الجمهورية اللبنانية
وزارة التربية والتعليم العالي
المديرية العامة للتعليم المهني والتقني

قرار رقم ٢٠٢٦/١٩٦

اعتماد منهاج التدريب على اختصاص

"مطوّر إنترنت الأشياء: تطوير تطبيقات بايثون لأجهزة إنترنت الأشياء"

الذي لا يؤدي لنيل شهادات رسمية في المديرية العامة للتعليم المهني والتقني

إن المديرة العامة للتعليم المهني والتقني بالتكليف،

بناءً على المذكرة الادارية رقم ٦٥/م/٢٠١٩ تاريخ ٢٠١٩/٠٥/٢٤ (قبول طلب المدير العام بالتكليف للتعليم المهني والتقني إعفاء من تكليفه بهذه المديرية العامة وتكليف مدير المعهد الوطني للعناية التمريضية بمهام المدير العام لها)،

بناءً على المرسوم رقم ٨٣٤٩ تاريخ ١٩٩٦/٠٥/٠٢(تنظيم المديرية العامة التعليم المهني والتقني)،

بناءً على القانون رقم ١٩٦٤/٦٢ تاريخ ١٩٦٤/٦/٣ (تنظيم التعليم المهني الخاص)،

بناءً على المرسوم رقم ٣٦٦٦ تاريخ ٢٠٠٠/٠٨/١٨ (تنظيم لجان المناهج والتدريب وتحديد التعويضات العائدة لها)،

بناءً على القرار رقم ٢٠٢٦/٨٨ تاريخ ٢٠٢٦/٢/١٠ (تأليف لجنة فنية متخصصة لدراسة المناهج التدريبية التي لا تؤدي لنيل الشهادات الرسمية)،

بناء على اقتراح رئيس مصلحة التأهيل المهني بالتكليف،

تقرر ما يأتي:

المادة الأولى: يُعتمَد في المديرية العامة للتعليم المهني والتقني، منهاج التدريب المُرفق بهذا القرار باللغتين العربية والانكليزية، العائد لإختصاص "مطوّر إنترنت الأشياء: تطوير تطبيقات بايثون لأجهزة إنترنت الأشياء" الذي لا يؤدي لنيل شهادة رسمية .

المادة الثانية: يُنشر هذا القرار في الجريدة الرسمية ويُبلغ حيث تدعو الحاجة.

٢٩ نيسان ٢٠٢٦

الدكوانه في:
المديرة العامة للتعليم المهني والتقني

د.هنادي بري

صورة طبق الأصل
مروه نابلسي

٢٠٢٦ - ٢٨٠

german cooperation
DEUTSCHE ZUSAMMENARBEIT

Co-funded by the European Union

Implemented by
giz Deutsche Gesellschaft für Internationale Zusammenarbeit (GIZ) GmbH

تَحْسِينُ جُودَةِ وَجَاذِبِيَّةِ التَّعْلِيمِ وَالتّدريب الْمِهْنِيّ وَالتِّقْنِيّ فِي لُبْنَان لِلفِئَاتِ الإِجْتِمَاعِيَّةِ المُسْتَضْعَفَةِ (QuA-VET)

منهج مطوّر إنترنت الأشياء

المُقرّر التّدريبي: تطوير تطبيقات بايثون لأجهزة إنترنت الأشياء

القطاع: معلوماتية إدارية

التاريخ:23 تموز/يوليو 2025

المقدّمة

إنّ مشروع "تحسين جودة وجاذبية التّعليم والتّدريب المهني والتقني (TVET) في لبنان للفئات الاجتماعية المستضعفة" متعدّد المانحين وممولّ بشكلٍ مشترك من قبل الاتّحاد الأوروبي والوزارة الاتّحادية للتّعاون الاقتصادي والتّنمية (BMZ). ويُنفَّذ هذا المشروع المشترك من قبل الوكالة الألمانية للتّعاون الدولي (GIZ) كمشروع محدّد ضمن المشروع الأوسع التّابع لـ BMZ بعنوان "تحسين جودة وجاذبية التّعليم والتّدريب المهني والتّقني في لبنان (QuA-VET)".

إنّ مشروع تحسين جودة وجاذبية التّعليم والتّدريب المهنيّ والتّقنيّ في لبنان (QuA-VET)، قد طوّر هذا المنهاج كجزء من سلسلة من المناهج في قطاع صناعة الأغذية وإنترنت الأشياء (IoT)، وذلك ضمن جهوده الشاملة لتعزيز استخدام المعايير من قبل مقدّمي التّدريب المهنيّ والتّقنيّ العامّ والخاصّ - المدارس، والمعاهد، والمنظّمات غير الحكوميّة، وغيرها من مؤسّسات التّدريب في لبنان.

إحدى الأهداف الرّئيسيّة لتطوير المعايير واعتمادها في وضع المنهاج هي تعزيز الاتّساق والجودة، بما في ذلك التّدريب الذي يجري خارج إطار المديرية العامّة للتعليم التقني والمهني (DGVTE). ومن الأهداف الأخرى في اعتماد مقاربة المقرّرات التّدريبيّة الصُغرى هي إدخال مزيد من المرونة (سواء خارجية أو داخلية) في نظام التّعليم والتّدريب المهني؛ وجعل التّعليم والتّدريب المهني أكثر جاذبيّة ورفع مكانته؛ وزيادة معدلات المشاركة و/أو تقليل معدلات التّسرّب المُبكر؛ وتعزيز الرّوابط مع سوق العمل لتلبية احتياجات الصّناعة.

يُمكن اعتبار عمليّة تصميم مقرّرات تدريبيّة وتقديمها على أنّها عمليّة تصميم منهاج وتقديمه. يقوم نموذج تصميم المنهاج بتقسيم العمليّة إلى أربع مراحل متّصلة ببعضها البعض، تشكل معاً ما يمكن أن نسمّيه المنهاج الموجّه نحو المخرجات (أو المبني على الكفايات):

- المعيار المهني: مستند يُحدّد ما يحتاج الأشخاص إلى تعلّمه وكيفيّة تقييم جودة ومحتوى التّعلّم،
- معيار المؤهلات: مستند يُحدّد ما يحتاج الأشخاص إلى تعلّمه،
- معيار التّقييم: مستند يُحدّد كيفيّة تقييم جودة ومحتوى التّعلّم، للتحقّق بعدها ممّا إذا كان المتعلّم كفؤا لأداء المهنة؛
- المعيار التّربويّ: مستند يُترجم معايير المؤهّلات في سياق تعليمي،
- المعيار التّدريبي أو التّعليمي: مستند يُخطط للأنشطة التّعليميّة والتّعلّميّة، مع مراعاة البيئات والموارد المطلوبة، والمدة واحتياجات المتعلّمين. ويقُدّم تفاصيل حول كلّ المهام والمهام الفرعية الّتي تشكل المُؤهّل.

يمكن استخدام هذا المنهاج لمجموعة متنوّعة من الأغراض، بما في ذلك:

- توجيه تطوير قدرات المتدرّبين
- اعتمّاد برامج تدريبيّة
- تقييم الأداء وتطوير المسار المهني
- دعم مقدّمي التّدريب الذين يقومون بتطوير أو مراجعة برامج تدريبيّة أساسيّة ومتقدّمة
- وضع معايير لتقييم التّعلّم القَبلي وتحديده
- تقديم توجيهات لأصحاب العمل لاختيار الموظفين وتوظيفهم وتدريبهم واستبقائهم

يحدّد هذا المنهاج المعايير نحو أفضل الممارسات في المهنة وتمّ تطويره حول عدد محدود من وحدات الكفاية (3 إلى 5) الّتي تجمع القدرات المطلوبة لأداء المهام وفقًا للممارسات الفُضلى المعترف بها دوليًا في القطاع. يمكن تقييم كلّ وحدة كفاية بشكل مستقل عن الأخرى ويمكن اكتسابها في أماكن مختلفة (نظامية، غير نظامية، لانظامية). سيتمّ إجراء تقييم الكفايات من قبل جهة مهنية مستقلّة عن مقدّم التّدريب، ممّا يضمن الموثوقيّة والشفافية والاتّساق في التّقييم وفي إقرار المؤهل.

بغض النظر عن حجم المشروع، يجب أن تتمّ المهام بطريقةٍ تتوافق مع إرشادات وأنظمة السّلامة والصحّة المهنية.

تقديم المنهاج

المنهاج هو مجموعة من المعايير الّتي يجب تفسيرها وتقديمها للمتدرّبين مع الأخذ في الاعتبار أنه عند انتهاء البرنامج التّدريبي، سيتمّ اختبارهم وفقًا للمعايير المحدّدة في معايير التّقييم الّتي تعكس الممارسات الفضلى. إنّ المعايير التّربويّة والتّدريبية هما المستندَيْن اللّذين سيستخدمهما المدرّب في تقديم المنهاج. تحدّد المعايير التّربويّة الإطار الزمني العام اللّازم لتعلّم الكفايات المحدّدة في كلّ وحدة كفاية، بينما تحدّد المعايير التّدريبيّة الممارسات الفضلى المعتمدة في أداء المهام من قبل عامل متخصّص ومؤهّل. يجب أن تُعتَبَر المعايير التّدريبيّة

كإرشادات للمدرّب لضمان أن المتدرّبين سيؤدون المهام على النحو المبيّن في المعايير التّدريبيّة. وسيقوم المدرّب بمتابعة تقدّم المتدرّبين في اكتساب القدرات من خلال ملء تقييم تكويني لكلّ متدرّب.

المواد والمرافق اللّازمة

لتقديم المُقرّر التّدريبيّ بشكل مناسبٍ، من الضروري توفير المرافق اللّازمة (ورش تدريبيّة عمليّة)، والمواد والأدوات الّتي تمّ ذكرها ووصفها في المعايير التّدريبيّة. يجب أن تكون الأدوات والمواد تلك الّتي تستخدم حاليًا في المهنة. ويجب ألا تكون النماذج أو المعدّات هادفة إلى الفهم فقط، بل يجب أن تكون أدوات ومواد تهدف وتساعد على ممارسة مهام المهنة.

من الضروري بالنسبة لتعلّم قدرات المهنة أن تُشبه التّمارين التّطبيقيّة ظروف العمل الفعليّة قدر الإمكان. وبالتّالي، تُعَدّ الأدوات والمواد، وكذلك الإرشادات المقدّمة بخصوص مكان التّدريب التّطبيقيّ، شرطًا أساسيًا لتوضيح تقنيات العمل من قبل المدرّب، تليها تمارين يقوم بها المتعلّمون بأنفسهم. وهذا يعني أيضًا أنه يجب احترام عدد الأدوات والمواد كما هو مذكور في المعايير التّدريبيّة لضمان تدريب عملي حقيقي. بالإضافة إلى ذلك، يجب استخدام المواد السمعية والبصرية لمساعدة المتدرّبين على تصوّر الإجراءات والعمليات الّتي لا يمكن توضيحها أو ممارستها.

مقاربتا التّعليم والتّعلّم

يجب أن يختار المدرّب مقاربة تعليميّة وتعلّميّة تشاركيّة يوفّر من خلالها أكبر قدر ممكن من الإمكانيّة للمتدرّب للاكتشاف والممارسة. وينبغي تشجيع المتدرّب على المشاركة الفعّالة في تنمية مهاراته من خلال المناقشة فضلًا عن التّمارين التّطبيقيّة في العالم الحقيقي لورشة العمل.

وتتّسم المقاربة التّعليميّة المرغوب فيها للتّدريب بما يلي:

- يستند التّعليم إلى التّوجيه العملي والعمل القائم على المشاريع في مجموعات؛
- تسترشد الخبرة التّعلّميّة بتغذية راجعة متكرّرة باستخدام ورقة التّقييم التكويني؛
- يتمحور التّعليم، إلى حدٍّ كبير، حول الميدان وذلك على أساس مشاكل عمل وحالات واقعيّة؛

وعلاوةً على ذلك، يُستخدم المعيار التّدريبي كأساس لوضع خطّة تدريبيّة تشمل ما يلي:

- الجدول الزمني، مع تخصيص المدّة المناسبة من وقت التّدريب لكلّ محتوى ومهمّة؛
- إسناد كلّ محتوى ومهمّة إلى مكان مناسب للتّدريب (صف في مدرسة/مركز تدريب أو ورشة عمل أو شركة).

التّقييم

بناءً على معايير التّقييم، سيتمّ بعد ذلك تقييم الأفراد لتحديد ما إذا كانت لديهم القدرات اللّازمة لأداءٍ فعّالٍ في العمل. سيضمن تقييم المتدرّبين من قبل كيانٍ مهنيٍّ مستقلٍّ عن مقدّم التّدريب، تقديم تغذية راجعة إلى مُقدّم التّدريب بشأن «جودة» التّدريب المُقدّم.

قائمة المحتويات

1 المعايير

1.1 المعيار المهنيّ

الرمز ISCO-08: 2512 مطوّرو البرمجيات	المهيار المهني **مطوّر إنترنت الأشياء**
تشمل المهنة عملية تطوير، واختبار، ونشر التطبيقات المستندة إلى لغة بايثون على أجهزة إنترنت الأشياء، مع التركيز على جمع البيانات، ومراقبة الأجهزة، والتحكم فيها.	

العملية	العنصر	الوصف
P1 تطوير تطبيقات بايثون لأجهزة إنترنت الأشياء	المخرج	كود تطبيق إنترنت الأشياء، وجهاز إنترنت الأشياء الذي يعمل بالتطبيق المنشور، وثائق التكوين
	العملية الفرعية	إعداد بيئة التطوير كتابة نصوص برمجية لجمع البيانات من مستشعرات إنترنت الأشياء تطوير برامج للتحكم في الأجهزة إنشاء اتصال شبكي باستخدام لغة بايثون للتكامل السحابي تنفيذ أنشطة نهاية العمل
	المدخل	منتج أو خدمة جديدة، تحديثات التكنولوجيا، متطلبات الصناعة، طلب الزبائن، اتجاهات السوق، متطلبات المشروع
	الموارد	المستشعرات، والمشغلات، والأنظمة المدمجة، ومصادر الطاقة، والشبكة، وأجهزة إنترنت الأشياء، وبرمجيات بايثون ومكتباتها، وخدمات السحابة

1.2 معيار الكفاية

رمز ISCO-08: 2512 مطوّرو البرمجيات		معيار الكفاية مطوّر إنترنت الأشياء
يجب أن يكون الشخص المختص قادرًا على تطوير واختبار تطبيقات بايثون على أجهزة إنترنت الأشياء، لجمع البيانات والتحكم في الجهاز.		
وحدة الكفاية	**القدرة/الكفاية** **مخرجات التعلّم**	**المعرفة**
CU1 إعداد بيئة التطوير	إعداد بيئة تطوير متكاملة (IDE) وبايثون لمايكرو بايثون تهيئة أدوات التطوير وبرامج التشغيل إنشاء اتصال بين وحدات التحكم الدقيقة استيراد المكتبات والاعتماديات الخارجية	مايكروبايثون برامج تشغيل الأجهزة المكتبات الخارجية بروتوكولات الاتصال التسلسلي معالجة منافذ USB/COM
CU2 كتابة نصوص جمع بيانات إنترنت الأشياء	كتابة نصوص بلغة بايثون لجمع البيانات من مستشعرات إنترنت الأشياء باستخدام مايكرو بايثون ربط المستشعرات برمز لقراءة بياناتها وعرضها تصحيح أخطاء جمع البيانات استخراج بيانات المستشعر	نصوص بايثون لجمع البيانات من أجهزة استشعار مختلفة أجهزة استشعار رقمية أو تناظرية طرق جمع البيانات أدوات تصحيح الأخطاء أنماط الأخطاء الشائعة
CU3 تطوير البرامج للتحكّم في الأجهزة	كتابة برامج تحكم بلغة بايثون للمشغلات والأجهزة الطرفية اختبار منطق التحكم لتشغيل الأجهزة بناءً على الشروط استخدام منافذ الدخول/الخروج العامة GPIOs ومخرجات تعديل عرض النبضة PWM تحسين منطق التحكم باستخدام اتصالات الأجهزة مع ضمان كفاءة الطاقة	بايثون المشغلات منافذ الدخول/الخروج العامة GPIO ومخرجات تعديل عرض النبضة PWM والمنطق الشرطي المكتبات
CU4 إنشاء اتصال بالشبكة باستخدام بايثون للتكامل السحابي	تهيئة إعدادات الشبكة لأجهزة إنترنت الأشياء كتابة نصوص برمجية بلغة بايثون لتوصيل الأجهزة بالمنصات السحابية نقل بيانات المستشعرات بأمان إلى السحابة استكشاف أخطاء اتصال الشبكة الأساسية ونقل البيانات وإصلاحها	بروتوكولات الشبكة المستخدمة في إنترنت الأشياء (Wi-Fi وMQTT وHTTP) أساسيات بايثون لاتصالات إنترنت الأشياء هيكل ووظيفة منصات إنترنت الأشياء السحابية مبادئ أمان البيانات للنقل السحابي
CU5 تنفيذ أنشطة نهاية العمل	إجراء التنظيف والنسخ الاحتياطي توثيق رمز المصدر تخزين أدوات الأجهزة إعداد التقارير وفقًا لسجلات المهام	تقنيات التوثيق أنظمة التحكم في الإصدارات وسير العمل باستخدام أدوات مثل Git التقارير والسجلات

