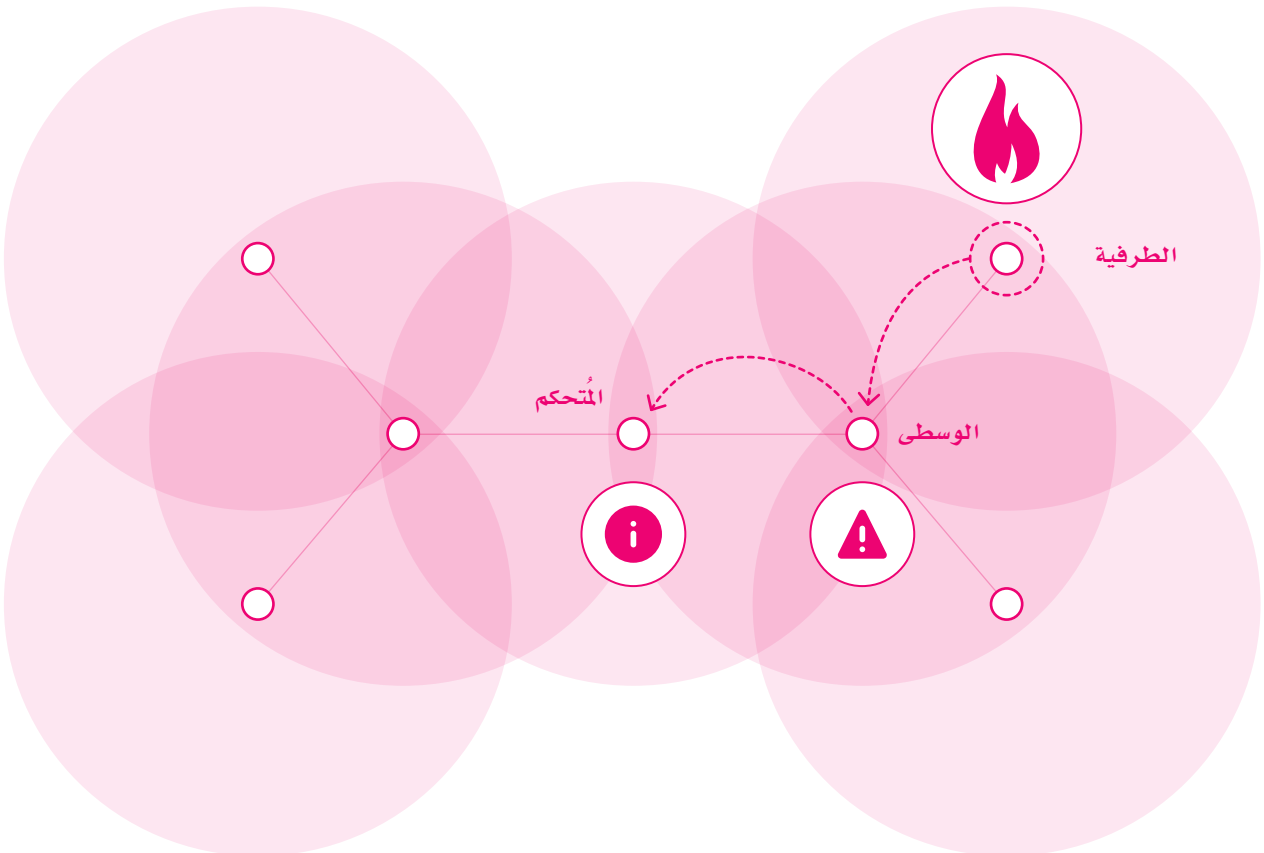
```
import time
while node.loop():

  if node.bufferSize() > 0:
    message = node.read()
    node.print(message)
```

بعد ذلك وكما حدث في عُقد الوسطى، ستسكُن وحدة المُتحكم، ولكن لمدة ثانيتين هذه المرة، ثم تطبع نصًا فارغًا.
وفي الختام ستسكُن لفترة قصيرة بنفس الطريقة التي حدثت مع العُقد الوسطى.

```
    time.sleep(2)
    node.print("")

  time.sleep(0.01)
```

المقطع البرمجي النهائي (controller.py)
Complete code (controller.py)

```
import time
while node.loop():

  if node.bufferSize() > 0:
    message = node.read()
    node.print(message)
    time.sleep(2)
    node.print("")

  time.sleep(0.01)
```



شكل 8.16: الشبكة بشكلها النهائي

الآن وبعد أن تعرَّفت على وظيفة المقاطع البرمجية، تابع عملك وقم بإنشائها.

لإنشاء البرنامج وتطبيقه على عُقدة وحدة المُتحكم:

لإنشاء المقطع البرمجي:

- اضغط على بايثون في شريط الأدوات. 1
- اكتب أوامر بايثون البرمجية في الحقل الفارغ. 2
- اكتب controller (وحدة المُتحكم) في حقل File name (اسم الملف). 3
- اضغط على Save (حفظ). 4
- أغلق نافذة محرر بايثون النصي. 5

شكل 8.17: إنشاء المقطع البرمجي

لإدراج البرنامج:

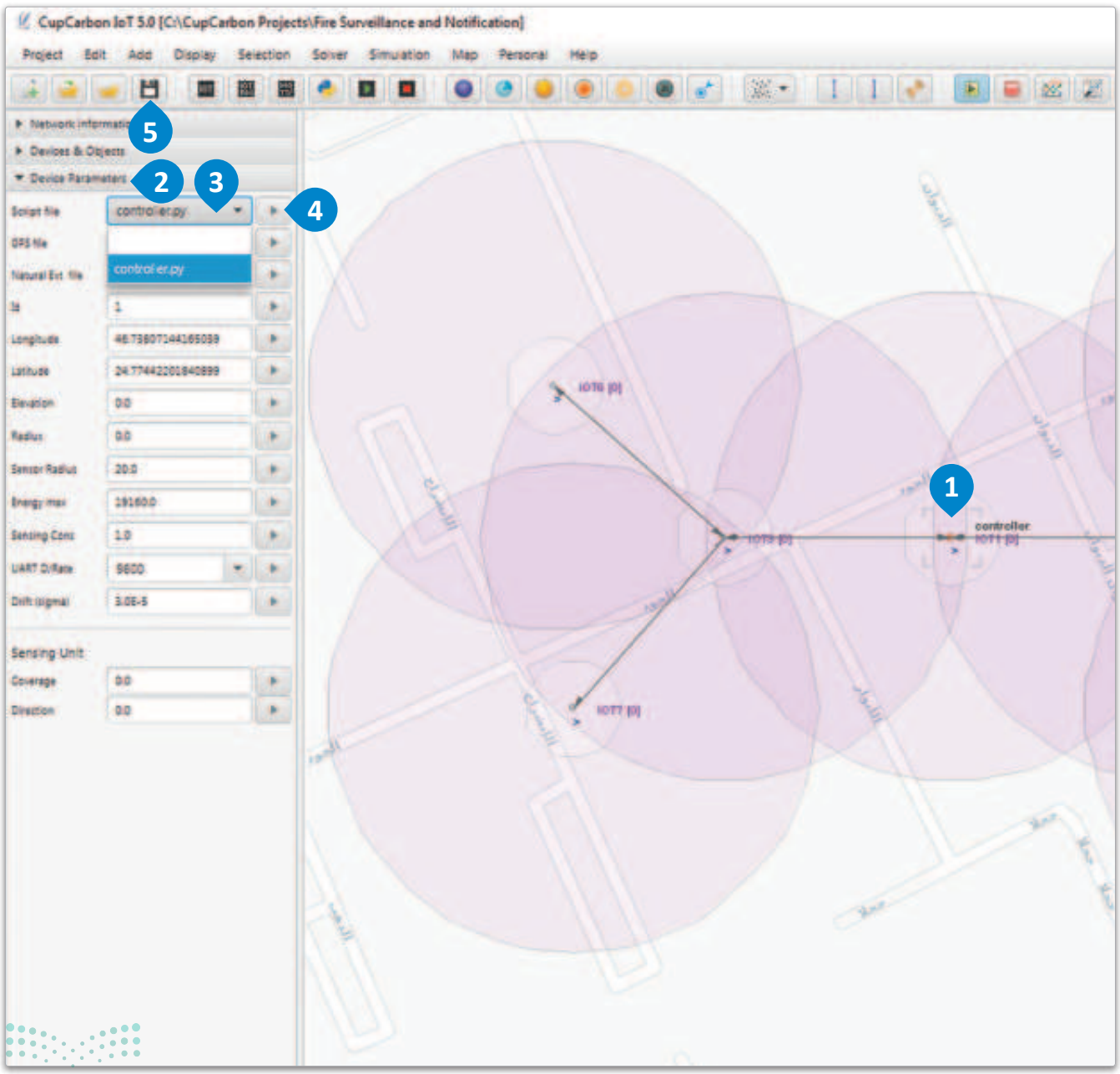
> اضغط على العُقدة. 1

> اضغط على علامة تبويب Device Parameters (مُعامِلات الجهاز) في قائمة Parameters (المُعامِلات). 2

> اضغط على صندوق Script file (ملف المقطع البرمجي). 3

> من القائمة المنسدلة، اختر ملف controller.py واضغط على الزر الموجود على اليمين لإدراج المقطع البرمجي في العُقدة. 4

> اضغط على Save Project (حفظ المشروع) من Toolbar (شريط الأدوات). 5

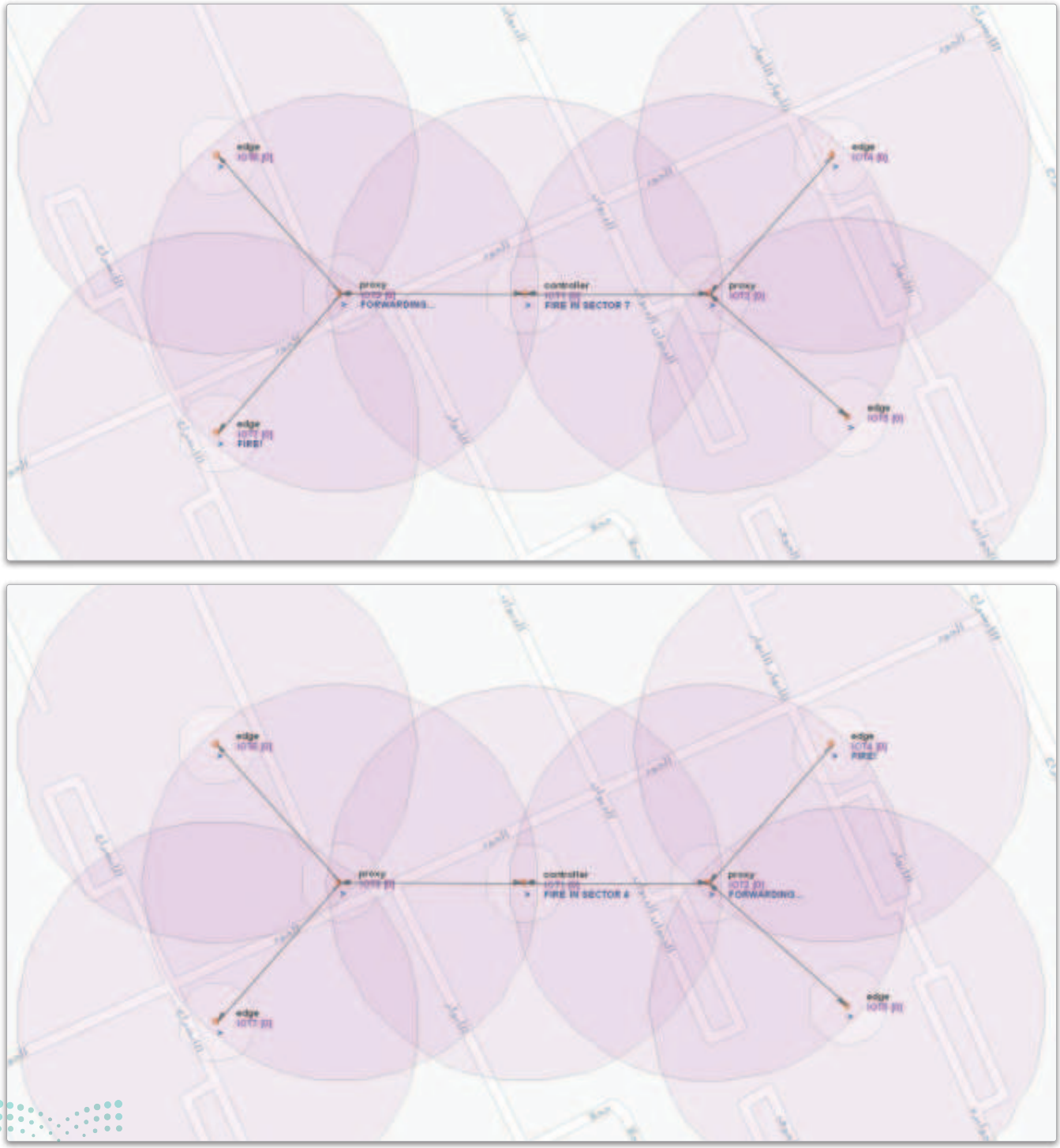


شكل 8.18: إضافة المقطع البرمجي

أنشئ المقاطع البرمجية الأخرى بنفس الطريقة، وانسخ أوامرها وطبقّها على العُقد المقابلة لها، بحيث تحتوي جميع العُقد على المقطع البرمجي. عند الانتهاء، اضغط على Run IoT Simulation (تشغيل محاكاة إنترنت الأشياء) من شريط الأدوات.

لاحظ أنه نظرًا لاستخدامك مُولدات أرقام عشوائية، فقد تشتعل حرائق في بعض القطاعات الموجودة على الأطراف أكثر من غيرها والتي قد لا تشتعل فيها حرائق على الإطلاق.

استخدم أسماءً نصية مُعبرة وواضحة للمقاطع البرمجية مثل edge.py و proxy.py.



شكل 8.19: حالات المحاكاة

تمرينات

1. وسِّع مشروعك لدعم عُقدة طرفية (Edge) تضاف لكل عُقدة وسطى (Proxy)، بحيث يكون لكل عُقدة وسطى ثلاثة عُقد طرفية. لا تنْسَ إضافة المقاطع البرمجية داخل العُقد الجديدة.

2. وسِّع مشروعك لدعم عُقدة وسطى إضافية، وأضف عقدتين طرفيتين جديدتين إلى الوسطى، بحيث يكون لدى وحدة المُتحكم الرئيسة ثلاث عُقد وسطى، ولكل عُقدة وسطى عُقدتين طرفيتين. لا تنْسَ إضافة المقاطع البرمجية داخل العُقد الجديدة.

3. حدِّد أي قسم من التعليمات البرمجية يُقرر تكرار حدوث الحرائق. عدِّل مشروعك في برنامج كاب كاربون (CupCarbon) لزيادة احتمال حدوث الحرائق أكثر من السابق.

4. قد يؤدي أي تأخير زمني (Latency) في شبكة المصنع إلى تأخير الاتصال بين العُقد. قم بتعديل برنامجك الخاص بعُقد الوسطى لجعل العُقد في وضع السكون لفترة أطول. هل لاحظت وجود أي تأخير أو فقدان لأي رسائل؟ دوِّن ملاحظاتك أدناه.