1.3 معيار التقييم

رمز ISCO-08: 2512 مطوّرو البرمجيات	معيار الكفاية مطوّر إنترنت الأشياء
يجب أن يكون الشخص المختص قادرًا على تطوير واختبار تطبيقات بايثون على أجهزة إنترنت الأشياء، لجمع البيانات والتحكم في الجهاز.	
وحدة الكفاية 1	إعداد بيئة التطوير
معايير ومؤشرات التقييم	يُعتبر المرشح مؤهلاً إذا كان بإمكانه/ها إظهار الأداء الأفضل في جميع مخرجات التعلم/القدرات الأربع خلال فترة التقييم المُخصصة: نعم/لا، تثبيت وإعداد بيئة التطوير المتكاملة (IDE) نعم/لا، تثبيت بايثون نعم/لا، تهيئة أدوات التطوير وبرامج التشغيل نعم/لا، إنشاء اتصال بين وحدات التحكم الدقيقة نعم/لا، استيراد المكتبات والاعتماديات الخارجية
إجراءات التقييم	يعمل المُقيّم على مراقبة المرشح أثناء تثبيت وإعداد بيئة التطوير المتكاملة (IDE)، وتثبيت بايثون وأي مكتبات مطلوبة. يقوم المرشح بتهيئة أدوات التطوير وتثبيت برامج التشغيل اللازمة بناءً على تعليمات الإعداد. ويتم إنشاء اتصال بين وحدات التحكم الدقيقة من خلال توصيل الأجهزة والتحقق من التفاعل بينها. كما يقوم المرشح باستيراد المكتبات والاعتماديات الخارجية كما هو مُدرج في متطلبات التطوير. يتم ضمان الجودة من خلال استخدام قائمة تحقق موحدة، ومراجعة النتائج بعد الانتهاء. سيتم تحديد وقت التقييم مسبقًا وبشكل يكفي لجميع المهام، بناءً على التجارب التشغيلية السابقة.
وحدة الكفاية 2	كتابة نصوص جمع بيانات إنترنت الأشياء
معايير ومؤشرات التقييم	يُعتبر المرشح مؤهلاً إذا كان بإمكانه/ها إظهار الأداء الأفضل في جميع مخرجات التعلم/القدرات الأربع: نعم/لا، كتابة نصوص بلغة بايثون لجمع البيانات من مستشعرات إنترنت الأشياء باستخدام مايكرو بايثون نعم/لا، ربط المستشعرات برمز نعم/لا، تصحيح أخطاء جمع البيانات نعم/لا، استخراج بيانات المستشعر
إجراءات التقييم	يعمل المُقيّم على مراقبة المرشح اثناء كتابة نصوص بايثون لجمع البيانات من مستشعرات إنترنت الأشياء باستخدام المتطلبات المُقدمة. ويقوم المرشح بربط المستشعرات بالرمز كي تستجيب المستشعرات ويتم نقل البيانات. خلال العملية، يُحدد المرشح أي أخطاء تتعلق بجمع البيانات ويُصحّحها. ثم يُستخرج المرشح بيانات المستشعر للاستخدام المُستقبلي. وتتم المحافظة على اتساق التقييم بين المُقيّمين والمؤسسات من خلال استخدام قائمة مراجعة معيارية للتقييم، ومعايير أداء واضحة، وجلسات المعايرة أو التحقق ما بعد التقييم لمواءمة قرارات الدرجات.
وحدة الكفاية 3	تطوير البرامج للتحكّم في الأجهزة
معايير ومؤشرات التقييم	يُعتبر المرشح مؤهلاً إذا كان بإمكانه/ها إظهار الأداء الأفضل في جميع مخرجات التعلم/القدرات الأربع: نعم/لا، برامج التحكم المُستخدمة في المُشغلات والأجهزة الطرفية نعم/لا، اختبار منطق التحكم لتشغيل الأجهزة بناءً على الظروف نعم/لا، استخدام منافذ الدخول/الخروج العامة GPIOs ومخرجات تعديل عرض النبضة PWM نعم/لا، تحسين منطق التحكم بناءً على نتائج الاختبار وملاحظات الأداء
إجراءات التقييم	يعمل المُقيّم على مراقبة المرشح باستخدام برامج التحكم لتشغيل المُشغلات والأجهزة الطرفية بناءً على متطلبات المهمة. يختبر المرشح منطق التحكم لتشغيل الأجهزة استجابةً لظروف مُحددة. تُستخدم دبابيس الإدخال/الإخراج للأغراض العامة (GPIO) ومخرجات تعديل عرض النبضة (PWM) لإدارة سلوك الجهاز. ثم يعمل المرشح على تحسين منطق التحكم بناءً على نتائج إجراءات الاختبار.
وحدة الكفاية 4	إنشاء اتصال بالشبكة باستخدام بايثون للتكامل السحابي
معايير ومؤشرات التقييم	يُعتبر المرشح مؤهلاً إذا كان بإمكانه/ها إظهار الأداء الأفضل في جميع مخرجات التعلم/القدرات الأربع: نعم/لا، تهيئة إعدادات الشبكة لأجهزة إنترنت الأشياء نعم/لا، كتابة نصوص برمجية بلغة بايثون لتوصيل الأجهزة بالمنصات السحابية

	نعم/لا، نقل بيانات المستشعرات بأمان إلى السحابة نعم/لا، استكشاف أخطاء اتصال الشبكة الأساسية ونقل البيانات وإصلاحها
إجراءات التقييم	يعمل المُقيّم على مراقبة المرشح أثناء تهيئة إعدادات الشبكة لأجهزة إنترنت الأشياء وفقًا لمتطلبات المهمة. يكتب المرشح نصوص بايثون لتوصيل أجهزة إنترنت الأشياء بالمنصات السحابية كما هو مُحدد. تُنقل بيانات المستشعر بأمان إلى السحابة. ثم يُحل المرشح مشاكل اتصال الشبكة الأساسية ونقل البيانات لضمان اتصال مستقر بين الأجهزة والسحابة.
وحدة الكفاية 5	تنفيذ أنشطة نهاية العمل
معايير ومؤشرات التقييم	يُعتبر المرشح مؤهلاً إذا كان بإمكانه/ها إظهار الأداء الأفضل في جميع مخرجات التعلم/القدرات الأربع: نعم/لا، إجراء التنظيف والنسخ الاحتياطي نعم/لا، توثيق رمز المصدر نعم/لا، تخزين أدوات الأجهزة نعم/لا، إعداد تقارير سجلات المهام
إجراءات التقييم	يعمل المُقيّم على مراقبة المرشح أثناء إجراء التنظيف وإنشاء نسخة احتياطية من ملفات المشروع بحسب ما يكون مطلوباً. يُوثّق المرشح رمز المصدر بإضافة التعليقات والمعلومات اللازمة لدعم الفهم والاستخدام المُستقبلي. تُجمع أدوات الأجهزة المُستخدمة أثناء المهمة وتُخزَّن في مواقعها المُخصصة. يُكمل المرشح العملية من خلال إعداد تقارير سجلات المهام وفق النموذج المخصص.

1.4 المعيار التّربوي

رمز ISCO-08: 2512 مطوّرو البرمجيات	**المعيار التعليمي** مطوّر إنترنت الأشياء
المعيار المهني	تطوير تطبيقات بايثون على أجهزة إنترنت الأشياء (رمز ISCO-08: 2512)
معيار التقييم	تطوير تطبيقات بايثون على أجهزة إنترنت الأشياء (رمز ISCO-08: 2512)
لمحة عامة	عند إتمام وحدة التدريب، سيكون المشارك قادرًا على تطوير، واختبار، ونشر التطبيقات المستندة إلى بايثون على أجهزة إنترنت الأشياء، مع التركيز على جمع البيانات، ومراقبة الأجهزة، والتحكم فيها بالإضافة إلى إنشاء اتصال بالشبكة باستخدام بايثون لتحقيق التكامل السحابي.
الكفايات	إعداد بيئة تطوير متكاملة (IDE) وبايثون لمايكرو بايثون تهيئة أدوات التطوير وبرامج التشغيل إنشاء اتصال بين وحدات التحكم الدقيقة استيراد المكتبات والاعتماديات الخارجية كتابة نصوص بلغة بايثون لجمع البيانات من مستشعرات إنترنت الأشياء باستخدام مايكرو بايثون ربط المستشعرات برمز لقراءة بياناتها وعرضها تصحيح أخطاء جمع البيانات استخراج بيانات المستشعر كتابة برامج تحكم بلغة بايثون للمشغلات والأجهزة الطرفية اختبار منطق التحكم لتشغيل الأجهزة بناءً على الشروط استخدام منافذ الدخول/الخروج العامة GPIOs ومخرجات تعديل عرض النبضة PWM تحسين منطق التحكم باستخدام اتصالات الأجهزة مع ضمان كفاءة الطاقة تهيئة إعدادات الشبكة لأجهزة إنترنت الأشياء كتابة نصوص برمجية بلغة بايثون لتوصيل الأجهزة بالمنصات السحابية نقل بيانات المستشعرات بأمان إلى السحابة استكشاف أخطاء اتصال الشبكة الأساسية ونقل البيانات وإصلاحها إجراء التنظيف والنسخ الاحتياطي توثيق رمز المصدر تخزين أدوات الأجهزة إعداد التقارير وفقًا لسجلات المهام
المعرفة	مايكروبايثون برامج تشغيل الأجهزة المكتبات الخارجية بروتوكولات الاتصال التسلسلي معالجة منافذ USB/COM نصوص بايثون لجمع البيانات من أجهزة استشعار مختلفة أجهزة استشعار رقمية أو تناظرية طرق جمع البيانات أدوات تصحيح الأخطاء أنماط الأخطاء الشائعة بايثون المشغلات منافذ الدخول/الخروج العامة GPIO ومخرجات تعديل عرض النبضة PWM والمنطق الشرطي المكتبات بروتوكولات الشبكة المستخدمة في إنترنت الأشياء (Wi-Fi وMQTT وHTTP) أساسيات بايثون لاتصالات إنترنت الأشياء هيكل ووظيفة منصات إنترنت الأشياء السحابية مبادئ أمان البيانات للنقل السحابي تقنيات التوثيق أنظمة التحكم في الإصدارات وسير العمل باستخدام أدوات مثل Git التقارير والسجلات
برنامج التعلّم	سيتم تغطية المواضيع التالية خلال دروس الصف:

	1. مقدمة في إنترنت الأشياء وبيئات التطوير 2. بايثون لجمع بيانات إنترنت الأشياء 3. الاتصال بين أجهزة التحكم الدقيقة والمستشعر 4. برمجة التحكم في الأجهزة باستخدام بايثون 5. التكامل السحابي والنقل الآمن للبيانات 6. إجراءات نهاية العمل وتوثيقها سيتم تنفيذ الأنشطة العملية التالية خلال دروس ورشة العمل: 1. تثبيت وتهيئة بايثون، وبيئات التطوير المتكاملة (IDEs)، وبرامج التشغيل 2. اختبار اتصال الجهاز 3. تطوير ونشر نصوص بايثون لجمع بيانات المستشعرات في الوقت الفعلي 4. تصحيح أخطاء النصوص والتحقق من صحة استخراج بيانات المستشعر 5. توصيل الدوائر 6. كتابة واختبار برامج التحكم لتشغيل المشغلات والأجهزة الطرفية 7. تحسين منطق التحكم باستخدام اتصالات الأجهزة 8. تكوين إعدادات الشبكة لاتصال الجهاز 9. كتابة نصوص بايثون لتوصيل الأجهزة بالمنصات السحابية 10. اختبار واستكشاف أخطاء النقل الآمن لبيانات المستشعرات وإصلاحها سيتم تنفيذ الأنشطة التالية خلال التدريب العملي: 1. تهيئة إعدادات الشبكة لتوصيل الجهاز (5 ساعات) 2. اختبار توصيل الجهاز ضمن سير العمل الفعلي (3 ساعات) 3. تطوير ونشر نصوص بايثون لجمع بيانات المستشعرات في الوقت الفعلي (16 ساعة) 4. تصحيح أخطاء النصوص في بيئة تشغيلية مباشرة (6 ساعات) 5. كتابة واختبار برامج التحكم لتشغيل المحركات والأجهزة الطرفية الخاصة بمكان العمل (18 ساعة) 6. تحسين منطق التحكم باستخدام اتصالات الأجهزة (ساعتان) 7. كتابة نصوص بايثون لتوصيل الأجهزة بالمنصات السحابية (15 ساعة) 8. اختبار واستكشاف أخطاء نقل البيانات الآمن (5 ساعات) 9. إجراء عمليات التنظيف والنسخ الاحتياطي (3 ساعات) 10. توثيق رمز المصدر وسجلات الصيانة (4 ساعات) 11. تخزين الأدوات وإكمال التقارير (3 ساعات) يتم تنسيق التدريب أثناء العمل مع الجهات المضيفة المعتمدة، وفقًا لخطة تدريب متفق عليها. وتتم مراقبة التقدم من خلال عمليات تسجيل الدخول المنتظمة للمشرف، وسجلات الأنشطة التي يوقعها كل من المتدرب والمشرف، والمراجعات الدورية من قبل مقدم التدريب لضمان تحقيق نتائج التعلّم.

الإطار الزمني للتعلّم

رمز الكفاية	الفصل الدراسي (ساعات)	دروس ورشة العمل (ساعات)	التدريب أثناء العمل (ساعات)	المجموع (ساعات)
وحدة الكفاية 1	1	2	3	6
وحدة الكفاية 2	2	8	22	32
وحدة الكفاية 3	1	8	20	29
وحدة الكفاية 4	2	12	25	39
وحدة الكفاية 5	1	3	10	14
المجموع (ساعات)	7	33	80	120

2 وحدة الكفاية 1: إعداد بيئة التطوير

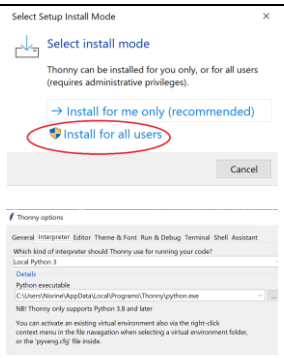
2.1 معيار التدريب - الوحدة 1

2.1.1 المبادىء التوجيهية للتدريب

ناتج القدرات/التعلّم:	الأدوات المطلوبة:	المواد والمستهلكات المطلوبة:
• إعداد بيئة تطوير متكاملة (IDE) وبايثون لمايكرو بايثون • تهيئة أدوات التطوير وبرامج التشغيل • إنشاء اتصال بين وحدات التحكم الدقيقة • استيراد المكتبات والاعتماديات الخارجية **موقع التدريب:** غرفة داخلية بمساحة تقريبية 4x5 متر، مجهزة بـ: • نافذة واسعة • طفاية حريق • مفاتيح ومقابس كهربائية متعددة • أثاث خشبي ثابت مناسب لأنشطة مختبر الإلكترونيات • مكاتب ومناضد لوضع أجهزة الكمبيوتر، والأدوات، والمواد اللازمة للتطبيق العملي • خزانة وأدراج لتنظيم وتخزين المكونات الإلكترونية ومواد المختبر	• كمبيوتر شخصي وشاشة أو كمبيوتر محمول مزود بمنفذ USB • اتصال بالإنترنت • حصيرة مضادة للكهرباء الساكنة، آمنة ضد التفريغ الكهروستاتيكي	• لوحة تطوير ESP8266 (مثل NodeMCU و Wemos D1 Mini) • كابل بيانات Micro USB (تأكد من أنه كابل بيانات، وليس كابل شحن فقط) • كتالوجات ومواصفات أجهزة إنترنت الأشياء • ورقة مرجعية لقواعد لغة مايكرو بايثون

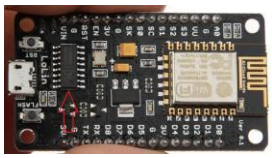
إعداد بيئة تطوير متكاملة (IDE) وبايثون لمايكرو بايثون

تنزيل/تثبيت بايثون • قم بتنزيل بايثون 3.8+ من Download Python|Python.org • اتبع تعليمات التثبيت وحدد "إضافة بايثون إلى المسار" • تحقق من تثبيت بايثون: ○ افتح موجه الأوامر بالضغط على Win + R ثم اكتب cmd واضغط على Enter ○ اكتب: *python --version* ○ إذا كان بايثون مثبتًا، سترى شيئًا مثل: Python 3.8.5 ○ ذا لم تنجح عملية التثبيت، قد لا تتم إضافة بايثون إلى مسار نظامك. عليك تكرار عملية تثبيت بايثون.	
اختر بيئة التطوير المتكاملة (IDE) وثبّتها • افتح المتصفح الخاص بك وانتقل إلى https://thonny.org • نزّل بيئة التطوير المتكاملة "Thonny" لنظام ويندوز • حدد موقع الملف الذي تم تنزيله (thonny-4.x.x.exe) في مجلد التنزيلات الخاص بك • قم بالنقر المزدوج لتشغيل برنامج التثبيت • حدد "التثبيت لجميع المستخدمين" ثم اقبل الاتفاقية • اتبع تعليمات برنامج التثبيت بالإعدادات الافتراضية • حدد "إضافة Thonny إلى PATH" إذا طُلب منك ذلك	



- افتح بيئة التطوير المتكاملة للتأكد من عملها
- تحقق من تثبيت بايثون: انتقل إلى الأدوات > الخيارات > المفسّر

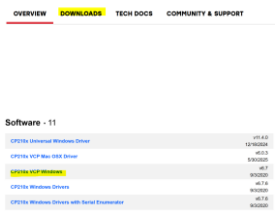
تهيئة أدوات التطوير وبرامج التشغيل



حدد شريحة USB-to-Serial

- افحص اللوحة لديك بشكل مباشر، ابحث عن العلامات على شريحة USB-to-Serial الصغيرة بالقرب من منفذ USB: "CH340G" ← مطلوب برنامج تشغيل CH340
- "CP2102" ← مطلوب برنامج تشغيل CP210x

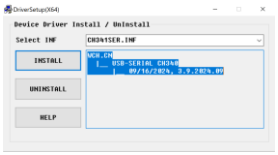
إذا لم يكن واضحًا: تحقق من ورقة بيانات اللوحة لديك



حمّل برنامج التشغيل

- حمّل برنامج تشغيل CH340 من الموقع الإلكتروني: https://www.wch.cn/downloads/CH341SER_ZIP.html
- تصفح الموقع الرسمي لـ Silicon Labs لتنزيل برنامج تشغيل CP210x من: https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers
- انقر على تبويب "التنزيلات" واختر: **برامج تشغيل CP210x لنظام Windows**

نصيحة: يُفضل تنزيل كلا برنامجي التشغيل إذا لم تكن متأكدًا، فهذا لن يؤدي إلى أي تعارض

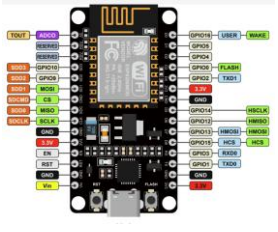


تثبيت برنامج التشغيل

- ثبّت CH340:
  - استخرج محتويات ملف zip. الذي تم تنزيله
  - شغّل SETUP.EXE (ويندوز)
  - اتبع التعليمات التي تظهر على الشاشة ← انقر على "تثبيت"
  - ستظهر لك رسالة "تم تثبيت برنامج التشغيل بنجاح"

- ثبّت CP210x:
  - شغّل برنامج التثبيت exe.
  - اقبل الشروط ← انقر على "التالي"
  - أكمل التثبيت

إنشاء اتصال بين وحدات التحكم الدقيقة



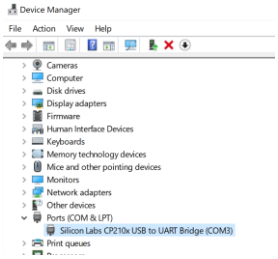
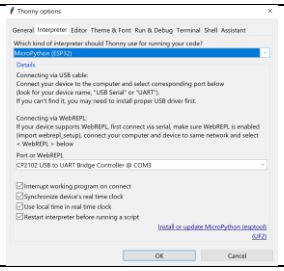
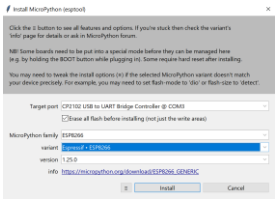
افحص لوحة ESP8266 للتعرف على تصميمها:

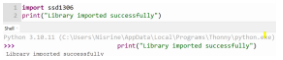
- حدد المكونات الرئيسية:

شريحة ESP8266 (المربع/المستطيل الأسود الصغير)
موصل USB (للبرمجة والطاقة)
مصابيح LED للحالة (للطاقة والاتصال)

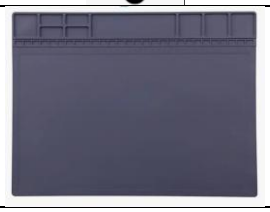
- حدد موقع توصيلات الدبابيس:

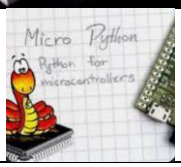
دبابيس الطاقة: 3.3 فولت، GND

	دبابيس GPIO: مدخل/مخرج عام للأجهزة الاستشعارية، والمشغلات، إلخ دبابيس خاصة: إعادة الضبط (RST)، التمكين (EN)، المدخل التناظري (ADC)
	**أوصل المتحكم الدقيق** • أوصل ESP8266 بالكمبيوتر باستخدام كابل بيانات Micro USB • تأكد من عدم وضع لوحة ESP32 على سطح معدني أثناء التشغيل لتجنب حدوث قصر كهربائي، وذلك بوضعها على مانع التسرب ESD • تأكد من إضاءة مصباح LED الخاص باللوحة (مؤشر الطاقة) • انتظر حتى يكتشف نظام ويندوز الجهاز
	**تحقق من منفذ COM** • افتح "إدارة الأجهزة": اضغط على Win + X ← "إدارة الأجهزة" • وسّع "المنافذ (COM وLPT)". ابحث عن: USB-SERIAL CH340 (COMx) لـ CH340 أو Silicon Labs CP210x USB to UART Bridge (COMx) لـ CP210x. ستحتاج إلى منفذ COM في ثوني لاحقًا إذا لم يظهر منفذ COM: • جرّب منفذ USB مختلفًا • جرّب كابل Micro USB مختلفًا (يجب أن يكون كابل بيانات) • أعد تثبيت برنامج التشغيل • أعد تشغيل الكمبيوتر إذا لزم الأمر. جرّب لوحة ESP8266 أخرى
	**تكوين ثوني** • افتح ثوني • انتقل إلى أدوات > خيارات > المفسّر • في المفسّر: حدد MicroPython (ESP8266) أو ESP32 • في المنفذ: حدد منفذ COM المحدد وانقر على موافق
 	**تفليش برنامج مايكروبايثون الثابت** • افتح ثوني • انتقل إلى أدوات > خيارات > المفسّر • انقر على "تثبيت أو تحديث مايكروبايثون (esptool)" • حدد المنفذ المستهدف • حدد عائلة مايكروبايثون: ESP8266 • حدد الإصدار: Espressif ESP8266 • حدد أحدث إصدار • انقر على "تثبيت" • انتظر التفليش وأعد ضبط ESP8266 لاحظ أنه في حال اكتمال التفليش وإعادة ضبط اللوحة دون أخطاء، فهذا يعني أن البرنامج الثابت firmware مثبت بنجاح
	**اختبر الاتصال مع المتحكم الدقيق** • انقر على "إيقاف/إعادة تشغيل الخادم الخلفي" سوف يظهر أمامك : MicroPython v1.x.x على x-xx-xx202؛ وحدة ESP مع ESP8266

استيراد المكتبات والاعتماديات الخارجية	
	**تجهيز المكتبة وتحميلها** • نزّل المكتبة من برامج تشغيل مايكرو بايثون على برامج تشغيل GitHub (مثال على المكتبة: ssd1306.py) • احفظها على جهاز الكمبيوتر (مثلاً، مجلد التنزيلات) • افتح ثوني • وصّل جهاز ESP8266 • تأكد من ظهور REPL في النافذة السفلية. هنا يمكنك كتابة أوامر مايكرو بايثون وتشغيلها مباشرةً على جهاز ESP8266 بعد نجاح الاتصال • انتقل إلى ملف > إفتح... > هذا الكمبيوتر • افتح ssd1306.py • انتقل إلى ملف > حفظ باسم... > جهاز مايكرو بايثون • احفظ باسم ssd1306.py • تحقق من التحميل: في متصفح ملفات ثوني (الشريط الجانبي الأيسر)، تأكد من ظهور ssd1306.py أسفل ملفات ESP8266
	**اختبر الاستيراد** • في نافذة REPL الخاصة بـ ثوني (أسفل لوحة التحكم)، اكتب ما يلي: import ssd1306 اكتب ("تم استيراد المكتبة بنجاح") • اضغط على Enter ○ إذا لم تظهر أي أخطاء وظهرت الرسالة: *Library imported successful* فهذا يعني أن مكتبة ssd1306 قد تم استيرادها بنجاح ○ إذا ظهر خطأ مثل *ModuleNotFoundError*، فهذا يعني أن المكتبة لم يتم تثبيتها أو تحميلها على جهازك بعد

2.1.2 الادوات والمواد

الادوات	الصورة	الكمية [1]	الملاحظة
جهاز كمبيوتر شخصي مع شاشة ولوحة مفاتيح وفأرة (مثبت عليه نظام تشغيل ويندوز) أو كمبيوتر محمول مزود بمنفذ USB مثبت عليه (Python 3.8+، و Thonny IDE، وبرامج تشغيل USB: برامج تشغيلCP210x / CH340، وصلاحيات إدارية لتثبيت التعريفات/البرمجيات)		1 لكل محطة	
اتصال بالإنترنت مع بيانات اعتماد شبكة WiFi		1 لكل فصل دراسي	
حصيرة مضادة للكهرباء الساكنة آمنة ضد التفريغ الكهروستاتيكي		1 لكل محطة	

المواد والمستهلكات	الصورة	الكمية	الملاحظة
لوحة تطوير ESP8266 (مثل NodeMCU وWemos D1 Mini)		1 لكل محطة	
كابل بيانات Micro USB		1 لكل محطة	(تأكد من أنه كابل بيانات، وليس كابل شحن فقط)
كتالوجات ومواصفات أجهزة إنترنت الأشياء ورقة مرجعية لقواعد لغة مايكرو بايثون		1 لكل محطة	

[1] إن الكمية المشار إليها للورشة تعني إمكانية استخدام المنتج لوحدات مختلفة في الورشة ذاتها، مما يعني أنه لا يُمكن تسليم وحدتين تستخدمان المنتج في نفس الوقت.
تحذير! قبل شراء المنتج، يُرجى التحقق من وجوده في الورشة لتجنب أي تكرار غير ضروري. كما يجب مراجعة القائمة والتحقق منها من قِبل خبير مُختص.

2.2 معايير التقييم- الوحدة 1

2.2.1 التقييم التكويني

اسم المتدرب:	التاريخ:	
القدرات/ناتج التعلّم	**المعايير لوحدة الكفاية 1- إعداد بيئة التطوير**	**نعم/لا**
إعداد بيئة تطوير متكاملة (IDE) وبايثون لمايكرو بايثون	هل قام المرشح بتنزيل وتثبيت بايثون +3.8 من موقع Python.org الرسمي؟	
	هل اتبع المرشح تعليمات التثبيت واختار "إضافة بايثون إلى المسار"؟	
	هل تحقق المرشح من تثبيت بايثون بفتح موجه الأوامر والتحقق من الإصدار باستخدام python --version؟	
تهيئة أدوات التطوير وبرامج التشغيل	هل تعرّف المرشح على شريحة USB-to-Serial على لوحة (CH340G) ESP8266 أو (CP2102) ؟	
	هل نزّل المرشح برنامج التشغيل الصحيح (CH340) أو (CP210x) من المواقع الرسمية؟	
	هل ثبّت المرشح برنامج التشغيل المحدد بنجاح؟	
إنشاء اتصال بين وحدات التحكم الدقيقة	هل حدد المرشح مكونات ESP8266 (الشريحة، موصل USB ، مصابيح LED)؟	
	هل حدد المرشح موقع الطاقة، ومنفذ GPIO ، والدبابيس الخاصة على اللوحة؟	
	هل ربط المرشح ESP8266 بالكمبيوتر باستخدام كابل بيانات micro-USB ؟	
	هل تأكد المرشح من وضع اللوحة على سطح آمن وغير موصل (مثل حصيرةESD) لتجنب حدوث قصر كهربائي؟	
	هل تحقق المرشح من تشغيل اللوحة (إضاءة مؤشر LED)؟	
	هل فتح المرشح إدارة الأجهزة، وحدد منفذ COM الصحيح، وعرّفه على أنه CH340أوCP210x ؟	
	إذا لم يظهر منفذ COM ، هل قام المرشح باستكشاف الأخطاء وإصلاحها (جرب منفذًا آخر، أو كابلًا آخر، أو أعاد تثبيت برنامج التشغيل، أو أعاد تشغيل الكمبيوتر)؟	
	هل فتح المرشح ثوني واختار مايكرو بايثون (ESP8266/ESP32) كمفسّر؟	
	هل حدد المرشح منفذ COM المحدد في ثوني وأكّد الاتصال؟	
	هل نقر المرشح على "أدوات" > "خيارات" > "المفسّر" في ثوني واختار تثبيت أو تحديث مايكرو بايثون (esptool) ؟	
	هل اختار المرشح المنفذ الصحيح، العائلة (ESP8266) والمتغير الصحيح (Espressif ESP8266)؟	
	هل ثبّت المرشح أحدث إصدار من مايكرو بايثون وانتظر اكتمال عملية التثبيت بنجاح؟	
	هل أعاد المرشح ضبط اللوحة وأكّد نجاح تثبيت البرنامج الثابت؟	
	هل أعاد المرشح تشغيل الخادم الخلفي في ثوني وأكّد أن REPL يعرض إصدار مايكرو بايثون الصحيح ومعلومات وحدة ESP8266 ؟	
استيراد المكتبات والاعتماديات الخارجية	هل نزّل المرشح مكتبة خارجية (مثل ssd1306.py) من برامج تشغيل مايكرو بايثون على GitHub ؟	
	هل فتح المرشح المكتبة في ثوني وحفظها على جهاز مايكرو بايثون باسم الملف الصحيح؟	
	هل أكّد المرشح ظهور ملف المكتبة في نظام ملفات ESP8266 داخل ثوني؟	
	هل ختبر المرشح استيراد المكتبة المُحمّلة في REPL الخاص بـ ثوني باستخدام import ssd1306؟	
	هل تحقق المرشح من نجاح الاستيراد بكتابة "تم استيراد المكتبة بنجاح"؟	
	في حال حدوث خطأ (مثل ModuleNotFoundError)، هل قام المرشح بحلّ المشكلة بإعادة تحميل المكتبة؟	

2.2.1 التقييم الختامي أو التجميعي

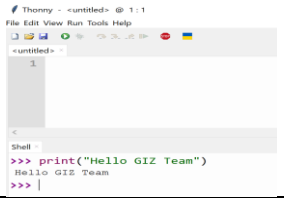
اسم المتدرب:	**التاريخ:**	
اسم المقيّم:	**التاريخ:**	
يعتبر المتدرب مؤهلاً إذا أثبت أداء أفضل الممارسات في جميع المعايير الثمانية التالية للتقييم المتعلقة بوحدة الكفاية 1 – **إعداد بيئة التطوير**		

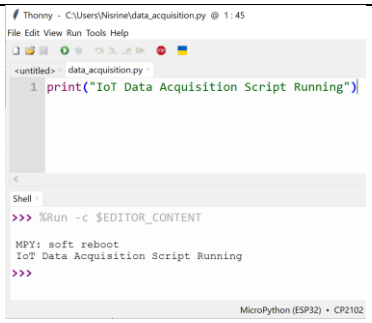
معيار التقييم	المؤهّل	نعم	لا (*)
تثبيت وإعداد بيئة التطوير المتكاملة المطلوبة	يجب على المرشح إكمال تثبيت بيئة التطوير المتكاملة (IDE) المطلوبة (مثل ثوني) وإعدادها الأولي في أقل من 15 دقيقة، مع التأكد من فتحها دون أخطاء والتعرّف على وحدة التحكم الدقيقة المتصلة.	☐	☐
تثبيت بايثون	يجب على المرشح إكمال تثبيت بايثون في أقل من 15 دقيقة، مع التأكد من إضافة بايثون إلى مسار النظام (PATH) والتحقق منه بتشغيل الأمر python --version أو python3 --version في الطرفية دون أخطاء.	☐	☐
تثبيت وتكوين برامج التشغيل وأدوات التطوير	يجب على المرشح إكمال تثبيت وتكوين برامج تشغيل USB-to-Serial وأدوات التطوير المطلوبة في أقل من 15 دقيقة، مع التأكد من تعرّف النظام وبيئة التطوير على جهاز إنترنت الأشياء أو وحدة التحكم الدقيقة دون أخطاء.	☐	☐
إنشاء اتصال مع المتحكم الدقيق	يجب على المرشح إنشاء اتصال بين وحدة التحكم الدقيقة وبيئة التطوير في أقل من 20 دقيقة، مع التأكد من التعرّف على الجهاز، واختيار منفذ الاتصال الصحيح، وتنفيذ أمر الاختبار بنجاح في REPL أو وحدة التحكم دون أخطاء.	☐	☐
استيراد مكتبات خارجية	يجب على المرشح إكمال عملية تحميل واستيراد المكتبات الخارجية المطلوبة إلى أداة التحكم الدقيقة في أقل من 15 دقيقة، مع التأكد من وضع ملفات المكتبة بشكل صحيح على الجهاز، وتنفيذ أمر الاستيراد بدون أخطاء في REPL أو البرنامج النصي، والتحقق من الوظيفة.	☐	☐
المعيار (*)	الدليل على عدم الامتثال إن وجد (**)		
توقيع المقيّم:	التاريخ:		
توقيع المتدرب:	التاريخ:		

3 وحدة الكفاية 2: كتابة نصوص جمع بيانات إنترنت الأشياء

3.1 معيار التدريب- الوحدة 2

3.1.1 المبادىء التوجيهية للتدريب

ناتج القدرات/التعلّم:	الأدوات المطلوبة:	المواد والمستهلكات المطلوبة:
• كتابة نصوص بلغة بايثون لجمع البيانات من مستشعرات إنترنت الأشياء باستخدام مايكرو بايثون • ربط المستشعرات برمز لقراءة بياناتها وعرضها • تصحيح أخطاء جمع البيانات • استخراج بيانات المستشعر **موقع التدريب:** غرفة داخلية بمساحة تقريبية 4x5 متر، مجهزة بـ: • نافذة واسعة • طفاية حريق • مفاتيح ومقابس كهربائية متعددة • أثاث خشبي ثابت مناسب لأنشطة مختبر الإلكترونيات • مكاتب ومناضد لوضع أجهزة الكمبيوتر، والأدوات، والمواد اللازمة للتطبيق العملي • خزانة وأدراج لتنظيم وتخزين المكونات الإلكترونية ومواد المختبر منطقة تربة (أو حاويات تربة) داخل الغرفة لاختبار أجهزة استشعار رطوبة التربة	• كمبيوتر شخصي وشاشة أو كمبيوتر محمول مزود بمنفذ USB • اتصال بالإنترنت • حصيرة مضادة للكهرباء الساكنة، آمنة ضد التفريغ الكهروستاتيكي • مقياس متعدد • بطارية محمولة • محول تيار مستمر 5 فولت بمنفذ USB صغير • صناديق تخزين للمكونات • مجموعة أدوات أساسية (مفكات براغي، كماشة، قاطع أسلاك)	• لوحة تطوير ESP8266 (مثل NodeMCU وWemos D1 Mini) • كابل بيانات Micro USB • وحدة استشعار DHT22/DHT11 • وحدة استشعار رطوبة التربة • وحدة استشعار الإضاءة المحيطة • بطارية ليثيوم أيون قابلة لإعادة الشحن 18650 مع حامل *1 • مصابيح LED • مقاومات • وحدة أزرار الضغط • مفاتيح • لوحة تجارب • أسلاك توصيل جامبر (أنثى-ذكر، ذكر-ذكر)
كتابة نصوص بلغة بايثون لجمع البيانات من مستشعرات إنترنت الأشياء باستخدام مايكرو بايثون		
	**افتح بيئة تطوير ثوني:** • افتح بيئة تطوير ثوني بالنقر المزدوج على رمز ثوني على سطح المكتب أو ابحث عن "ثوني" في قائمة البدء لفتح بيئة التطوير • أوصل نظام ESP8266 أو ESP32 بجهاز الكمبيوتر باستخدام كابل USB. • تأكد من اكتشاف ثوني للجهاز (تحقق من الزاوية اليمنى السفلية للبحث عن "MicroPython (ESP8266)" ومنفذ COM الصحيح) • تأكد من عمل REPL بكتابة: اكتب ("*Hello MicroPython*") وتأكد من ظهور الرسالة بدون أخطاء	



أنشئ نصًا برمجيًا بصيغة .py.