5. وسِّع مشروعك ليدعم احتمال حدوث تسرب المياه وحدوث الفيضان. عدِّل برنامجك للقطاعات المُعرضة للحرائق، بحيث يعني إرجاع القيمة من دالة توليد الأرقام العشوائية ( ) randint القيمة 2 حدوث تسرب للمياه أو فيضان في هذا القطاع. على العُقدة القيام بطباعة الرسالة المناسبة وإرسالها.

رابط الدرس الرقمي

www.ien.edu.sa

الدرس الثالث

إنترنت الأشياء والأجهزة المحمولة المؤتمتة

الصناعة الذكية والأتمتة Smart Industry and Automation

تُعدُّ الأتمتة ميزة مهمة للتقنية الحديثة، وكذلك فهي عاملٌ مُساهمٌ بشكلٍ رئيس في الثورة الصناعية الرابعة. تُعزّز الصناعة الذكية من خلال تقنيات الأتمتة التي تزيد من الإنتاجية، مما يُتيح تحقيق المزيد من الأرباح. ستُنشئ في المشروع الآتي محاكاة لنظام يفحص منطقة تخزين المصنع للحاويات التي تحتوي على مواد قابلة للتلف إذا تُركت دون تبريد طوال الليل، وذلك باستخدام مركبة تفتيش آلية.

ستتخذ مركبة التفتيش الآلية مسارًا مُحددًا سابقًا في منطقة تخزين المصنع، وستُوضع علامات على الحاويات وفقًا لمحتوياتها من مواد قابلة للتّلف، ومواد طويلة الأمد لا تحتاج إلى التبريد. تحتوي كل حاوية على **رقاقة إنترنت الأشياء** (IoT Tag) تُرسل رسالة باستخدام موجاتها اللاسلكية لتُبلغ المركبة الآلية بمحتوياتها. توجد أيضًا بعض محطات الشحن في كافة أنحاء منطقة التخزين لشحن بطارية المركبة التي ستنخفض أثناء حركة المركبة.

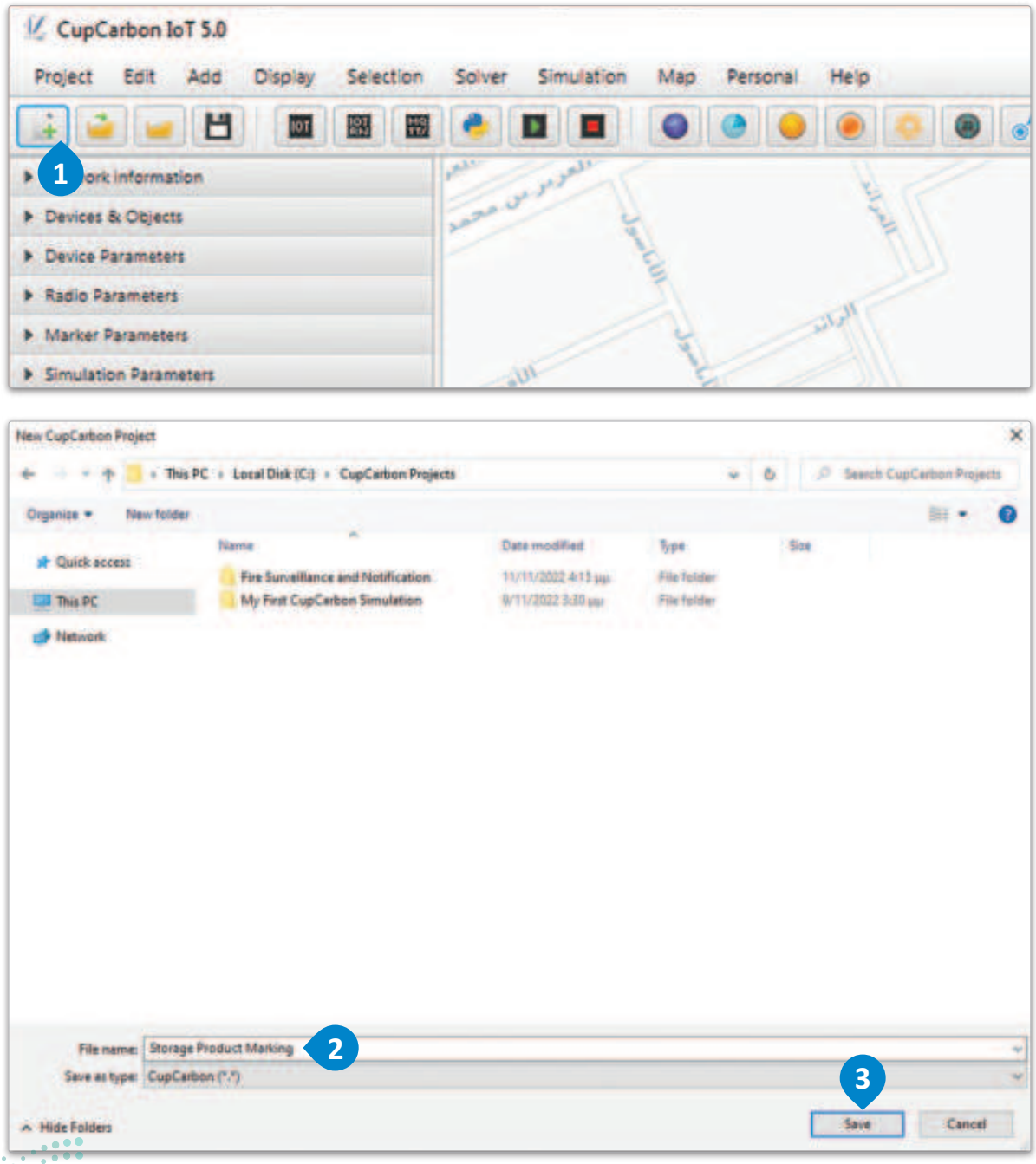
شكل 8.20: مركبة صناعية آلية

لنبدأ بإنشاء مشروع جديد:

لإنشاء مشروع جديد:

> اضغط على New Project (مشروع جديد) من Toolbar (شريط الأدوات). 1

> اختر الموقع الذي تريده لحفظ المشروع، اكتب "Storage Product Marking" في حقل File name (اسم الملف)، 2 واضغط على Save (حفظ). 3