- انقر على ملف > جديد، أو اضغط على Ctrl + N. سيُفتح محرر نصوص برمجية فارغ في ثوني، جاهزًا لكتابة كود إنترنت الأشياء.

احفظ النص البرمجي:

- انقر على ملف > حفظ باسم
- اختر مكان الحفظ: على جهاز الكمبيوتر الخاص بك للتحرير المحلي وتحميله لاحقًا، أو مباشرةً على وحدة التحكم الدقيقة إذا شغّلت النص البرمجي فورًا
- أعطِ تسمية واضحة لملفك، على سبيل المثال: data_acquisition.py

- جهّز لكتابة الشيفرة البرمجية في أعلى النص البرمجي:
  - استورد المكتبات اللازمة (مكتبات الآلة، والوقت، والمستشعرات)
  - هيّئ المستشعر
  - اكتب حلقة جمع البيانات
- اختبر الحفظ والتشغيل:
  - اكتب سطر اختبار في النص البرمجي:
  - اكتب ("IoT Data Acquisition Script Running")
  - انقر على تشغيل (F5) لضمان تنفيذ النص البرمجي، وظهور الرسالة في Shell/REPL.

استخدم import machine، و import time، ومكتبات المستشعرات الخاصة

- حدد المكتبات المطلوبة لجمع بيانات إنترنت الأشياء:

المكتبات الأساسية:

- machine للتحكم في GPIO والأجهزة
- Time للتوقيت والتأخير

المكتبات الخاصة بالمستشعر:

مثال: dht لمستشعرات DHT11/DHT22

- افتح نص إنترنت الأشياء:
  - افتح بيئة تطوير ثوني
  - افتح النص البرمجي data_acquisition.py أو أنشئ ملف .py جديد لاختبار عمليات الاستيراد
- اكتب عبارات الاستيراد في أعلى النص البرمجي:

import machine
import time
import dht # أو مكتبة المستشعر الخاصة بك

- احفظ النص البرمجي

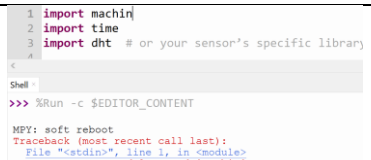
نصيحة: اضغط على Enter بعد كتابة كل سطر ليعمل بشكل صحيح في بايثون

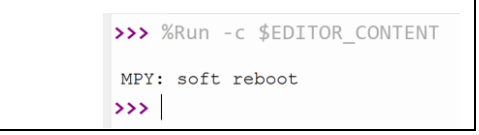
يُستخدم الرمز # للتعليقات، وأي شيء بعد # سوف يتجاهله مفسّر بايثون، وهو موجود فقط لمساعدتك على فهم الكود

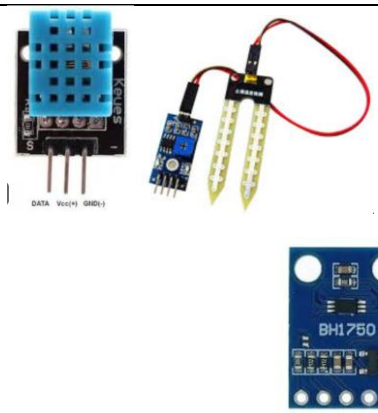
```
1 import machine
2 import time
3 import dht  # or your sensor's specific library
```

اختبار عبارات الاستيراد

- شغّل البرنامج النصي باستخدام زر التشغيل (F5). إذا لم تظهر أي أخطاء، فهذا يعني أن عملية الاستيراد ناجحة
- إذا ظهر خطأ مثل ModuleNotFoundError:
- تأكد من تحميل المكتبة إلى المتحكم الدقيق في ثوني،
- افتح ملف المكتبة من جهاز الكمبيوتر الخاص بك، ثم استخدم "ملف" > "حفظ باسم..." > جهاز MicroPython، ثم تأكد من ظهورها في قائمة ملفات الجهاز
- تأكد من صحة كتابة اسم المكتبة







راجع المستشعرات الثلاثة ومتطلبات التوصيل

- حدد:

نوع المستشعر (DHT11، رطوبة التربة، مستشعر الضوء)
دبوس GPIO المستخدم لإشارة المستشعر
هل يجب ضبط الدبوس على وضع الإدخال (قراءة البيانات) أو وضع الإخراج (إرسال الإشارات)

- استخدم ورقة بيانات المستشعر أو مخطط الأسلاك المختبري للتأكد من أرقام الدبوس وتوافق الجهد

اكتب كود تهيئة مستشعر DHT 11

- قم بتهيئة دبوس GPIO لمستشعر DHT11 في الكود
- استخدم فئة machine.Pin لإعداد الدبوس
- مثال على دبوس إدخال: دبوس مستشعر = machine.Pin (5,machine.Pin.IN)# GPIO5 كمدخل
- قم بتهيئة مستشعر DHT 11 باستخدام المكتبة
  - استخدم وظيفة التهيئة من مكتبة المستشعر لديك

مثال:

import dht #press enter

مستشعر = #(dht.DHT11(sensor.pin # أنشئ غرض مستشعر مرتبط بـ GPIO5

- اختبر التهيئة
  - أضف اختبار طباعة بسيطًا للتأكد من عدم وجود أخطاء:
  - اكتب ("GPIO pin and sensor initialized successfully")
- شغّل البرنامج النصي في ثوني:
  - إذا ظهرت الرسالة بدون أخطاء، فهذا يعني أن التهيئة ناجحة
  - إذا ظهرت أخطاء، تحقق من الأسلاك، أو أرقام الدبابيس، أو استيراد المكتبة

ملاحظة: عند كتابة البرنامج النصي الكامل، اتبع هيكلًا واضحًا:

- استيراد المكتبات
- تهيئة الدبابيس والمستشعرات
- استخدام حلقة لقراءة البيانات، وطباعتها، وتخزينها
- معالجة الأخطاء

```
import dht  # or your sensor's specific library
sensor_pin = machine.Pin(5, machine.Pin.IN)  # GPIO5 as input
sensor = dht.DHT11(sensor_pin)  # Create a sensor object linked to GPIO5
print("GPIO pin and sensor initialized successfully")
```

```
Shell
>>> %Run -c $EDITOR_CONTENT

MPY: soft reboot
GPIO pin and sensor initialized successfully

>>>
```

اكتب رمز تهيئة رطوبة التربة

يأتي مستشعر رطوبة التربة عادةً مع:

- مسبار (دبوسان معدنيان يدخلان في التربة)
- لوحة إشارة (مع دبابيس طاقة ومخرج)

توزيع الدبابيس:

AO (مخرج تناظري) ← يتصل بدبوس المحول التناظري الرقمي (ADC) في ESP8266 (A0)

DO (مخرج رقمي) ← اختياري، يمكن توصيله بدبوس GPIO للحصول على عتبة رقمية

- قم بتهيئة مكتبة التحويل التناظري الرقمي

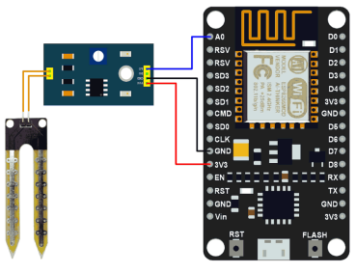
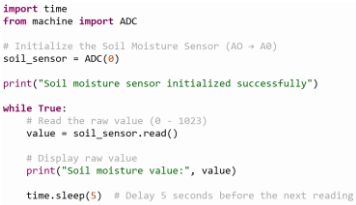
من محوّل تناظري رقمي مستورد من الجهاز

- أربط دبوس AO في المستشعر بـ A0 في ESP8266:

مستشعر التربة = ADC(0) # دبوس المحول التناظري الرقمي مثبت على A0 في ESP8266

```
from machine import ADC
# Initialize ADC on pin A0
soil_sensor = ADC(0)

while True:
    moisture_value = soil_sensor.read()  # Returns 0-1023
    print("Soil moisture:", moisture_value)
```

	• قراءة قيمة الرطوبة = soil sensor.read () • إضافة اختبار طباعة بسيط للتأكد من عدم وجود أخطاء: اكتب ("Soil moisture:", moisture_value) • تشغيل البرنامج النصي في ثوني: ○ إذا ظهرت الرسالة بدون أخطاء، تكون عملية التهيئة ناجحة ○ إذا ظهرت أخطاء، تحقق من الأسلاك، أو أرقام الدبابيس، أو استيراد المكتبة
	اكتب كود تهيئة مستشعر الضوء يُخرج مستشعر شدة الضوء إشارة تناظرية، لذلك نستخدم دبوس المحول التناظري الرقمي (A0) في وحدة ESP8266. قم بتهيئة دبوس المحول التناظري الرقمي لمستشعر الضوء في الكود: من machine استورد ADC # قم بتهيئة المحول التناظري الرقمي على الدبوس A0 مستشعر الضوء = ADC(0) # ADC هو دبوس المحول التناظري الرقمي الوحيد في وحدة ESP8266 • اختبر التهيئة القيمة = light_sensor.read() # يعيد value 0-1023 اكتب ("Light sensor initialized, value:", value) • شغّل البرنامج النصي في ثوني: إذا عُرضت قيمة بين **0** و**1023**، تكون عملية التهيئة ناجحة.
ربط المستشعرات بكود لقراءة بياناتها وعرضها	
	أوصل مستشعر DHT 11 بـ ESP8266 • حدد: نوع المستشعر (مثلاً: DHT11، مستشعر رطوبة التربة، مستشعر الضوء) توصيلات الدبابيس وأرقام دبابيس GPIO جهد المستشعر ومتطلبات الاتصال. • أوصل الدائرة المؤلفة من وحدة DHT11 ووحدة التطوير Esp8266 أو ESP32: ○ أوصل Vcc للوحدة DHT11 بجهد Vcc (3.3 فولت) لوحدة التطوير ESP8266 باستخدام السلك الأحمر ○ أوصل سلك التأريض (0 فولت) للوحدة DHT11 بسلك التأريض لوحدة التطوير ESP8266 باستخدام السلك الأسود أوصل بيانات وحدة DHT11 بـ D1 (GPIO5) لوحدة التطوير ESP8266 باستخدام السلك البرتقالي
	اكتب كود لقراءة بيانات المستشعر (مثال: مستشعر درجة الحرارة والرطوبة DHT11 أو DHT 22) • أنشئ مثيلًا للمستشعر مرتبطًا بدبوس GPIO الخاص به *import dht* *sensor = dht.DHT11(machine.Pin(5)) #orDHT22* (اختصار DHT هو مستشعر الرطوبة ودرجة الحرارة الرقمي) • اقرأ بيانات المستشعر باستخدام الطريقة التالية من مكتبة المستشعر: • لـ DHT11: sensor.measure() temperature = sensor.temperature() humidity = sensor.humidity() • اعرض بيانات المستشعر في REPL باستخدام دالة اكتب () لعرض بيانات المستشعر بوضوح. مثال:

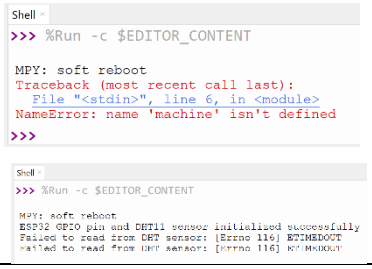
	اكتب ("Temperature:", temperature, "°C") اكتب ("Humidity:", humid, "%") • ضمّن في حلقة للقراءة المستمرة (حلقة لا نهائية) • ضع كود القراءة والعرض داخل حلقة while True: while True: *sensor.measure()* *temperature = sensor.temperature()* *humidity = sensor.humidity()* *اكتب ("Temperature:", temperature, "°C")* *اكتب ("Humidity:", humidity, "%")* time.sleep(5) نصيحة: نحتاج إلى منح المستشعرات وقتًا للاستقرار بين القراءات. لهذا السبب نستخدم: time.sleep(5) # انتظر 5 ثوانٍ قبل القراءة التالية لإبطاء الحلقة. هذا يُمكّن المراقبة المستمرة في REPL. • احفظ النص البرمجي وانقر على **تشغيل (F5)** • في حالة حدوث أخطاء، انصح المتعلمين: ○ بالتحقق من التوصيلات وشيفرة التهيئة ○ بالتحقق من أسماء وظائف مكتبة المستشعرات الدقيقة • احفظ البرنامج النصي وقم بتشغيله في ثوني
	**توصيل مستشعر رطوبة التربة بـ ESP8266** • حدّد: ○ نوع المستشعر: مستشعر رطوبة التربة (مخرج تناظري) ○ توصيلات الدبابيس: VCC، GND، AO (مخرج تناظري) ○ جهد المستشعر: 3.3 فولت ○ الاتصال: إشارة تناظرية ← دبوس المحول التناظري الرقمي (A0) ESP8266 • أوصل الدائرة المكونة من وحدة مستشعر رطوبة التربة ولوحة تطوير ESP8266: • أوصل كابل VCC لوحدة رطوبة التربة بجهد 3.3 فولت (Vcc) لوحدة التطوير ESP8266 باستخدام سلك أحمر • أوصل كابل GND لوحدة رطوبة التربة بسلك أسود • أوصل كابل AO (المخرج التناظري) لوحدة رطوبة التربة بسلك A0 (دبوس المحول التناظري الرقمي) لوحدة التطوير ESP8266 باستخدام سلك برتقالي
	**اكتب كود لقراءة مستشعر رطوبة التربة** • قم بتهيئة غرض المستشعر في الكود. يوفر مستشعر رطوبة التربة مخرجاً تناظريًا، لذلك نستخدم دبوس المحول التناظري الرقمي (A0) في وحدة التحكم ESP8266. من machine استورد ADC Soil_sensor = ADC(0) # أنشئ غرض مستشعر تربة مرتبط بـ A0 • اقرأ البيانات من المستشعر، واستخدم وظيفة اقرأ() من فئة المحول التناظري الرقمي لجمع البيانات القيمة= soil_sensor.read() # تعيد 0-1023 • اعرض بيانات المستشعر في REPL، واستخدم وظيفة اكتب () لعرض بيانات المستشعر. اكتب ("Soil moisture value:", value) • ضمّن في حلقة للقراءة المستمرة (حلقة لا نهائية) • لفّ كود القراءة والعرض داخل حلقة while True: *import time* من machine استورد ADC # قم بتهيئة مستشعر رطوبة التربة (AO → A0)

```
ADC(0) =Soil_sensor
اكتب ("تم تهيئة مستشعر رطوبة التربة بنجاح")
:while True
# قراءة القيمة الخام (0 - 1023)
القيمة= ()soil_sensor.read
# عرض القيمة الخام
اكتب (“Soil moisture value:”,value)
time.sleep(5)  # تأخر 5 ثوانٍ قبل القراءة التالية
```

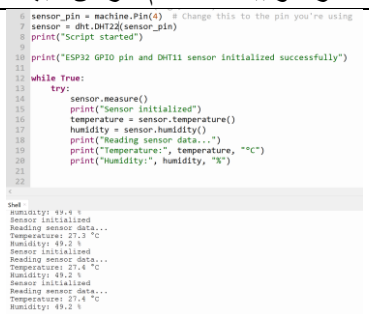
هذا التأخير يعطي المستشعر وقتًا للاستقرار ويمنع إرسال بيانات REPL بشكل عشوائي.