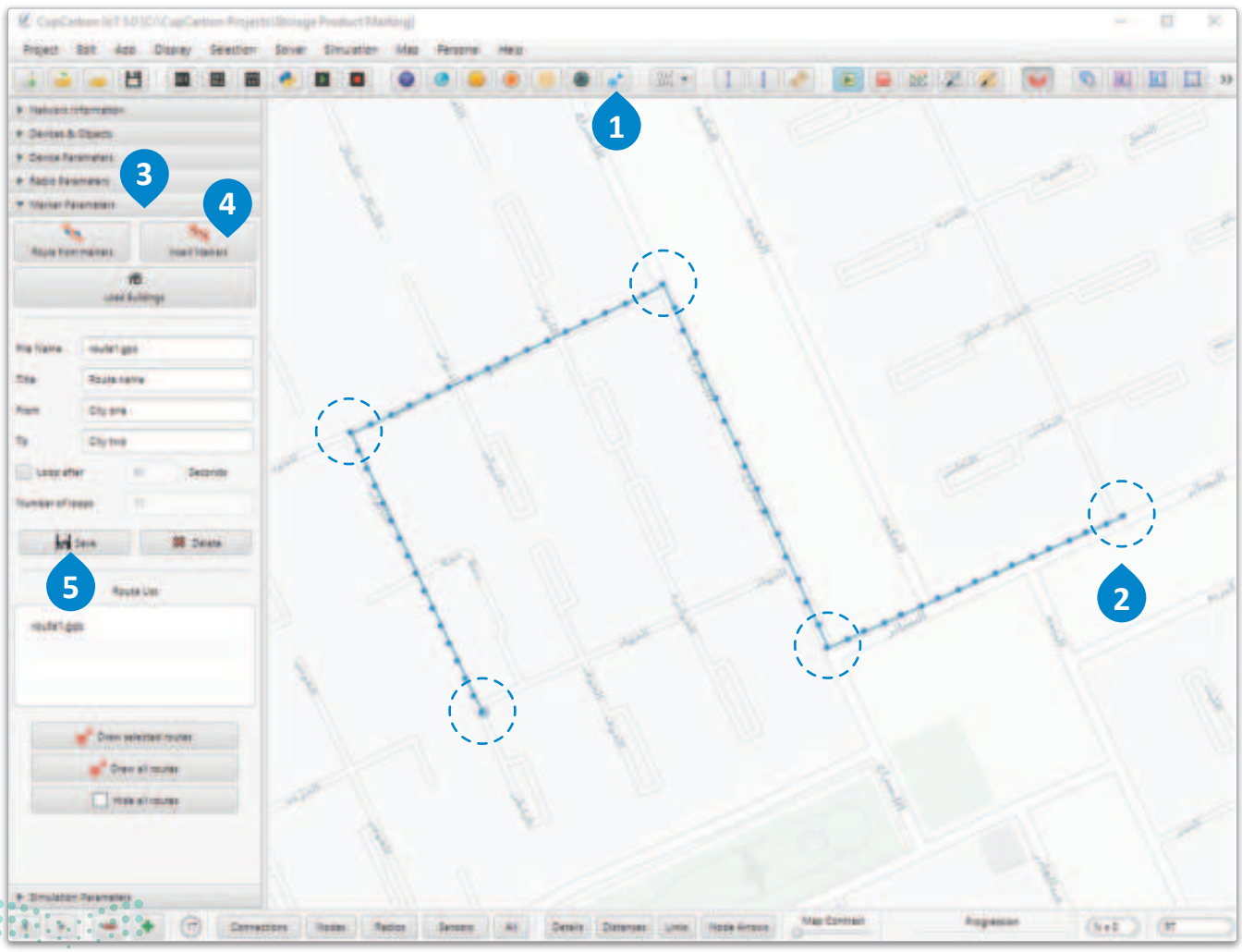
شكل 8.21: إنشاء مشروع جديد

إنشاء مسار مُحدد سابقًا Creating a Predetermined Route

في البداية، ستنشئ المسار الذي ستسير عليه مركبة التفتيش. أولًا، ستضع بعض العلامات على الخريطة لتحديد طبيعة المسار، ثم ستقوم بإضافة بعض العلامات الأخرى لتحديد المسار بشكل دقيق.

لإنشاء المسار:

- اضغط زر Marker (علامة) من Toolbar (شريط الأدوات). 1
- اضغط على الخريطة 5 مرات كما يظهر في الصورة أدناه، مما يُنشئ خطوطًا على طول المسارات. 2
- اضغط على Esc في لوحة المفاتيح.
- اضغط على علامة تبويب Marker Parameters (مُعامِلات العلامات) في قائمة Parameters (المُعامِلات). 3
- أضف المزيد من العلامات إلى المسار بالضغط على كل علامة من العلامات الأربعة المضافة، والضغط على Insert Markers (إدراج العلامات) 4 مرات. 4
- اضغط على Save (حفظ). 5



شكل 8.22: إنشاء مسار

إضافة عُقدة مركبة التفتيش Adding the Inspector Vehicle Node

ستتم محاكاة مركبة التفتيش بواسطة عُقدة تتحرك على طول المسار الذي أنشأته سابقًا. ستقوم أيضًا بزيادة نطاق الاستشعار للعُقدة (الدائرة الداخلية) بحيث يمكنها الوصول إلى نطاق محطة الشحن.

لإضافة عُقدة مركبة التفتيش:

> اضغط على IoT Node (عُقدة إنترنت الأشياء) من شريط الأدوات. 1

> اضغط على الخريطة لإضافة العُقدة. 2

> اضغط على All (الكل) من State bar (شريط الحالة). 3

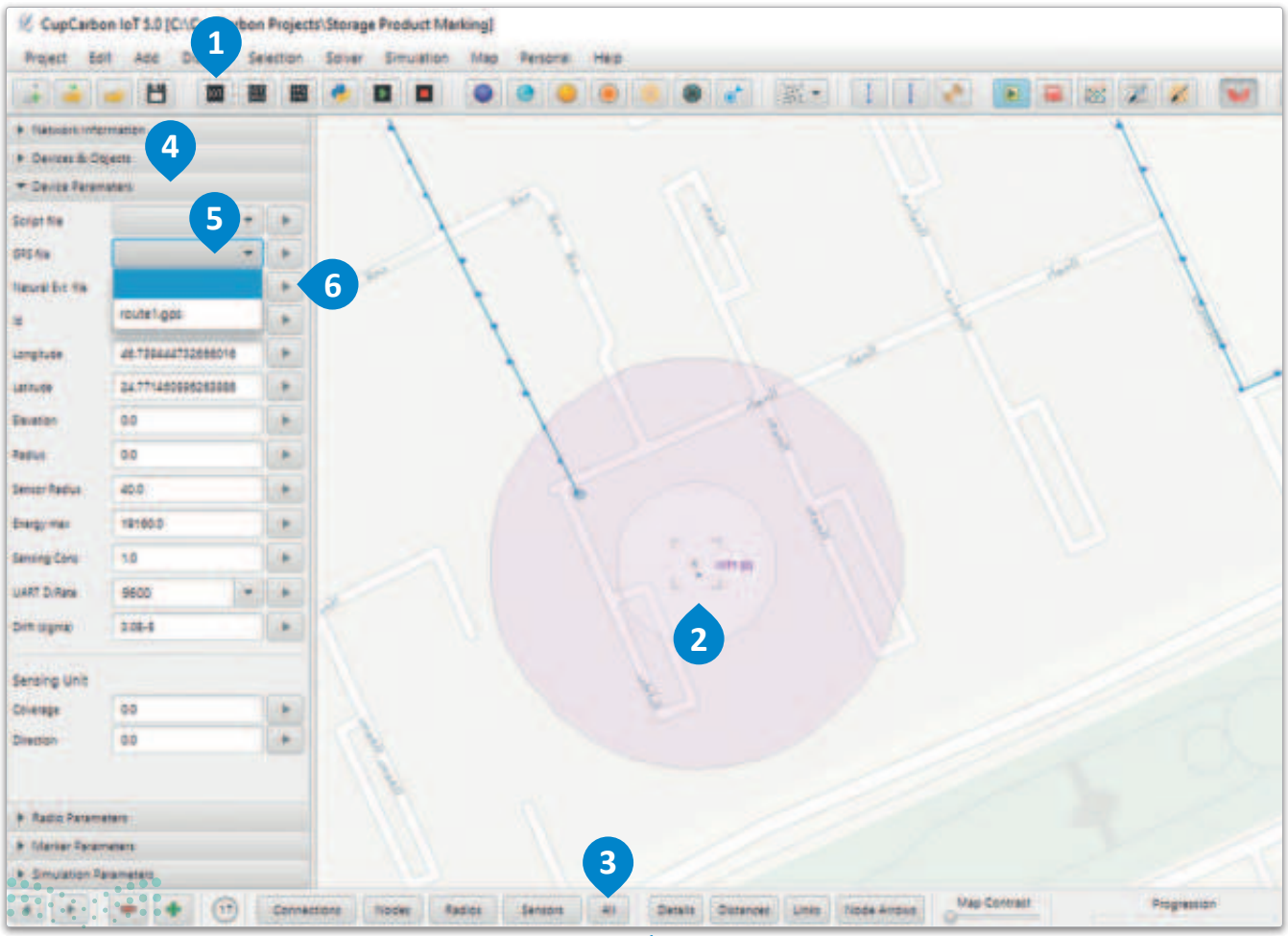
> اضغط على Esc في لوحة المفاتيح.