- تشغيل الكود واختباره
  - احفظ النص البرمجي وانقر على زر التشغيل (F5) في ثوني.
  - في حال حدوث أخطاء، انصح المتعلمين:
  - بالتحقق من الأسلاك (VCC → 3.3V، GND → GND، AO → A0)
  - بالتحقق من استخدام طريقة المحول التناظري الرقمي الصحيحة (read())
  - بالتأكد من استخدام A0 فقط للإدخال التناظري (ESP8266 يحتوي على محول تناظري رقمي واحد فقط)

تصحيح أخطاء جمع البيانات



قد تنشأ الأخطاء من مصادر مختلفة عند العمل مع أدوات التحكم الدقيقة والمستشعرات. بعض هذه الأخطاء متعلق بالأجهزة، مثل التوصيلات المفكوكة أو التالفة، أو استخدام دبابيس GPIO خاطئة، أو توصيلات دبابيس غير صحيحة، أو عدم كفاية مصدر الطاقة للمستشعر. أما البعض الآخر ينشأ من الكود نفسه، بما في ذلك أخطاء الصياغة مثل فقدان النقطتين أو الأخطاء المطبعية، أو أرقام دبابيس GPIO غير الصحيحة، أو نقص أو استيراد مكتبات خاطئة. وأخيرًا، تحدث أخطاء وقت التشغيل أثناء التنفيذ. قد تشمل هذه الأخطاء مشاكل مثل *ModuleNotFoundError*، أو *OSError* عند محاولة القراءة من المستشعر، أو ببساطة عدم عرض البيانات في وحدة تحكم REPL.



استخدم عبارات الطباعة لتصحيح الأخطاء

- أضف عبارات طباعة لتحديد مكان فشل البرنامج النصي:

اكتب *("Script began")*
اكتب *("Sensor initialized")*
اكتب *("Reading sensor data...")*

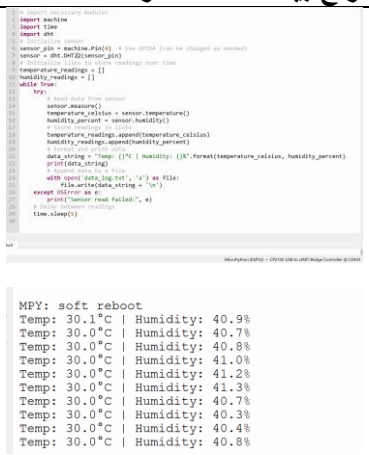
تحقق من توصيلات الأجهزة

- تأكد من توصيل أسلاك المستشعر بدبابيس GPIO الصحيحة
- تأكد من أن خطوط الطاقة (VCC) والأرضي (GND) موصولة بإحكام باستخدام جهاز القياس المتعدد
- تأكد من توصيل المتحكم الدقيق بشكل صحيح واكتشافه بواسطة الكمبيوتر

تحقق من إعدادات دبابيس GPIO في عمليات استيراد الكود والمكتبات

- تأكد من تطابق أرقام دبابيس GPIO في الكود مع التوصيلات المادية
- تحقق من الوضع الصحيح (مثل: machine.Pin.IN للإدخال)
- تأكد من استيراد المكتبات الضرورية دون أخطاء إملائية:

```
import machine
import time
import dht
```

	• تأكد من تحميل المكتبات الخارجية (مثل: dht.py، ssd1306.py) إلى المتحكم الدقيق إذا لزم الأمر **معالجة أخطاء وقت التشغيل الخاصة بالمستشعر** • إصلاح أخطاء OSError أو فشل القراءة من خلال: ○ إضافة تأخيرات مناسبة باستخدام time.sleep() ○ تغليف قراءات المستشعر داخل كتل try-except: *sensor.measure()* *temp = sensor.temperature()* *اكتب (temp)* *except OSError as e:* *اكتب ("Sensor read error:", e)*
	**وثّق المشكلة وأصلحها** • احتفظ بسجل بسيط لتصحيح أخطاء: ○ الخطأ الذي تمت ملاحظته ○ الإصلاح الذي تم تطبيقه
استخراج بيانات المستشعر	
 	**خزّن البيانات في متغيرات** • استخدم أسماء متغيرات واضحة ووصفية: *temperature_celsius = sensor.temperature()* *humidity_percent = sensor.humidity()*
 	**جمع البيانات على مدار الوقت** • شرح كيفية تخزين البيانات في قوائم لمراقبة الاتجاهات: *قراءات درجة الحرارة = []* *قراءات الرطوبة = []* *temperature_readings.append(temperature_celsius)* *humidity_readings.append(humidity_percent)*

تنسيق البيانات للوضوح

- إنشاء سلاسل إخراج قابلة للقراءة:

data_string = "Temperature: {}°C | Humidity: {}%".format(temperature_celsius, humid_percent)

اكتب (data_string)

```
import time
from machine import ADC

# Initialize the Soil Moisture Sensor (AO → A0)
soil_sensor = ADC(0)

# Create a list to store moisture readings
moisture_readings = []

print("Collecting soil moisture data...")

while True:
    # Read current value (0-1023)
    value = soil_sensor.read()

    # Store reading in the list
    moisture_readings.append(value)

    # Display reading
    print("Soil moisture value:", value)
    print("All readings so far:", moisture_readings)

    # Wait 5 seconds before the next reading
    time.sleep(5)
```

```
Soil Moisture Value: 420
Soil Moisture Value: 715
Soil Moisture Value: 310
```

جهّز البيانات للتخزين أو النقل

- اكتب البيانات إلى ملفات

باستخدام open('data_log.txt', 'a') كملف:

file.write(data_string + '\n')

سيتم إضافة كل قراءة إلى data_log.txt على نظام الملفات الداخلي لـ ESP8266.
تُحفظ كل قيمة جديدة في سطر جديد.

ملاحظة: ESP8266 لها سعة تخزين محدودة. يمكن تحقيق تسجيل طويل الأمد عادةً عبر نقل البيانات إلى بطاقة SD أو إرسالها عبر Wi-Fi (MQTT/HTTP).

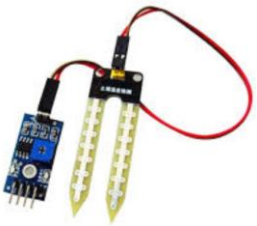
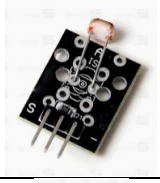
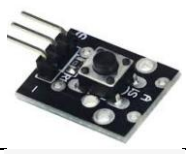
```
import time
from machine import ADC

# Initialize the Soil Moisture Sensor (AO → A0)
soil_sensor = ADC(0)

print("Logging soil moisture data...")

while True:
    # Read soil moisture value
    value = soil_sensor.read()

    # Create readable string
    data_string = "Soil Moisture Value: {}".format(value)
    print(data_string)

    # Write data to a log file
    with open('data_log.txt', 'a') as file:
        file.write(data_string + '\n')

    time.sleep(5)  # Log every 5 seconds
```

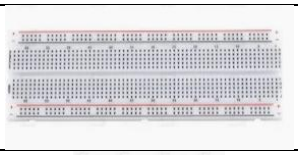
3.1.2 الادوات والمواد

الادوات	الصورة	الكمية [2]	الملاحظة
جهاز كمبيوتر شخصي مع شاشة ولوحة مفاتيح وفأرة (مثبت عليه نظام تشغيل ويندوز) أو كمبيوتر محمول مزود بمنفذ USB مثبت عليه (Python 3.8+، وThonny IDE، وبرامج تشغيل USB: برامج تشغيل CP210x / CH340، وصلاحيات إدارية لتثبيت التعريفات/البرمجيات)		1 لكل محطة	
اتصال بالإنترنت مع بيانات اعتماد شبكة WiFi		1 لكل محطة	
مقياس متعدد		1 لكل محطة	
حصيرة مضادة للكهرباء الساكنة آمنة ضد التفريغ الكهروستاتيكي		1 لكل محطة	
بطارية محمولة		1 لكل محطة	
محول تيار مستمر ٥ فولت بمنفذ USB صغير		1 لكل محطة	
صناديق تخزين للمكونات		1 لكل محطة	

[2] إن الكمية المشار إليها للورشة تعني إمكانية استخدام المنتج لوحدات مختلفة في الورشة ذاتها، مما يعني أنه لا يُمكن تسليم وحدتان تستخدمان المنتج في نفس الوقت.
تحذير! قبل شراء المنتج، يُرجى التحقق من وجوده في الورشة لتجنب أي تكرار غير ضروري. كما يجب مراجعة القائمة والتحقق منها من قِبل خبير مُختص.

مجموعة أدوات أساسية (مفكات براغي، كماشة، قاطع أسلاك)		1 لكل محطة	

المواد والمستهلكات	الصورة	الكمية (لكل محطة)	الملاحظة
لوحة تطوير ESP8266 (مثل NodeMCU وWemos D1 Mini)		1	
كابل بيانات Micro USB		1	(تأكد من أنه كابل بيانات، وليس كابل شحن فقط)
وحدة استشعار DHT22/DHT11		1	
وحدة استشعار رطوبة التربة		1	
وحدة استشعار الإضاءة		1	
مصابيح LED		5	
مقاومات 470 أوم، 1 كيلو أوم و 10 كيلو أوم		10 لكل منها	
وحدة أزرار الضغط		2	
مفتاح التشغيل/الإيقاف		2	

لوحة تجارب		1	
أسلاك توصيل جامبر (أنثى-ذكر، ذكر-ذكر)		رزمة لكل منها	
بطارية ليثيوم أيون قابلة لإعادة الشحن 18650		1	
حامل بطارية ليثيوم أيون قابلة لإعادة الشحن 18650 مع سلك		1	

3.2 معايير التقييم- الوحدة 2

3.2.1 التقييم التكويني

اسم المتدرب:	التاريخ:	
القدرات/ناتج التعلّم	**المعايير لوحدة الكفاية 2- كتابة نصوص جمع بيانات إنترنت الأشياء**	**نعم/لا**
كتابة نصوص بلغة بايثون لجمع البيانات من مستشعرات إنترنت الأشياء باستخدام مايكرو بايثون	هل فتح المرشح بيئة التطوير المتكاملة (IDE) الخاصة بثوني بشكل صحيح وقام بتوصيل ESP8266/ESP32 عبرUSB ؟	
	هل تحقق المرشح من اكتشاف ثوني للجهاز ومن عمل REPL ؟	
	هل أنشأ المرشح نصًا برمجيًا جديدًا بصيغة py. وحفظه بتسمية ملف واضحة؟	
	هل استورد المرشح المكتبات اللازمة (الخاصة بالجهاز، والوقت، والمستشعر) في أعلى النص البرمجي؟	
	هل كتب المرشح سطر اختبار أولي (اكتب ("IoT Data Acquisition Script Running")) وشغّل النص البرمجي بنجاح في ثوني؟	
	هل حدد المرشح المكتبات الأساسية والمستشعرات المطلوبة واستوردها بشكل صحيح؟	
	هل ختبر المرشح عبارات الاستيراد وعالج أخطاءً مثل ModuleNotFoundError في حال حدوثها؟	
	هل راجع المرشح أنواع المستشعرات، ودبابيس GPIO ، وأنماط الدبابيس قبل البرمجة؟	
ربط المستشعرات برمز لقراءة بياناتها وعرضها	هل كتب المرشح كود تهيئة مستشعرDHT11 ، بما في ذلك إعداد دبوس GPIO ؟	
	هل قام المرشح بتهيئة غرض مستشعر DHT11 باستخدام مكتبة المستشعرات واختباره؟	
	هل كتب المرشح كود تهيئة مستشعر رطوبة التربة باستخدام المحول التناظري الرقمي وقيم القراءة التجريبية؟	
	هل كتب المرشح كود تهيئة مستشعر الضوء باستخدام المحول التناظري الرقمي وقيم القراءة التجريبية؟	
	هل ربط المرشح مستشعر DHT11 بوحدة التحكم ESP8266 باستخدام دبابيس VCC و GND وبيانات صحيحة؟	
	هل أنشأ المرشح كود لقراءة بيانات مستشعر DHT11 وعرض درجة الحرارة والرطوبة فيREPL ؟	
	هل دمج المرشح قراءة المستشعر وعرضه في حلقة مستمرة مع تأخيرات مناسبة؟	
	هل ربط المرشح مستشعر رطوبة التربة بوحدة التحكم ESP8266 بشكل صحيح وكتب كود لقراءة وعرض القيم التناظرية بشكل مستمر؟	
تصحيح أخطاء جمع البيانات	هل استخدم المرشح عبارات الطباعة بفعالية لتحديد مواضع فشل البرنامج النصي؟	
	هل تحقق المرشح من توصيلات الأجهزة، وضمن توصيل المستشعرات بدبابيس GPIO وخطوط الطاقة الصحيحة؟	
	هل تحقق المرشح من تطابق تكوينات دبابيس GPIO في الكود مع الأسلاك الفعلية وعمليات استيراد المكتبة الصحيحة؟	
	هل عالج المرشح أخطاء وقت التشغيل الخاصة بالمستشعر، مثل استخدام try-except لخطأ zSError أو حالات فشل القراءة الأخرى؟	
	هل لاحظ المرشح الأخطاء وطبّق الإصلاحات في سجل تصحيح الأخطاء؟	
استخراج بيانات المستشعر	هل خزّن المرشح قراءات المستشعر في متغيرات وصفية (مثل: درجة الحرارة المئوية، نسبة الرطوبة)؟	
	هل جمّع المرشح قراءات متعددة على مدار الوقت وخزّنها في قوائم لمراقبة الاتجاهات؟	
	هل نسّق المرشح البيانات لإخراج واضح باستخدام سلاسل نصية قبل الطباعة؟	
	هل حفظ المرشح البيانات في ملف (data_log.txt) على ESP8266 أم جهزّها للإرسال؟	
	هل ضمن المرشح إضافة البيانات المحفوظة وتنسيقها للاستخدام لاحقًا؟	

3.2.2 التقييم الختامي أو التجميعي

اسم المتدرب:	**التاريخ:**	
اسم المقيّم:	**التاريخ:**	
يعتبر المتدرب مؤهلاً إذا أثبت أداء أفضل الممارسات في جميع المعايير التالية للتقييم المتعلقة بوحدة الكفاية 2 – **كتابة نصوص جمع بيانات إنترنت الأشياء**		

معيار التقييم	المؤهّل	نعم	لا (*)
إعداد البرنامج النصي وتهيئته	يجب أن يكون المرشح قادرًا، في غضون 20 دقيقة أو أقل، على إنشاء وحفظ نص برمجي MicroPython في بيئة التطوير المتكاملة Thonny IDE، واستيراد المكتبات المطلوبة، وتهيئة دبابيس GPIO بالأوضاع الصحيحة، وتهيئة المستشعر المتصل باستخدام أوامر المكتبة المناسبة دون أخطاء.	☐	☐
جمع البيانات وعرضها	يجب أن يكون المرشح قادرًا، في غضون 20 دقيقة أو أقل، على تنفيذ حلقة while في نص برمجي MicroPython لقراءة بيانات المستشعر باستمرار باستخدام الطرق الصحيحة، وعرض البيانات المستخرجة بوضوح في REPL باستخدام عبارات "اكتب ()"، وإضافة فترة زمنية مناسبة بين القراءات لضمان استقرار التشغيل.	☐	☐
توصيل الأجهزة والتحقق منها	يجب أن يكون المرشح قادرًا، في غضون 10 دقائق أو أقل، على توصيل مستشعر إنترنت الأشياء بالمتحكم الدقيق بشكل صحيح وفقًا لمخطط الأسلاك، والتحقق من التوصيلات المادية والطاقة، والتأكد من أن المتحكم الدقيق يكتشف إعداد المستشعر لجمع مزيد من البيانات.	☐	☐
تهيئة الكود وقراءته وعرضه	يجب أن يكون المرشح قادرًا، في غضون 15 دقيقة أو أقل، على كتابة وتنفيذ نص برمجي بلغة MicroPython يُهيئ دبابيس GPIO والمستشعر، ويقرأ البيانات باستخدام أساليب المكتبة المناسبة، ويعرض بيانات المستشعر المباشرة بوضوح في REPL دون أخطاء.	☐	☐
تصحيح أخطاء جمع البيانات	يجب أن يكون المرشح قادرًا، في غضون 10 دقائق أو أقل، على تحديد الأخطاء التي تحدث أثناء جمع بيانات المستشعر، وتطبيق خطوات تصحيح الأخطاء المنهجية، بما في ذلك فحص اتصالات الأجهزة، والتحقق من تكوينات GPIO، وفحص استيرادات المكتبة، واستخدام عبارات الطباعة، وتطبيق معالجة try-except، والتأكد من تنفيذ النص البرمجي دون أخطاء مع عرض بيانات المستشعر المباشرة الصحيحة في REPL.	☐	☐
استخراج بيانات المستشعر	يجب أن يكون المرشح قادرًا، في غضون 10 دقائق أو أقل، على استخراج بيانات المستشعر من خلال قراءتها باستخدام أساليب المكتبة المناسبة، وتخزين البيانات في متغيرات ذات تسمية واضحة، وتنسيق البيانات لضمان الوضوح، وعرضها في REPL لتأكيد قراءات صحيحة ومستقرة مناسبة للتحليل أو التسجيل أو معالجة إنترنت الأشياء.	☐	☐
المعيار (*)	الدليل على عدم الامتثال إن وجد (**)		
توقيع المقيّم:	التاريخ:		
توقيع المتدرب:	التاريخ:		

4 وحدة الكفاية 3: تطوير البرامج للتحكّم في الأجهزة

4.1 معيار التدريب- الوحدة 3

4.1.1 المبادىء التوجيهية للتدريب

ناتج القدرات/التعلّم:	الأدوات المطلوبة:	المواد والمستهلكات المطلوبة:
• كتابة برامج تحكم بلغة بايثون للمشغلات والأجهزة الطرفية • اختبار منطق التحكم لتشغيل الأجهزة بناءً على الشروط • استخدام منافذ الدخول/الخروج العامة GPIOs ومخرجات تعديل عرض النبضة PWM • تحسين منطق التحكم باستخدام اتصالات الأجهزة مع ضمان كفاءة الطاقة **موقع التدريب:** غرفة داخلية بمساحة تقريبية 4x5 متر، مجهزة بـ: • نافذة واسعة • طفاية حريق • مفاتيح ومقابس كهربائية متعددة • أثاث خشبي ثابت مناسب لأنشطة مختبر الإلكترونيات • مكاتب ومناضد لوضع أجهزة الكمبيوتر، والأدوات، والمواد اللازمة للتطبيق العملي • خزانة وأدراج لتنظيم وتخزين المكونات الإلكترونية ومواد المختبر منطقة تربة (أو حاويات تربة) داخل الغرفة لاختبار أجهزة استشعار رطوبة التربة	• كمبيوتر شخصي وشاشة • اتصال بالإنترنت • مقياس متعدد • محول تيار مستمر 12 فولت • محول تيار مستمر 5 فولت بمنفذ ميكرو USB • صناديق تخزين للمكونات • مجموعة أدوات أساسية (مفكات براغي، كماشة، قاطع أسلاك) • بطارية 12 فولت، 3 آمبير أو أكثر • حصيرة مضادة للكهرباء الساكنة وآمنة ضد التفريغ الكهروستاتيكي	• لوحة تطوير ESP8266 (مثل NodeMCU وWemos D1 Mini) • كابل بيانات Micro USB (تأكد من أنه كابل بيانات، وليس مجرد كابل شحن) • وحدة استشعار DHT22/DHT11 • وحدة استشعار مستوى الماء • وحدة استشعار رطوبة التربة • وحدة استشعار الإضاءة المحيطة • بطارية ليثيوم أيون قابلة لإعادة الشحن 18650 مع حامل • مصابيح LED RGB • مصابيح LED • مقاومات • جرس إنذار • وحدة أزرار الضغط • مفاتيح • محرك سيرفو • محرك تيار مستمر • وحدة جسر H • وحدة مرحلات • مضخة • لوحة تجارب • أسلاك توصيل جامبر (أنثى-ذكر، ذكر-ذكر) • وحدة تزويد الطاقة للوحة التجارب 3.3 فولت 5 فولت

كتابة برامج تحكم بلغة بايثون للمشغلات والأجهزة الطرفية

افتح بيئة التطوير

- شغّل بيئة التطوير المتكاملة Thonny IDE.
- وصّل لوحة تطوير ESP8266 عبر منفذ USB بالكمبيوتر.
- تأكد من توصيل ESP8266، ومن التعرّف على منفذ COM.
- افتح ملف بايثون جديد لبرنامج التحكم.

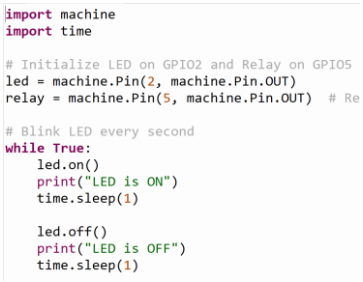
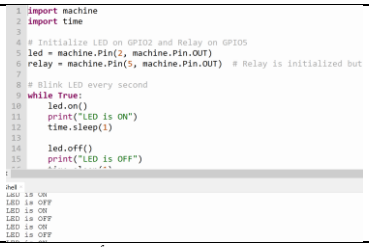
استيراد المكتبات المطلوبة

تُستخدم مكتبة Machine للتحكم في GPIO وPWM. وتُستخدم مكتبة Time للتحكم في التأخيرات والتوقيت.

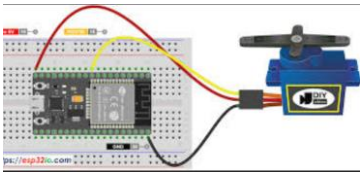
- ابدأ الكود بكتابة:

import machine

```
import machine
import time
led = machine.Pin(2, machine.Pin.OUT)
relay = machine.Pin(5, machine.Pin.OUT)
```

	import time • انقر على "ملف" ثم اختر "حفظ باسم". احفظ البرنامج النصي باسم control_actuator.py للتوضيح • قم بتحديد وتهيئة دبابيس GPIO • اضبط دبابيس GPIO التي ستتحكم في المشغلات كمخرجات. مثال: *led = machine.Pin(2, machine.Pin.OUT)* للمرحل أو مشغل المحرك: *relay = machine.Pin(5, machine.Pin.OUT)* • استخدم أسماء متغيرات واضحة ووصفية لكل مشغل
 	**اكتب منطق التحكم الأساسي** • ابدأ بتشغيل المُشغِّل وإيقافه: led.on() # يُشغِّل مصباح LED time.sleep(1) # ينتظر ثانية واحدة led.off() # يُطفئ مصباح LED time.sleep(1) • اختبر ذلك بإنشاء حلقة وامضة بسيطة: *while True*: *led.on* *اكتب ("LED is ON")* *time.sleep(1)* *led.off()* *اكتب ("LED is OFF")* *time.sleep(1)* نصيحة: لإيقاف الحلقات اللانهائية أثناء التنفيذ في ثوني، استخدم Ctrl+C.
 	**احفظ النص البرمجي وحمّله** • احفظ النص البرمجي باسم واضح • شغّل النص البرمجي من ثوني لمراقبة استجابة المُشغّل
اختبار منطق التحكم لتشغيل الأجهزة بناءً على الشروط	
 	**أنشئ البرنامج النصي للاختبار المنطقي** • افتح بيئة التطوير المتكاملة ثوني وأنشئ ملف test_control_logic.py. • استورد المكتبات اللازمة: import time • قم بتهيئة "حالة جهاز" مُحاكاة: "device_state = "OFF • اكتب حلقة لاختبار المنطق: count = 0 while True: if count % 2 == 0: device_state = "ON" غيرها: device_state = "OFF" اكتب ("Loop:", count, "Device State:", device_state)

	count += 1 time.sleep (1) • نفّذ البرنامج النصي • لاحظ مُخرجات REPL • أضف تعليقات للتوضيح البنية: if condition: #action غيرها: #alternative action
	تعديل الشروط للممارسة • إضافة شروط متداخلة (مثل حالات متعددة) • محاكاة قيمة مستشعر: قيمة_المستشعر = 25 إذا كانت قيمة_المستشعر > 20: حالة_الجهاز = "تشغيل" وإلا: حالة_الجهاز = "إيقاف" أسفل النموذج نصيحة: من الأفضل استخدام أسماء متغيرات واضحة ووصفية، على سبيل المثال، درجة_الحرارة_مئوية.
استخدام منافذ الدخول/الخروج العامة GPIOs ومخرجات تعديل عرض النبضة PWM	
	اكتب كود التحكم في منفذ GPIO • افتح بيئة التطوير المتكاملة ثني وأنشئ نصًا برمجيًا جديدًا (gpio_pwm_control.py) • استورد المكتبات: *import machine* *import time* • تهيئة دبوس GPIO كمخرج: led = machine.Pin(2, machine.Pin.OUT) • التحكم في الدبوس لتشغيل وإيقاف مؤشر LED: *led.on()* *time.sleep(1)* *led.off()* *time.sleep(1)* • استخدم عبارات "اكتب()" للإشارة إلى الإجراء في حال لم يتم توصيل الأجهزة.
	اكتب كود التحكم في تعديل عرض النبضة (PWM) • اشرح تعديل عرض النبضة (PWM) كطريقة لمحاكاة المخرجات التناظرية من خلال تغيير دورة عمل الإشارات الرقمية. • هيّىء تعديل عرض النبضة (PWM) على دبوس GPIO: pwm = machine.PWM(machine.Pin(2)) pwm.freq(1000) # اضبط التردد على 1 كيلوهرتز • غيّر دورة العمل للتحكم في الكثافة أو السرعة: for duty in range(0, 1024, 200) pwm.duty(duty) اكتب ("Duty cycle:", duty)

	time *import machine*(0.5) *import time* .sleep • شغّل البرنامج النصي ولاحظ المخرجات في REPL • جرّب مثالاً آخر: تعتيم مصباح LED باستخدام تعديل عرض النبضة (PWM) pwm = machine.PWM(machine.Pin(2)) pwm.freq(1000) # تردد 1 كيلوهرتز *.while True* for duty in range :(100 ,1024 ,0) # زيادة السطوع pwm.duty(duty) اكتب ("Duty:", duty) time.sleep(0.2) for duty in range (100- ,1- ,1023) # تقليل السطوع pwm.duty(duty) اكتب ("Duty:", duty) time.sleep(0.2)
تحسين منطق التحكم باستخدام اتصالات الأجهزة مع ضمان كفاءة الطاقة	
	**إعداد الأجهزة** • توصيل ESP8266 بالكمبيوتر المحمول ○ استخدم كابل بيانات USB (ليس كابل شحن فقط) لتوصيل لوحة ESP8266 بالكمبيوتر المحمول • افتح ثوني • اذهب إلى الأدوات > الخيارات > المفسّر: ○ حدد MicroPython (ESP8266) ○ اضبط منفذ COM الصحيح (بالعادة يتم اكتشافه تلقائيًا) • توصيل الدائرة على لوحة التجارب ○ مثال على مخرج GPIO: أوصل GPIO5 (D1) ← الطرف الأول للمقاوم (Ω220) أوصل الطرف الثاني للمقاوم بالطرف الموجب لـ LED (الدبوس الطويل) أوصل الطرف الثاني للدبوس الثنائي لـ LED (الطرف القصير) بالأرضي (0 فولت) ○ مثال على تعديل عرض النبضة (PWM): أوصل GPIO14 (D5) ← سلك إشارة السيرفو سيرفو VCC ← 3.3 فولت أو خارجي 5 فولت (حسب الطراز) سيرفو GND ← GND ○ مثال على جهاز الإدخال: زر ضغط: دبوس واحد إلى GPIO4 (D2)، والآخر إلى الأرضي (GND) (استخدم السحب الداخلي في الكود)
	اكتب منطق التحكم الأولي • افتح برنامج ثوني لكتابة نص برمجي بلغة MicroPython مع منطق التحكم GPIO وPWM. مقتطف توضيحي: *from machine import Pin, PWM* *from time import sleep* *led = Pin(5, Pin.OUT) # GPIO5* *servo = PWM(Pin(14), freq=50) # GPIO14 for PWM*

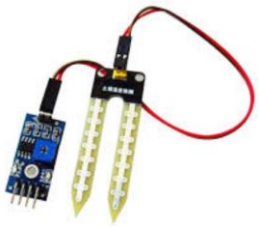
	button = Pin(4, Pin.IN, Pin.PULL_UP) *while True:* *if button.value() == 0:* *led.on()* *servo.duty(40) # Example position* *else:* *led.off()* *servo.duty(75)* *sleep(0.1)* • احفظ الملف باسم main.py في ثوني • انقر فوق "تشغيل" أو اضغط على F5 لتحميل الكود • تأكد من أن ESP8266 ينفذ المنطق تلقائيًا عند إعادة التشغيل لاحظ أن الكود سيُعاد تشغيله تلقائيًا عند إعادة ضبط اللوحة أو تشغيلها
	عملية التحسين • راقب سلوك الأجهزة: o هل يتفاعل مؤشر LED كما هو متوقع؟ o هل يصل محرك السيرفو إلى الزاوية الصحيحة؟ o هل تتم الانتقالات بسلاسة؟ • إضافة فترات تأخير لمعالجة اهتزاز الأزرار • عاير زوايا محرك السيرفو • استخدم عبارات الطباعة في Thonny REPL لتصحيح الأخطاء مباشرةً • اضبط عتبات المنطق أو قيم دورة العمل • عدّل الكود وأعد النشر • أعد التحقق من السلوك مع كل تغيير • استخدم التعليقات لتوثيق التغييرات أو التفسيرات المنطقية • احرص على وجود إصدارات متعددة (مثل الإصدار 1، الإصدار 2، ...) لتتبع تقدم التطوير

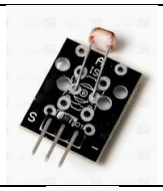
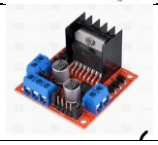
4.1.2 الادوات والمواد

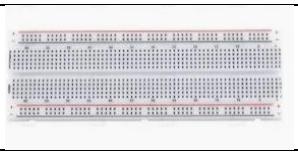
الادوات	الصورة	الكمية [3]	الملاحظة
جهاز كمبيوتر شخصي مع شاشة ولوحة مفاتيح وفأرة (مثبت عليه نظام تشغيل ويندوز OS) أو كمبيوتر محمول مزود بمنفذ USB مثبت عليه (Python 3.8+، وThonny IDE، وبرامج تشغيل USB: برامج تشغيل CP210x / CH340، وصلاحيات إدارية لتثبيت التعريفات/البرمجيات، مع بيانات اعتماد شبكة WiFi)		1 لكل محطة	
اتصال بالإنترنت		1 لكل فصل دراسي	
مقياس متعدد		1 لكل محطة	
بطارية محمولة		1 لكل محطة	
محول تيار مستمر ٥ فولت بمنفذ USB صغير		1 لكل محطة	
محول تيار مستمر 12 فولت		1 لكل محطة	

[3] إن الكمية المشار إليها للورشة تعني إمكانية استخدام المنتج لوحدات مختلفة في الورشة ذاتها، مما يعني أنه لا يُمكن تسليم وحدتان تستخدمان المنتج في نفس الوقت.
تحذير! قبل شراء المنتج، يُرجى التحقق من وجوده في الورشة لتجنب أي تكرار غير ضروري. كما يجب مراجعة القائمة والتحقق منها من قِبل خبير مُختص.

صناديق تخزين للمكونات		1 لكل محطة	
بطارية 12 فولت1، 3 آمبير أو أكثر + ملاقط		1 لكل فصل دراسي	
حصيرة مضادة للكهرباء الساكنة آمنة ضد التفريغ الكهروستاتيكي		1 لكل محطة	

المواد والمستهلكات	الصورة	الكمية (لكل محطة)	الملاحظة
لوحة تطوير ESP8266 (مثل NodeMCU وWemos D1 Mini)		1	
كابل بيانات Micro USB		1	(تأكد من أنه كابل بيانات، وليس كابل شحن فقط)
وحدة استشعار DHT22/DHT11		1	
وحدة استشعار مستوى الماء		1	
وحدة استشعار رطوبة التربة		1	

وحدة استشعار الإضاءة		1	
RGB LED		1	
مصابيح LED		5	
مقاومات 470 أوم، 1 كيلو أوم و 10 كيلو أوم		10 لكل منها	
جرس الكتروني		1	
وحدة أزرار الضغط		2	
مفتاح التشغيل/الإيقاف		2	
موتور سيرفو		1	
محرك تيار مستمر		1	
وحدة التتابع		1	
وحدة جسر H		1	
مضخة 12 فولت		1 لكل فصل دراسي	

لوحة تجارب		1	
وحدة إمداد طاقة لوحة التجارب 3.3 فولت، 5 فولت		1	
أسلاك توصيل جامبر (أنثى-ذكر، ذكر-ذكر)		رزمة لكل منها	

4.2 معايير التقييم- الوحدة 3

4.2.1 التقييم التكويني

اسم المتدرب:	التاريخ:	
القدرات/ناتج التعلّم	**المعايير لوحدة الكفاية 3 – تطوير البرامج للتحكّم بالأجهزة**	**نعم/لا**
كتابة برامج تحكم بلغة بايثون للمشغلات والأجهزة الطرفية	هل فتح المرشح بيئة التطوير المتكاملة (IDE) الخاصة بـ Thonny وأولها بلوحة ESP8266بشكل صحيح عبرUSB ؟	
	هل أكّد المرشح اكتشاف ESP8266 والتعرّف على منفذ COM الصحيح؟	
	هل أنشأ المرشح ملف بايثون جديد لبرنامج التحكم وحفظه باسم واضح (مثل control_actuator.py)؟	
	هل استورد المرشح المكتبات المطلوبة (machine and time) في أعلى البرنامج النصي؟	
	هل حدد المرشح وهيّأ دبابيس GPIO للمشغلات باستخدام بناء الجملة الصحيح وأسماء المتغيرات الوصفية؟	
	هل كتب المرشح منطق تحكم أساسي لتشغيل وإيقاف المشغل باستخدام دالة time.sleep() بشكل صحيح؟	
	هل أنشأ المرشح حلقة وميض بسيطة واختبرها للتحقق من استجابة المشغل؟	
	هل قام المرشح بحفظ النص وتحميله لمراقبة عمل المشغّل؟	
اختبار منطق التحكم لتشغيل الأجهزة بناءً على الشروط	هل أنشأ المرشح نصًا برمجيًا بلغة بايثون لاختبار منطق التحكم (test_control_logic.py)؟	
	هل استورد المرشح المكتبات اللازمة وهيأ حالات الجهاز التي جرت محاكاتها؟	
	هل كتب المرشح حلقة لاختبار الشروط المنطقية وعرض المخرجات في REPL ؟	
	هل استخدم المرشح عبارات if-else بشكل صحيح لتبديل حالات الجهاز بناءً على الشروط؟	
	هل عدّل المرشح الشروط المتداخلة واختبرها أو قام بمحاكاة قيم المستشعرات للتدريب؟	
	هل استخدم المرشح أسماء متغيرات وصفية لتحسين قابلية قراءة الكود وصيانته؟	
استخدام منافذ الدخول/الخروج العامة GPIOs ومخرجات تعديل عرض النبضة PWM	هل قام المرشح بتهيئة دبابيس GPIO كمخرجات وأجهزة تحكم (مثل مصابيح LED) باستخدام وظيفة التشغيل (on) ووظيفة الإيقاف (off) ؟	
	هل كتب المرشح عبارات الطباعة للإشارة إلى الإجراءات عند عدم توصيل الأجهزة؟	
	هل قام المرشح بتهيئة تعديل عرض النبضة (PWM) على دبابيس GPIO وضبط التردد بشكل صحيح؟	
	هل غيّر المرشح دورة العمل للتحكم في شدة أو سرعة المشغلات؟	
	هل كتب المرشح حلقة واختبرها لزيادة وخفض مخرج تعديل عرض النبضة (PWM) تدريجيًا (مثل تعتيم مصابيح LED)؟	
	هل لاحظ المرشح مخرجات تعديل عرض النبضة (PWM) في REPL وأكّد السلوك الصحيح؟	
تحسين منطق التحكم باستخدام اتصالات الأجهزة مع ضمان كفاءة الطاقة	هل قام المرشح بتوصيل لوحة ESP8266 بالكمبيوتر المحمول باستخدام كابل بيانات USB مناسب، واختار مُفسّر MicroPython الصحيح في ثوني؟	
	هل جمّع المرشح دوائر الأجهزة لمخرجات GPIO ، وأجهزة PWM (مثل: محرك سيرفو)، وأجهزة الإدخال (مثل: زر الضغط) بشكل صحيح؟	
	هل كتب المرشح منطق تحكم بلغة بايثون يتفاعل مع أجهزة الإدخال (مثل: ضغطات الأزرار) ويتحكم في المُشغّلات وفقًا لذلك؟	
	هل اختبر المرشح سلوك الأجهزة وراقبه لضمان استجابة المُشغّلات كما هو متوقع؟	
	هل حسّن المرشح منطق التحكم بإضافة تأخيرات، ومعايرة المُشغّلات، وضبط العتبات، وإعادة نشر الكود؟	
	هل استخدم المرشح عبارات الطباعة للتصحيح المباشر وتوثيق التغييرات باستخدام التعليقات؟	
	هل حافظ المرشح على إصدارات متعددة من النصوص البرمجية لتتبع تقدم التطوير والتحسينات؟	