> اضغط على العُقدة واضغط على الزرين Shift ⇧ + 0 معًا أربع مرات لزيادة نطاق الاستشعار.

> اضغط على علامة التبويب Device Parameters (مُعامِلات الجهاز) في قائمة parameter (المُعامِلات). 4

> اضغط على الصندوق الموجود على يمين ملف GPS. 5

> من القائمة المنسدلة، حدِّد برنامج route1.gps واضغط على الزر الموجود على اليمين لإدراج المسار في العُقدة. 6



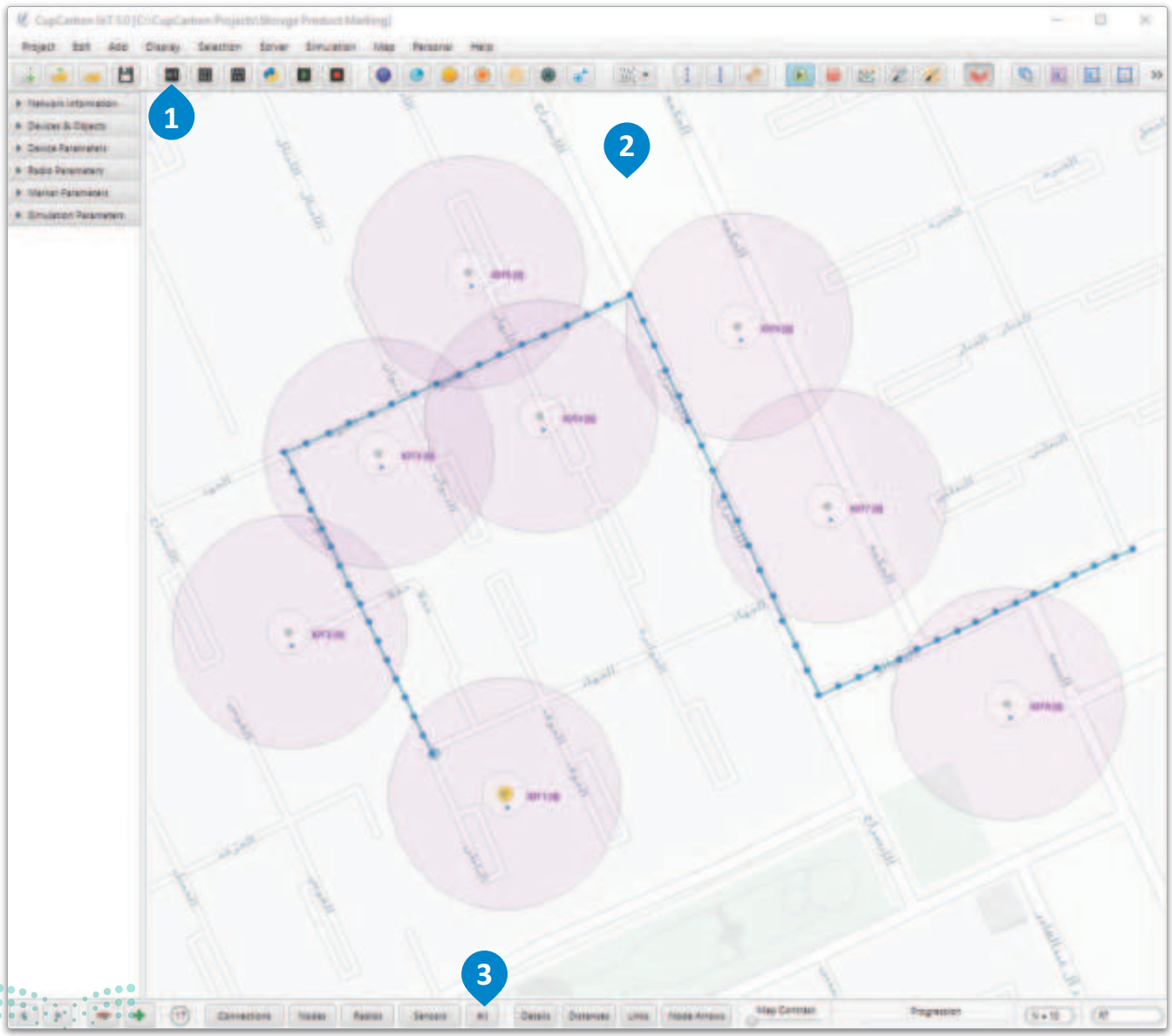
شكل 8.23: إنشاء عُقدة مركبة التفتيش

إضافة عُقد الحاوية Adding Container Nodes

حان الوقت الآن لإضافة العُقد التي تُمثل الحاويات.

لإضافة عُقد الحاوية،

- اضغط على IoT Node (عُقدة إنترنت الأشياء) من شريط الأدوات. 1
- اضغط على الخريطة وأضف 7 عُقد بالقرب من المسار، بحيث يتضمن نصف قطر كل منها علامة واحدة من المسار على الأقل. 2
- اضغط على All (الكل) من شريط الحالة. 3
- اضغط على زر Esc في لوحة المفاتيح.