4.2.2 التقييم الختامي أو التجميعي

اسم المتدرب:	**التاريخ:**	
اسم المقيّم:	**التاريخ:**	
يعتبر المتدرب مؤهلاً إذا أثبت أداء أفضل الممارسات في جميع المعايير التالية للتقييم المتعلقة بوحدة **الكفاية 3- تطوير البرامج للتحكّم في الأجهزة**		

معيار التقييم	المؤهّل	نعم	لا (*)
كتابة برامج تحكم بلغة بايثون للمشغلات والأجهزة الطرفية	يجب أن يكون المرشح قادرًا، في غضون 15 دقيقة أو أقل، على إنشاء وحفظ نص برمجي بلغة MicroPython في بيئة التطوير المتكاملة Thonny IDE، واستيراد المكتبات المطلوبة، وتهيئة دبابيس GPIO للمشغّلات أو الأجهزة الطرفية، وكتابة وتنفيذ منطق التحكّم لتشغيل الأجهزة وإيقافها ضمن حلقة، والتحقق من صحة التشغيل من خلال مخرجات REPL دون أخطاء.	☐	☐
اختبار منطق التحكم	يجب أن يكون المرشح قادرًا، في غضون 15 دقيقة أو أقل، على إنشاء وحفظ نص برمجي بلغة MicroPython في بيئة التطوير المتكاملة Thonny IDE، وكتابة حلقة while مع شرط تبديل متغير حالة جهاز تمت محاكاته، وعرض حالة الجهاز في REPL باستخدام عبارات ()print، وتنفيذ النص البرمجي، والتحقق من تغير حالة الجهاز بشكل صحيح وفقًا لمنطق الشرط دون أخطاء.	☐	☐
تكوين دبابيس GPIO للتحكم في الأجهزة الرقمية باستخدام MicroPython واستكشاف أخطاء GPIO الشائعة وإصلاحها	يجب أن يكون المرشح قادرًا، في غضون 10 دقائق أو أقل، على تهيئة دبوس GPIO واحد على الأقل كمخرج، وإظهار التحكم الرقمي عبر تبديل الحالة (HIGH/LOW)، واستكشاف أخطاء GPIO وإصلاحها وإجراء التصحيحات اللازمة.	☐	☐
إنشاء وتعديل إشارات PWM للتحكم في السلوك التناظري	يجب أن يكون المرشح قادرًا، في غضون 10 دقائق أو أقل، على إعداد مخرج PWM على طرف GPIO، وضبط دورة العمل أو التردد لإنشاء استجابة قابلة للملاحظة من الأجهزة، واستكشاف أخطاء PWM وإصلاحها، وإجراء التصحيحات اللازمة.	☐	☐
تحسين منطق التحكم باستخدام اتصالات الأجهزة	يجب أن يكون المرشح قادرًا، في غضون 30 دقيقة أو أقل، على توصيل مستشعر ومشغّل بـ ESP8266، وتحميل برنامج تحكم، وتشغيل الكود في كل من المحاكاة والأجهزة، والتحقق من السلوك، وتحليل النتائج، وتحسين منطق التحكم لضمان استجابة النظام بشكل صحيح.	☐	☐
المعيار (*)	الدليل على عدم الامتثال إن وجد (**)		
توقيع المقيّم:	التاريخ:		
توقيع المتدرب:	التاريخ:		

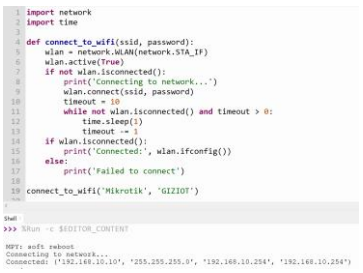
5 وحدة الكفاية 4: إنشاء اتصال بالشبكة باستخدام بايثون للتكامل السحابي

5.1 معيار التدريب - الوحدة 4

5.1.1 المبادئ التوجيهية للتدريب

ناتج القدرات/التعلّم:	الأدوات المطلوبة:	المواد والمستهلكات المطلوبة:
• تهيئة إعدادات الشبكة لأجهزة إنترنت الأشياء • كتابة نصوص برمجية بلغة بايثون لتوصيل الأجهزة بالمنصات السحابية • نقل بيانات المستشعرات بأمان إلى السحابة • استكشاف أخطاء اتصال الشبكة الأساسية ونقل البيانات وإصلاحها **موقع التدريب:** غرفة داخلية بمساحة تقريبية 4x5 متر، مجهزة بـ: • نافذة واسعة • طفاية حريق • مفاتيح ومقابس كهربائية متعددة للكومبيوترات • أثاث خشبي ثابت مناسب لأنشطة مختبر الإلكترونيات • مكاتب ومناضد لوضع أجهزة الكمبيوتر، والأدوات، والمواد اللازمة للتطبيق العملي • خزانة وأدراج لتنظيم وتخزين المكونات الإلكترونية ومواد المختبر منطقة تربة (أو حاويات تربة) داخل الغرفة لاختبار أجهزة استشعار رطوبة التربة	• كمبيوتر شخصي وشاشة • اتصال بالإنترنت • مقياس متعدد • بطارية محمولة • محول تيار مستمر 12 فولت • محول تيار مستمر 5 فولت بمنفذ USB صغير • صناديق تخزين للمكونات • مجموعة أدوات أساسية (مفكات براغي، كماشة، قاطع أسلاك) • حصيرة مضادة للكهرباء الساكنة، آمنة ضد التفريغ الكهروستاتيكي	• لوحة تطوير ESP8266 (مثل NodeMCU و Wemos D1 Mini) • كابل بيانات Micro USB (تأكد من أنه كابل بيانات، وليس مجرد كابل شحن) • وحدة استشعار DHT22/DHT11 • وحدة استشعار مستوى الماء • وحدة استشعار رطوبة التربة • وحدة استشعار الإضاءة المحيطة • بطارية ليثيوم أيون قابلة لإعادة الشحن 18650 مع حامل • مصابيح RGB LED • مصابيح LED • مقاومات • جرس كهربائي • وحدة أزرار الضغط • مفاتيح • وحدة استشعار المسافة بالموجات فوق الصوتية • لوحة تجارب • أسلاك توصيل جامبر (أنثى-ذكر، ذكر-ذكر) • وحدة إمداد الطاقة للوحة التجارب 3.3 فولت، 5 فولت

تهيئة إعدادات الشبكة لأجهزة إنترنت الأشياء



تكوين إعدادات الواي فاي باستخدام مايكروبايثون

- تأكد من أن الجهاز ضمن نطاق جهاز توجيه الواي فاي
- تأكد من أن شبكتك تدعم شبكة انترنت بتردد 2.4 جيغاهرتز (لا تدعم ESP8266/ESP32 تردد 5 جيغاهرتز)
- اكتب نصًا أساسيًا لاتصال الواي فاي

```
import network
import time

def connect_to_wifi(ssid, password):
    wlan = network.WLAN(network.STA_IF)
    wlan.active(True)
    if not wlan.isconnected():
        print('Connecting to network...')
        wlan.connect(ssid, password)
        timeout = 10
        while not wlan.isconnected() and timeout > 0:
            time.sleep(1)
            timeout -= 1
    if wlan.isconnected():
        print('Connected:', wlan.ifconfig())
    else:
        print('Failed to connect')
connect_to_wifi('Mikrotik', 'GIZIOT')
```

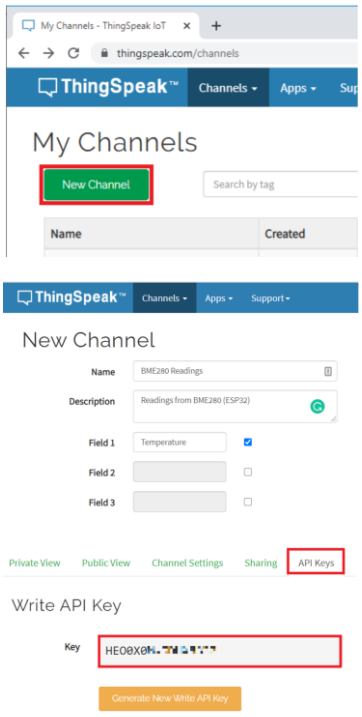
```
if not wlan.isconnected():
print('Connecting to network...')
wlan.connect(ssid, password)
timeout = 10
while not wlan.isconnected() and timeout > 0:
time.sleep(1)
timeout -= 1
if wlan.isconnected():
print('Connected:', wlan.ifconfig())
else:
print('Failed to connect')

connect_to_wifi('Your_SSID', 'Your_PASSWORD')
```

- احفظ الملف باسم boot.py أو wifi_connect.py
- حمّله إلى ESP باستخدام إدارة ملفات ثوني
- أعد تشغيل اللوحة وراقب المخرجات على غلاف ثوني

كتابة نصوص برمجية بلغة بايثون لتوصيل الأجهزة بالمنصات السحابية

إعداد حساب سحابي (ThingSpeak)

- أدخل الرابط: https://thingspeak.mathworks.com في عنوان المتصفح الخاص بك
- انقر على زر "ابدأ مجانًا" Get Started For Free لإنشاء حساب ThingSpeak
- افتح القنوات Channels← اختر قنواتي My Channels← أنشئ قناة جديدة Create a new channel
- اكتب اسمًا لقناتك وأضف وصفًا
  - إذا كنت ترغب في نشر قراءات متعددة (مثل الرطوبة والضغط)، يمكنك تفعيل عدة مجالات
- انقر على زر "حفظ القناة" لإنشاء قناتك وحفظها



واجهة برمجة التطبيقات (API)

لإرسال قيم من ESP8266 أو ESP32 إلى ThingSpeak، ستحتاج إلى مفتاح كتابة واجهة برمجة التطبيقات (Write API Key). افتح علامة تبويب "مفاتيح واجهة برمجة التطبيقات" وانسخ مفتاح كتابة واجهة برمجة التطبيقات (Write API Key) ومعرّف القناة إلى مكان آمن لأنك ستحتاج إليهما لاحقًا.

ملاحظة هامة: لا تشارك مفتاح واجهة برمجة التطبيقات (API Key) علنًا لأنه سيتيح الوصول إلى بياناتك

اكتب نصًا برمجيًا بلغة مايكروبايثون: HTTP Post إلى ThingSpeak

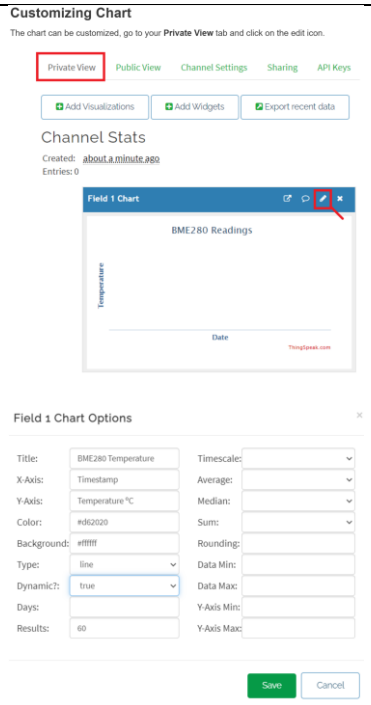
- اكتب الكود التالي:

```
import network
import urequests
import time

def connect_wifi(ssid, password):
wlan = network.WLAN(network.STA_IF)
```

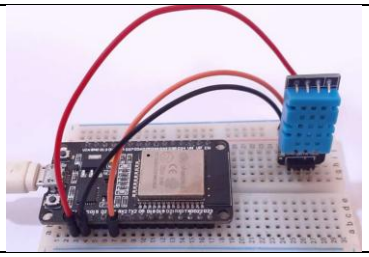
```
wlan.active(True)
if not wlan.isconnected:()
wlan.connect(ssid, password)
while not wlan.isconnected:()
(1)time.sleep
print("Connected:", wlan.ifconfig())

def send_to_thingspeak(api_key, value):
url =
"https://api.thingspeak.com/update?api_key={}&field1={}".for
mat(api_key, value)
response = urequests.get(url)
print("Server response:", response.text)
response.close()

# مثال الاستخدام
connect_wifi('YourSSID', 'YourPassword')
:while True
# استبدال بقراءة المستشعر الفعلية sensor_value = 42
send_to_thingspeak('YOUR_API_KEY', sensor_value)
time.sleep()
```

تخصيص المخطط

- انتقل إلى علامة تبويب "العرض الخاص" Private View
- انقر على زر التعديل Edit
  - أضف عنوانًا للمخطط
  - خصّص لون الخلفية، ومحوري x وy، وغيرها
  - عندما تنتهي، اضغط على زر "حفظ" Save



نقل بيانات المستشعرات بأمان إلى السحابة



أوصل DHT11 أو DHT22 بموصل ESP8266 أو ESP32

- أدخل حساس DHT11 في لوحة التجارب
- أوصل دبوس VCC الخاص بالحساس بدبوس 3.3 فولت في وحدة ESP.
  أوصل الأرضي (GND) بالأرضي
- أوصل دبوس بيانات الحساس بمنفذ GPIO2 في وحدة ESP

تحقق من قراءات المستشعر باستخدام نص برمجي بسيط بلغة بايثون على ثوني

- اكتب النص التالي:

```
import dht
import machine
import time

sensor = dht.DHT11(machine.Pin(2))

while True:
sensor.measure()
temp = sensor.temperature()
hum = sensor.humidity()
print('Temp:', temp, 'C Humidity:', hum, '%')
time.sleep()
```

إعداد اتصال واي فاي آمن

- استخدم وحدة الشبكة في مايكرو بايثون

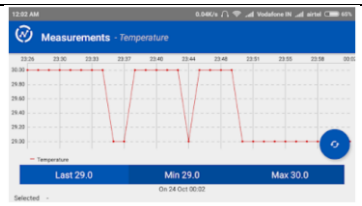
```
import network

sta_if = network.WLAN(network.STA_IF)
sta_if.active(True)
sta_if.connect('YOUR_SSID', 'YOUR_PASSWORD')

while not sta_if.isconnected():
pass

print('Connected:', sta_if.ifconfig())
```

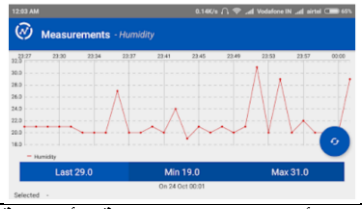
اكتب نصًا برمجيًا أساسيًا للاتصال بشبكة الواي فاي، وقراءة مستشعر DHT، وإرسال البيانات إلى ThingSpeak

- اكتب النص البرمجي التالي:

```
import network
import urequests
import utime
import dht
from machine import Pin

# بيانات اعتماد الواي فاي
ssid = 'YOUR_SSID'
password = 'YOUR_PASSWORD'

# الاتصال بشبكة الواي فاي
```

	sta = network.WLAN(network.STA_IF) sta.active(True) sta.connect(ssid, password) while not sta.isconnected(): print("...جارٍ الاتصال بشبكة Wi-Fi") utime.sleep() print("متصل، عنوان IP:", sta.ifconfig()[0]) # إعداد المستشعر sensor = dht.DHT11(Pin(2)) # منفذ GPIO2 هو D4 # ThingSpeak API api_key = 'YOUR_API_KEY' url = 'https://api.thingspeak.com/update' # حلقة While True: sensor.measure() temp = sensor.temperature() hum = sensor.humidity() print(f'Temp: {temp}°C | Humidity: {hum}%') # إرسال البيانات full_url = f"{url}?api_key={api_key}&field1={temp}&field2={hum}" try: response = urequests.get(full_url) print("Sent to ThingSpeak:", response.text) response.close() except: print("Failed to send data.") utime.sleep(15) • احفظ الكود باسم main.py • حمّله إلى خادم ESP باستخدام ثوني • قم بتشغيله وراقب إخراج REPL للتحقق من السجلات والأخطاء





التحقق عبر السحابة

- سجّل الدخول إلى ThingSpeak.com
- افتح لوحة معلومات القناة الخاصة بك
- اطلع على مخططات درجة الحرارة والرطوبة في الوقت الفعلي، مع تحديثها كل 15 ثانية