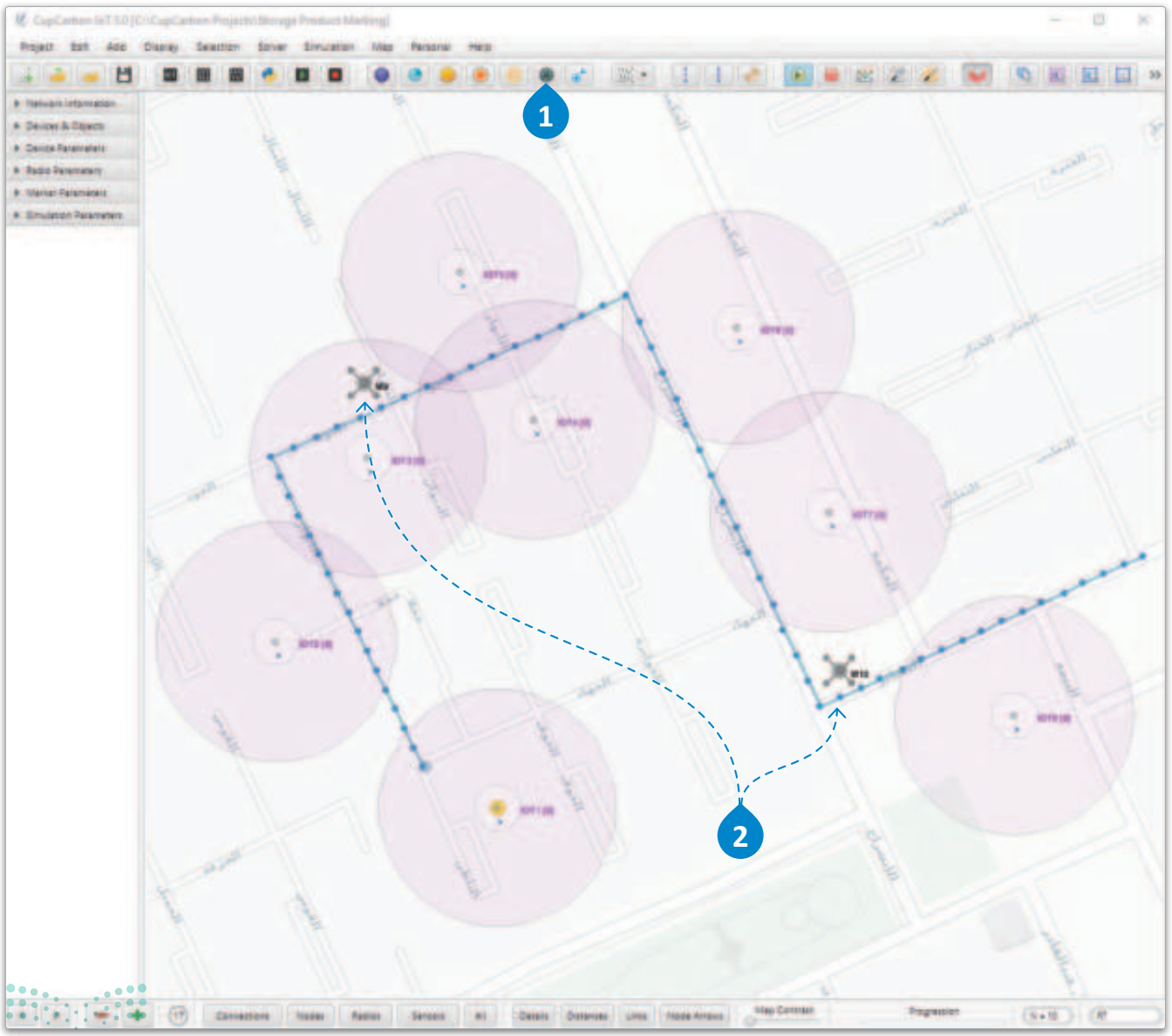
شكل 8.24: إنشاء عُقد الحاوية

إضافة محطات الشحن Adding Charging Stations

تستهلك مركبة التفتيش الطاقة خلال حركتها في المخزن، مما يتطلب إعادة شحنها. ستقوم بإضافة بعض محطات الشحن على المسار لشحن المركبة أثناء مرورها بقربها، ولهذا الغرض سيتم استخدام نطاق الاستشعار الداخلي.

لإضافة نقاط لمحطات الشحن:

- اضغط على Mobile (الهاتف المحمول) من Toolbar (شريط الأدوات). 1
- اضغط على map (الخريطة) وأضف عُقدتين على طول المسار بحيث يُمكنهما استشعار المركبة. 2
- اضغط على زر Esc في لوحة المفاتيح.



شكل 8.25: إنشاء عُقد محطة الشحن

إنشاء المقاطع البرمجية Creating the Scripts

ستلقي الآن نظرة على كافة المقاطع البرمجية التي ستستخدمها بِدءًا من برمجة الحاويات. مع العلم بأن برمجة النوعين المختلفين من الحاويات ستكون متطابقة.

ابدأ بإضافة المكتبة اللازمة، وطباعة نص فارغ على العُقدة لإزالة أي نصوص مطبوعة من عمليات التنفيذ السابقة.

```
import time


node.print("")
```

ستقوم حاويات المواد القابلة للتلف ببث رسالة تتضمن محتوياتها ومُعرِّفها ليُمكن استخدامها من قِبل مركبة التفتيش لتصنيف كل حاوية. المُعرِّف (ID) عدد صحيح، ويجب تحويل نوعه إلى مُتغير نصي قبل ربطه بالنص. تُوضع مسافة بين معلومات المحتويات والمُعرِّف بحيث يمكن إرسال نصٍ واحدٍ فقط في آنٍ واحد باستخدام دالة ()send للإرسال، ثم يتعين عليك إرسال جزئيتين من المعلومات يُفصل بينهما بواسطة المسافة.

```
while node.loop():


    node.send("CONSUMABLES " + str(node.id()))
```

بعد تحليل الحاوية للنص المُستقبل، ستُرسل إما "1" والتي تعني أنه يجب اختيارها، أو ستُرسل "2" والتي تعني أنه يجب ألا يتم ذلك. بدورها، ستطبع الحاوية النص الذاتي "PICK" (التقط) أو "DO NOT PICK" (لا تلتقط)، ثم ستسكُن لمدة ثانية واحدة.

```
    message = node.read()
    if message == "1":
        node.print("PICK")
    elif message == "2":
        node.print("DO NOT PICK")


    time.sleep(1)
```

المقطع البرمجي النهائي (consumples.py)
Complete Code (consumables.py)

```
import time

node.print("")
while node.loop():

  node.send("CONSUMABLES " + str(node.id()))
  message = node.read()
  if message == "1":
    node.print("PICK")
  elif message == "2":
    node.print("DO NOT PICK")

  time.sleep(1)
```

وجه الاختلاف بين البرمجة الخاصة بالمواد القابلة للتلف والمواد طويلة الأمد هو النص المُرسل، ففي المقطع البرمجي الخاص بالمواد طويلة الأمد سيتغير النص من "CONSUMABLES" (قابلة للتلف) إلى "NONCONSUMABLES" (طويلة الأمد).

المقطع البرمجي النهائي (nonconsumables.py)
Complete Code (nonconsumables.py)

```
import time

node.print("")
while node.loop():

  node.send("NONCONSUMABLES " + str(node.id()))
  message = node.read()
  if message == "1":
    node.print("PICK")
  elif message == "2":
    node.print("DO NOT PICK")

  time.sleep(1)
```

هنا المقطع البرمجي الخاص بمركبة التفتيش. في البداية ستُهيّأ البطارية بضبط أقصى طاقة لها لتعادل 100 وحدة طاقة باستخدام الدالة ()battery.setEMax، ثم ضبط مستواها الحالي إلى الحد الأقصى مع دالة ()battery.init.

```
import time


node.battery.setEMax(100.0)
node.battery.init()
```

ستستهلك المركبة بمرور الزمن قدرًا مُعينًا من الطاقة. ولمحاكاة ذلك، استخدم الدالة battery.consume(1.0) لتنفيذ استهلاك وحدة طاقة لكل فترة زمنية محددة.

```
while node.loop():


  node.battery.consume(1.0)
```

لاكتشاف ما إذا كانت أي محطة شحن موجودة في نطاق المركبة، استخدم الدالة ()isSensorDetecting، وعند اكتشاف محطة، استخدم ()battery.init لشحنها إلى الحد الأقصى.

```
  if node.isSensorDetecting():
    node.battery.init()
```

يتعين على المركبة التحقق الآن من جميع الرسائل التي استقبلتها، ثم الرد على مُرسليها (الحاويات). سيتم في البداية تخزين مُتغير القراءة المحلي في recvMsg، ثم باستخدام دالة ()split سيُفصل النص إلى جزأين وفقًا للمساحة المُستخدمة سابقًا، على شكل مصفوفة باسم splitMsg. وهذا يعني أنه في الخلية الأولى من المصفوفة splitMsg [0] سيُحتفظ بمحتويات الحاوية، بينما تحتفظ الخلية الثانية splitMsg [1] بمُعرِّف الحاوية.

```
  for n in range(node.bufferSize()):
    recvMsg = node.read()
    splitMsg = recvMsg.split()
```

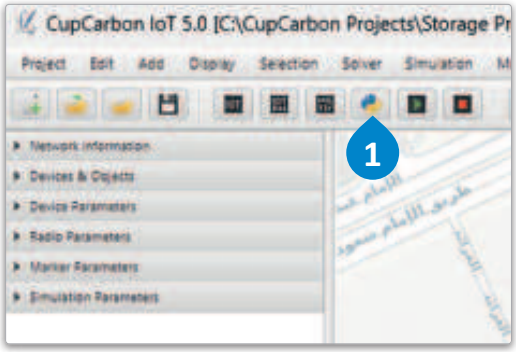
إذا كان نص المحتوى "CONSUMABLES"، فستُرسل النص "1" بواسطة دالة ()send إلى حاوية المُرسل باستخدام مُعرِّفها، أما إذا كان نص المحتوى "NONCONSUMABLES" فسيُرسل النص "2". وفي الختام ستسكُن لمدة 200 مللي ثانية؛ لأنها تحتاج إلى تحقيق استجابة أكثر من عُقد الحاوية بصفتها تتواصل مع المزيد من العُقد.

```
  if splitMsg[0] == "CONSUMABLES":
    node.send("1", int(splitMsg[1]))
  elif splitMsg[0] == "NONCONSUMABLES":
    node.send("2", int(splitMsg[1]))

  time.sleep(0.2)
```

المقطع البرمجي النهائي (inspector.py)

Complete Code (inspector.py)

```
import time

node.battery.setEMax(100.0)
node.battery.init()

while node.loop():

  node.battery.consume(1.0)

  if node.isSensorDetecting():
    node.battery.init()

  for n in range(node.bufferSize()):
    recvMsg = node.read()
    splitMsg = recvMsg.split()
    if splitMsg[0] == "CONSUMABLES":
      node.send("1", int(splitMsg[1]))
    elif splitMsg[0] == "NONCONSUMABLES":
      node.send("2", int(splitMsg[1]))

  time.sleep(0.2)
```



لإنشاء المقطع البرمجي:

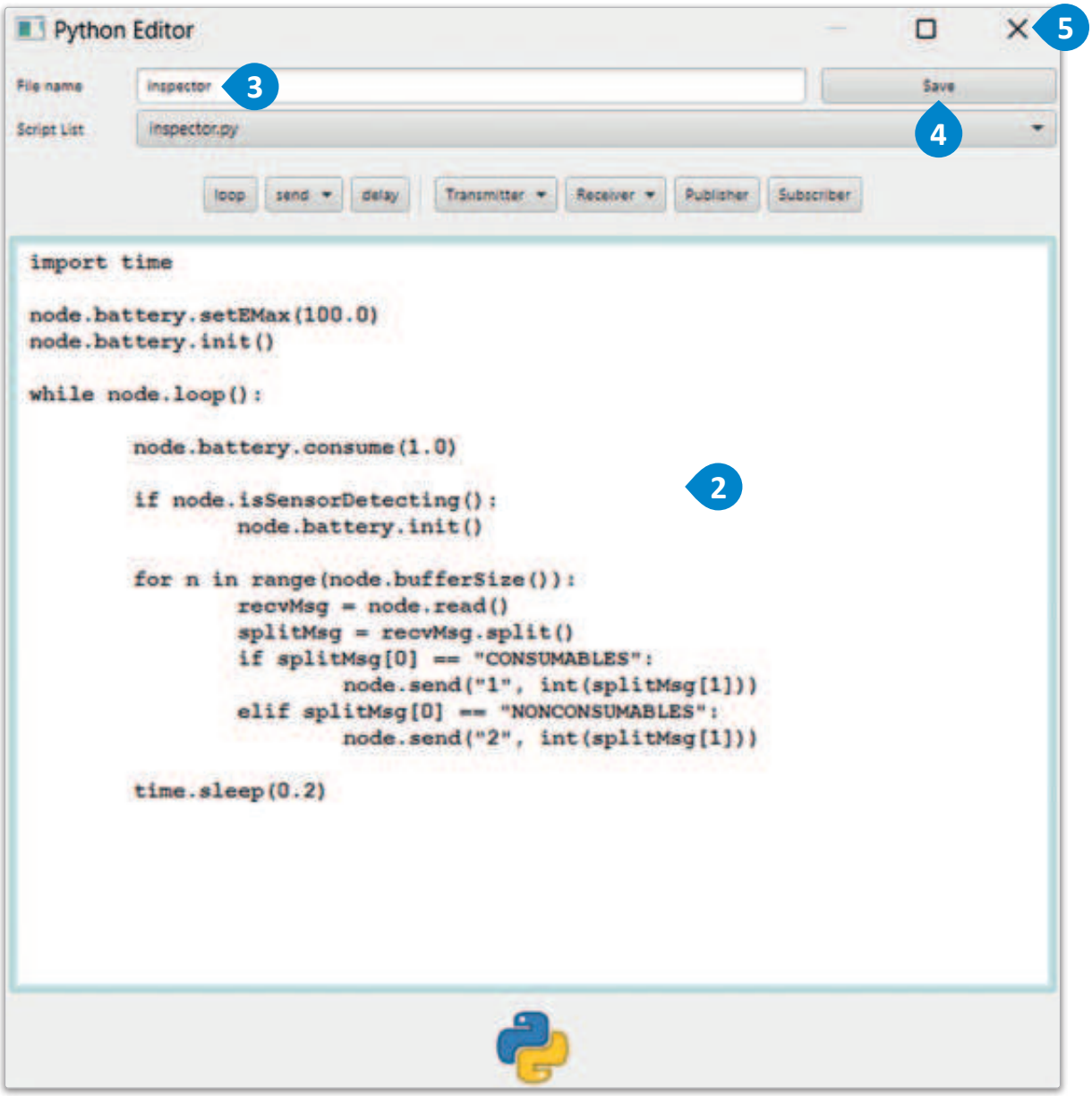
> اضغط على Python (بايثون) من شريط الأدوات. 1

> اكتب التعليمات البرمجية في الحقل النصي. 2

> في حقل File name (اسم الملف)، اكتب inspector. 3

> اضغط على Save (حفظ). 4

> أغلق نافذة محرر بايثون النصي. 5



شكل 8.26: إنشاء المقطع البرمجي

لإدراج المقطع البرمجي:

> اضغط على عُقدة مركبة التفتيش. 1

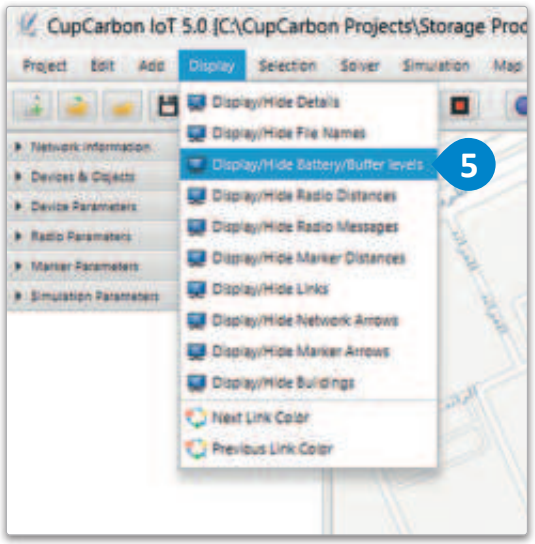
> اضغط على علامة تبويب Device Parameters (مُعامِلات الجهاز) في قائمة Parameters (المُعامِلات). 2

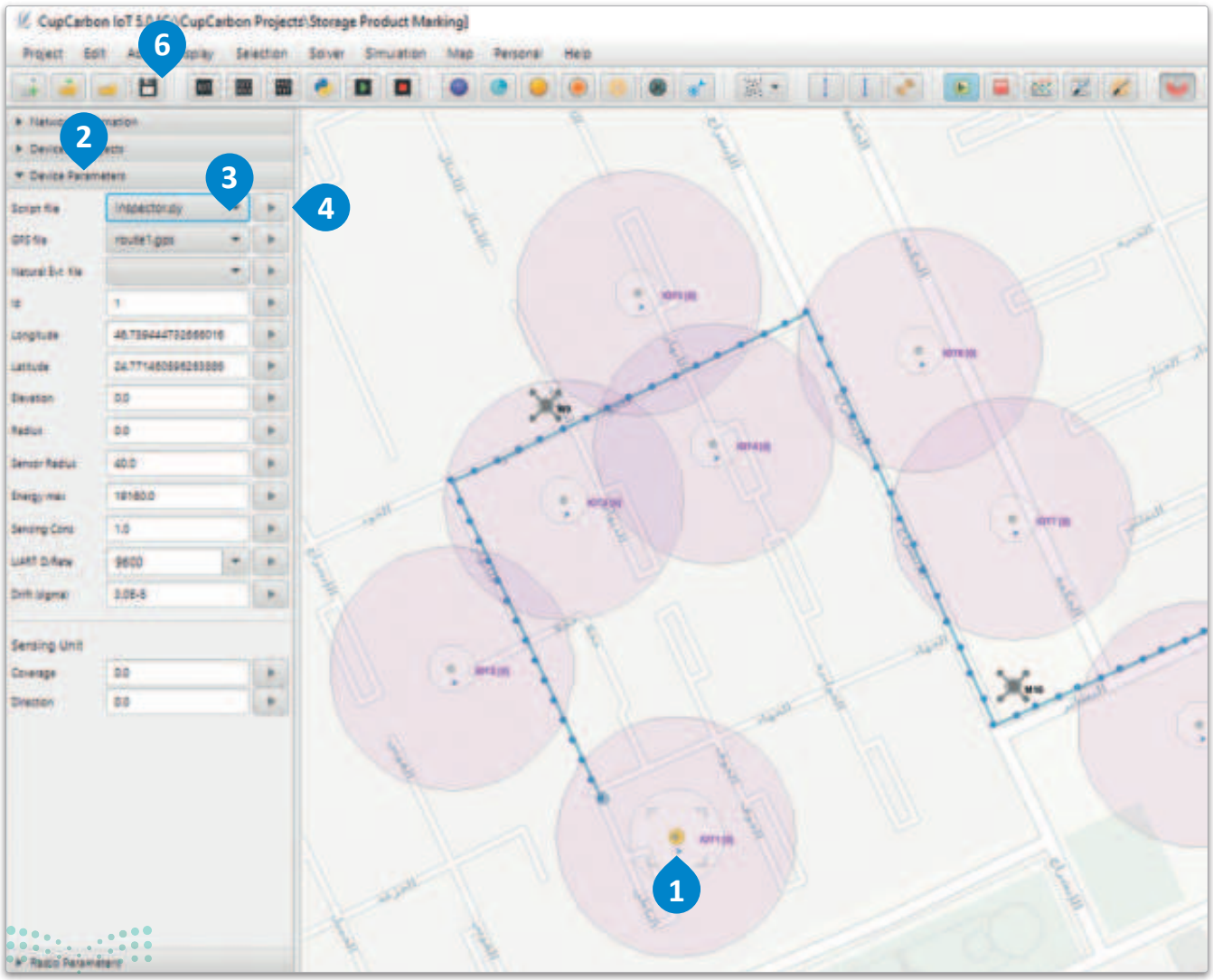
> اضغط صندوق Script file (ملف المقطع البرمجي). 3

> من القائمة المنسدلة، اختر Inspector.py واضغط على الزر الموجود على اليمين لإدراج المقطع البرمجي في العُقدة. 4

> اضغط على Display > Display/Hide Battery/Buffer levels (عرض> عرض / إخفاء مستويات البطارية / المخزن المؤقت)، من شريط Menu (القوائم). 5

> اضغط على Save Project (حفظ المشروع) من Toolbar (شريط الأدوات). 6





شكل 8.27: إدراج المقطع البرمجي

أنشئ المقاطع البرمجية consumables.py و nonconsumables.py بنفس الطريقة، وطبِّق المقطع البرمجي الأول على بعض عُقد الحاويات، والثاني على بقيتها، بحيث تحتوي جميع عُقد الحاوية على أحد هذين المقطعين.

عند الانتهاء، يمكنك الضغط على زر Run IoT Simulation (تشغيل محاكاة إنترنت الأشياء) من شريط الأدوات لبدء المحاكاة.

شكل 8.28: تشغيل المحاكاة

تمرينات

1. وسِّع مشروعك بإضافة المزيد من العُقد وإنشاء مسار بالمزيد من العلامات.

   لا تنْسَ إضافة المقاطع البرمجية في العُقد الجديدة.

2. حدِّد ما إذا كان مشروعك يستخدِم أقل عدد ممكن من مُحطات الشحن. حاول إزالة محطة، ونقل الأخريات لاختبار فرضيتك.

3. عدِّل البرمجة الخاصة بمركبة التفتيش لكي تستهلك المزيد من الطاقة، ولكي تُستنزف بطاريتها بشكل أسرع. دوِّن نتائجك أدناه.

4. وسِّع مشروعك عن طريق إنشاء نوعٍ ثالثٍ من عُقد الحاوية وهو حاوية فارغة ستُرسل النص Empty (فارغة)، ولن تُحدّد بواسطة مركبة التفتيش.

5. قد يكون لبطء اتصال شبكة المصنع آثار خطيرة على وظائف النظام. عدِّل البرمجة الخاصة بعُقدة مركبة التفتيش لجعل العُقدة تسكُن لمدة أطول. هل حدث تأخير أو فقدان لأي رسائل؟ دوِّن ملاحظاتك أدناه.

المشروع

يُعدُّ الاتصال مهمًا جدًا داخل المصنع للتنسيق بين الإدارات والقطاعات المختلفة. عندما يُطبّق الاتصال المتوازي ببراعة، يمكن زيادة الكفاءة والإنتاجية بشكل كبير.

1 ستقوم بمحاكاة نظام توصيل داخل المصنع يتكون من مركبة توصيل تتحرك على مسار مُحدد سابقًا بحيث تُوصل الأجزاء والمواد إلى قطاعات مختلفة. أنشئ شبكة تتكون من وحدة المُتحكم الرئيسة مع 3 عُقد وسطى و3 عُقد طرفية لكل عُقدة وسطى.

2 سوف تمر مركبة التوصيل على كل عُقدة طرفية وتوصل أي أجزاء أو مواد. اكتب مقطعًا برمجيًا للعُقد الطرفية لطلب الأجزاء أو المواد عن طريق إرسال نص بهذا الطلب إلى المركبة، واكتب مقطعًا برمجيًا لتزويد المركبة بما طُلب بإعطاء تأكيد.

3 وسِّع مشروعك بحيث تقوم العُقد الطرفية بعد تلقي طلبها بإعادة توجيه الرسالة إلى العُقد الوسطى المقابلة لها بحيث يمكن متابعة الإنتاج. وفي المقابل، ستقوم العُقد الوسطى بإعادة توجيه الرسالة إلى وحدة المُتحكم الرئيسة، والتي ستطبع رسالة إعلامية تفيد بأن طلب القطاع قد تم تلبيته.

4 وسِّع مشروعك ليشمل بطارية في المركبة تُستهلك طاقتها بالكامل في كل تحركٍ لها. أضف عدة محطات شحن عبر الطريق. هل تستخدِم أقل عدد مُمكن من محطات الشحن؟

ماذا تعلّمت

- التعرُّف على تقنيات إنترنت الأشياء في الصناعة.
- استخدام برنامج كاب كاربون (CupCarbon) لمحاكاة الشبكات.
- إنشاء مقاطع برمجية بلغة البايثون لبرمجة عُقد الشبكة.
- استخدام بيئة محاكاة كاب كاربون لإنشاء مشاريع إنترنت الأشياء.

المصطلحات الرئيسة

مؤشرات الأداء الرئيسة	Key Performance Indicators
بروتوكول مودبس	Modbus Protocol
تقنية التشغيل	Operational Technology
الصناعة الذكية	Smart Industry

المصنع المُتصل	Connected Factory
التصنيع القائم على البيانات	Data-Driven Manufacturing
الرقمنة	Digitization
حوسبة طرفية	Edge Computing
الأتمتة الصناعية وأنظمة التحكم	Industrial Automation and Control Systems

ملاحظات

ملاحظات