استكشاف أخطاء اتصال الشبكة الأساسية ونقل البيانات وإصلاحها

قراءة حالة الجهاز

- الوصول إلى مخرجات شاشة التسلسل
- مراجعة رسائل حالة الاتصال:
  - "جارٍ الاتصال بشبكة Wi-Fi..."
  - "تم الاتصال بنجاح" + عنوان IP
  - "فشل الاتصال" أو خطأ في المصادقة
- شرح كيفية التحقق من معلومات IP والشبكة: استخدم sta_if.ifconfig() في MicroPython لاسترداد وتفسير تكوين IP
- اختبر هذا البرنامج النصي الذي يحاول الاتصال بشبكة الواي فاي:

```
import network
import time

# استبدل ببيانات اعتماد Wi-Fi الخاصة بك
SSID = "Your_SSID"
PASSWORD = "Your_PASSWORD"

def connect_to_wifi():
wlan = network.WLAN(network.STA_IF)
wlan.active(True)

if not wlan.isconnected():
print("Connecting to Wi-Fi...")
wlan.connect(SSID, PASSWORD)

# Wait until connected or timeout after 10 seconds
timeout = 10
while not wlan.isconnected() and timeout > 0:
print("Waiting for connection...")
time.sleep(1)
timeout -= 1

if wlan.isconnected():
print("Connected successfully!")
print("Network config:", wlan.ifconfig())
else:
print("Failed to connect.")
```

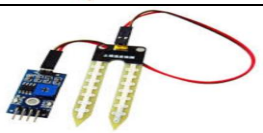
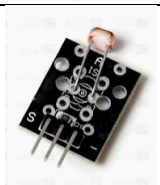
	connect_to_wifi()
	معالجة الأخطاء في نصوص بايثون • تنفيذ تسجيل الأخطاء بإضافة عبارات طباعة أو تسجيل أخطاء للتشخيص • عرض توضيحي لعينة الكود: عرض نص برمجي بسيط لمعالجة أخطاء طلبات HTTP *import urequests* *try:* *response = urequests.get('https://api.thingspeak.com/update?api_key=YOUR_API_KEY&field1=25')* *if response.status_code == 200:* *print("Data sent successful")* *else:* *print("Failed with status code:", response.status_code)* *response.close()* *except Exception as e:* *print("Error happened:", e)*
	تصحيح أخطاء إرسال البيانات • تحقق من استجابات HTTP: وعند إرسال البيانات، راجع رمز الحالة المُعاد. على سبيل المثال: 200 - تم إرسال البيانات بنجاح 403 - الوصول ممنوع؛ تحقق من مفتاح API أو الأذونات 404 - لم يتم العثور على المورد؛ تحقق من عنوان URL أو نقطة النهاية 500 - خطأ في الخادم؛ حاول مرة أخرى لاحقًا أو تحقق من سجلات الخادم • تحقق من البيانات على منصات السحابة من خلال تسجيل الدخول إلى لوحة معلومات المنصة وتأكد ما إذا تم استلام البيانات، أو تخزينها، أو رفضها. • إدارة حدود سرعة واجهة برمجة التطبيقات: في حال إرسال طلبات متعددة في وقت قصير، توقف مؤقتًا بين الطلبات لتجنب تجاوز الحد المسموح به. • اختبر اتصال الشبكة عبر إرسال أمر لشبكة ping إلى شبكة الواي فاي أو أرسل طلبات اختبار بسيطة لتأكيد الوصول إلى الإنترنت قبل إرسال البيانات.

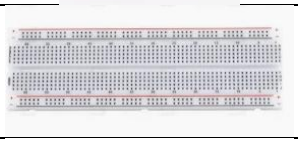
5.1.2 الادوات والمواد

الادوات	الصورة	الكمية [4]	الملاحظة
جهاز كمبيوتر شخصي مع شاشة ولوحة مفاتيح وفأرة (مثبت عليه نظام تشغيل ويندوز OS) أو كمبيوتر محمول مزود بمنفذ USB مثبت عليه (Python 3.8+، وThonny IDE، وبرامج تشغيل USB: برامج تشغيل CP210x / CH340) صلاحيات إدارية لتثبيت التعريفات/البرمجيات، مع بيانات اعتماد شبكة WiFi)		1 لكل محطة	
اتصال بالإنترنت		1 لكل فصل دراسي	
مقياس متعدد		1 لكل محطة	
بطارية محمولة		1 لكل محطة	
محول تيار مستمر ٥ فولت بمنفذ USB صغير		1 لكل محطة	
محول تيار مستمر 12 فولت		1 لكل محطة	
صناديق تخزين للمكونات		1 لكل محطة	

[4] إن الكمية المشار إليها للورشة تعني إمكانية استخدام المنتج لوحدات مختلفة في الورشة ذاتها، مما يعني أنه لا يُمكن تسليم وحدتان تستخدمان المنتج في نفس الوقت.
تحذير! قبل شراء المنتج، يُرجى التحقق من وجوده في الورشة لتجنب أي تكرار غير ضروري. كما يجب مراجعة القائمة والتحقق منها من قِبل خبير مُختص.

مجموعة أدوات أساسية (مفكات، كماشة، قواطع أسلاك)		1 لكل محطة	
حصيرة مضادة للكهرباء الساكنة آمنة ضد التفريغ الكهروستاتيكي		1 لكل محطة	

المواد والمستهلكات	الصورة	الكمية (لكل محطة)	الملاحظة
لوحة تطوير ESP8266 (مثل NodeMCU وWemos D1 Mini)		1	
كابل بيانات Micro USB		1	(تأكد من أنه كابل بيانات، وليس كابل شحن فقط)
وحدة استشعار DHT22/DHT11		1	
وحدة استشعار مستوى الماء		1	
وحدة استشعار رطوبة التربة		1	
وحدة استشعار الإضاءة		1	
RGB LED		1	
مصابيح LED		5	

مقاومات 470 أوم، 1 كيلو أوم و 10 كيلو أوم		10 لكل منها	
جرس الكتروني		1	
وحدة أزرار الضغط		2	
مفتاح التشغيل/الإيقاف		2	
لوحة تجارب		1	
وحدة إمداد طاقة لوحة التجارب 3.3 فولت، 5 فولت		1	
أسلاك توصيل جامبر (أنثى-ذكر، ذكر-ذكر)		رزمة لكل منها	
بطارية ليثيوم أيون قابلة لإعادة الشحن 18650		1	
حامل بطارية ليثيوم أيون قابلة لإعادة الشحن 18650 مع سلك		1	

5.2 معايير التقييم- الوحدة 4

5.2.1 التقييم التكويني

اسم المتدرب:	التاريخ:	
القدرات/ناتج التعلّم	**المعايير لوحدة الكفاية 4 - إنشاء اتصال بالشبكة باستخدام بايثون للتكامل السحابي**	**نعم/لا**
تهيئة إعدادات الشبكة لأجهزة إنترنت الأشياء	هل تأكد المرشح من أن جهاز إنترنت الأشياء ضمن نطاق شبكة الواي فاي وأن الشبكة تدعم تردد 2.4 جيغاهرتز؟	
	هل كتب المرشح نصًا برمجيًا أساسيًا بلغة MicroPython لتوصيل الجهاز بشبكة الواي فاي؟	
	هل استخدم المرشح بشكل صحيح network.WLAN (network.STA_IF) لتفعيل واجهة الواي فاي ؟	
	هل أنشأ المرشح حلقة وصل تنتظر اتصالاً ناجحاً بالواي فاي مع انتهاء المهلة؟	
	هل عرض المرشح حالة الاتصال وتكوين IP في REPL ؟	
	هل حفظ المرشح نص الواي فاي بشكل صحيح (مثل boot.py أو wifi_connect.py) ورفعه إلى الجهاز؟	
	هل أعاد المرشح تشغيل اللوحة وراقب مخرجات واجهة Thonny shell لتأكيد الاتصال؟	
كتابة نصوص برمجية بلغة بايثون لتوصيل الأجهزة بالمنصات السحابية	هل أنشأ المرشح حسابًا سحابيًا مثل (ThingSpeak) وهيأ قناةً تحتوي على حقول لبيانات المستشعر؟	
	هل استرد المرشح مفتاح برمجة التطبيقات (Write API Key) ومعرّف القناة وحفظهما بأمان؟	
	هل كتب المرشح نصًا برمجيًا بلغة MicroPython للاتصال بالسحابة عبر طلبات HTTP (urequests)؟	
	هل استورد المرشح المكتبات اللازمة (الشبكة، urequests، الوقت) وأنشأ اتصالًا بالواي فاي في النص البرمجي؟	
	هل نسّق المرشح عنوان URL لطلب HTTP بشكل صحيح لإرسال البيانات إلى قناة السحابة؟	
	هل أنشأ المرشح حلقةً لإرسال بيانات المستشعر أو بيانات الاختبار إلى السحابة بشكل دوري؟	
	هل خصّص المرشح عرض مخطط منصة السحابة وتحقق من صحة تعيين حقول البيانات؟	
نقل بيانات المستشعرات بأمان إلى السحابة	هل قام المرشح بتوصيل مستشعر DHT11/DHT22 بشكل صحيح بـ ESP8266/ESP32 (VCC، GND، دبوس البيانات لتصحيح GPIO)؟	
	هل تحقق المرشح من قراءات المستشعر باستخدام نص برمجي بلغة بايثون في ثوني؟	
	هل أنشأ المرشح اتصال آمن بالواي فاي باستخدام وحدة الشبكة في MicroPython ؟	
	هل كتب المرشح نصًا برمجيًا يقرأ بيانات المستشعر وأرسلها إلى السحابة باستخدام مفتاح API المناسب؟	
	هل عالج المرشح أخطاءً في معالجة طلبات HTTP الفاشلة عند إرسال البيانات؟	
	هل راقب المرشح مخرجات REPL للحصول على سجلات نقل ناجحة وقراءات المستشعر؟	
	هل تحقق المرشح من عرض البيانات بشكل صحيح على منصة السحابة في الوقت الفعلي؟	
استكشاف أخطاء اتصال الشبكة الأساسية ونقل البيانات وإصلاحها	هل قرأ المرشح وفسّر مخرجات شاشة التسلسل لتحديد رسائل حالة الاتصال؟	
	هل استخدم المرشح وظيفة sta_if.ifconfig() للتحقق من عنوان IP وتكوين الشبكة؟	
	هل نفذ المرشح واختبر نصوص اتصال بالواي فاي مع معالجة سليمة لحالات فشل الاتصال وانقطاعات الوقت؟	
	هل سجل المرشح الأخطاء أو استخدم عبارات الطباعة لتشخيص حالات فشل طلبات HTTP ؟	
	هل تحقق المرشح من رموز استجابة HTTP (مثل: 200، 403، 404، 500) وفسّرها بشكل صحيح؟	
	هل تحقق المرشح من استقبال البيانات على منصة السحابة وأكّد ما إذا كانت محفوظة، أو مرفوضة، أو متأخرة؟	
	هل تولى المرشح إدارة حدود معدل واجهة برمجة التطبيقات واختبار الاتصال بالشبكة لضمان نقل البيانات بشكل موثوق؟	

5.2.2 التقييم الختامي أو التجميعي

اسم المتدرب:	**التاريخ:**	
اسم المقيّم:	**التاريخ:**	
يعتبر المتدرب مؤهلاً إذا أثبت أداء أفضل الممارسات في جميع المعايير التالية للتقييم المتعلقة بوحدة **الكفاية 4- إنشاء اتصال بالشبكة باستخدام بايثون للتكامل السحابي**		

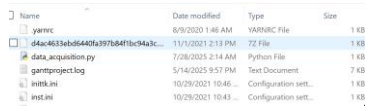
معيار التقييم	المؤهّل	نعم	لا (*)
تهيئة إعدادات الشبكة لأجهزة إنترنت الأشياء	يجب أن يكون المرشح قادرًا، في غضون 20 دقيقة أو أقل، على تهيئة إعدادات الشبكة لجهاز إنترنت الأشياء عبر تحديث مُعرّف مجموعة الخدمات (SSID) وكلمة المرور في البرنامج النصي أو ملف التكوين، ونشر التغييرات على الجهاز، والتحقق من اتصاله بنجاح بشبكة الواي فاي المحددة باستخدام عنوان IP صالح.	☐	☐
دمج إنترنت الإشياء مع السحابة	يجب أن يكون المرشح قادرًا، في غضون 60 دقيقة أو أقل، على توصيل جهاز إنترنت الأشياء بمنصة سحابية بنجاح عبر كتابة برنامج نصي بلغة بايثون لتهيئة إعدادات الواي فاي.	☐	☐
تكوين الأجهزة والشبكات	يجب أن يكون المرشح قادرًا، في غضون 30 دقيقة أو أقل، على توصيل أجهزة الاستشعار بجهاز التحكم الدقيق بشكل صحيح وإنشاء اتصال مستقر بالواي فاي.		
نقل البيانات بشكل آمن والتكامل السحابي	يجب أن يكون المرشح قادرًا، في غضون 30 دقيقة أو أقل، على كتابة برنامج نصي بلغة بايثون ونشره وتشغيله، لقراءة بيانات الاستشعار ونقلها بأمان إلى منصة السحابة، ومعالجة الأخطاء، والتحقق من استلام البيانات.	☐	☐
استكشاف أخطاء اتصال الشبكة الأساسية ونقل البيانات وإصلاحها	يجب أن يكون المرشح قادرًا، في غضون 30 دقيقة أو أقل، على تحديد وحل مشكلات الاتصال الأساسية بالشبكة ونقل البيانات من خلال تصحيح أخطاء نصوص بايثون، والتحقق من إعدادات الواي فاي ومنصة السحابة، وضمان تحميل بيانات المستشعر بنجاح وتأكيدها على لوحة معلومات السحابة.	☐	☐
المعيار (*)	الدليل على عدم الامتثال إن وجد (**)		
توقيع المقيّم:	التاريخ:		
توقيع المتدرب:	التاريخ:		

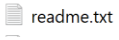
6 وحدة الكفاية 5 : تنفيذ أنشطة نهاية العمل

6.1 معيار التدريب- الوحدة 5

6.1.1 المبادىء التوجيهية للتدريب

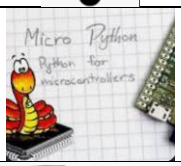
<table>
<tr>
<td>ناتج القدرات/التعلّم:
• إجراء التنظيف والنسخ الاحتياطي
• توثيق رمز المصدر
• تخزين أدوات الأجهزة
• إعداد التقارير وفقًا لسجلات المهام

موقع التدريب:
غرفة داخلية بمساحة تقريبية 4x5 متر، مجهزة بـ:
• نافذة واسعة
• طفاية حريق
• مفاتيح ومقابس كهربائية متعددة للكومبيوترات
• أثاث خشبي ثابت مناسب لأنشطة مختبر الإلكترونيات
• مكاتب ومناضد لوضع أجهزة الكمبيوتر، والأدوات، والمواد اللازمة للتطبيق العملي
• خزانة وأدراج لتنظيم وتخزين المكونات الإلكترونية ومواد المختبر</td>
<td>الأدوات المطلوبة:
• جهاز كمبيوتر شخصي مع شاشة أو كمبيوتر محمول مزود بمنفذ USB
• نظام تشغيل ويندوز مُثبّت
• بايثون 3.8+
• بيئة التطوير المتكاملة: Thonny/VS) (Code/uPyCraft
• برامج تشغيل USB: برامج تشغيل CP210x / CH340
• بيانات اعتماد شبكة الواي فاي
• اتصال بالإنترنت
• صلاحيات إدارية لتثبيت برامج التشغيل/البرامج
• حصيرة مضادة للكهرباء الساكنة
• خزانة أدراج لتخزين المكونات
• مجموعة أدوات أساسية (مفكات براغي، كماشة، قاطع أسلاك)</td>
<td>المواد والمستهلكات المطلوبة:
• لوحة تطوير ESP8266 (مثل NodeMCU وWemos D1 Mini)
• كابل بيانات Micro USB (تأكد من أنه كابل بيانات، وليس مجرد كابل شحن)
• وحدة استشعار DHT22/DHT11
• وحدة استشعار رطوبة التربة
• وحدة استشعار الإضاءة المحيطة
• بطارية ليثيوم أيون قابلة لإعادة الشحن 18650 مع حامل
• مصابيح LED
• مقاومات
• لوحة تجارب
• أسلاك توصيل جامبر (أنثى-ذكر، ذكر-ذكر)
• وحدة إمداد الطاقة للوحة التجارب 3.3 فولت، 5 فولت</td>
</tr>
<tr><td colspan="3">تهيئة إعدادات الشبكة لأجهزة إنترنت الأشياء</td></tr>
<tr><td colspan="3">إجراء التنظيف والنسخ الاحتياطي</td></tr>
<tr>
<td>  </td>
<td colspan="2">تنظيف جهاز الكمبيوتر المُخصص للتطوير
• افتح المجلد الذي تُحفظ فيه البرامج النصية
• تحقق من وجود ملفات .pyc، أو .log، أو .temp، أو .old، أو .zip
• احذف أو أرشف الملفات غير الضرورية</td>
</tr>
<tr>
<td></td>
<td colspan="2">أنشئ مجلد نسخ احتياطي محلي
• أنشئ مجلدًا باسم IoT_Backup <التاريخ>
• انسخ ملفات .py والتكوينات الأساسية إلى داخله
• أضف ملف README.txt لوصف المحتوى والإصدار</td>
</tr>
<tr>
<td>  </td>
<td colspan="2">سجّل عملية التنظيف والنسخ الاحتياطي على بطاقة الذاكرة
• املأ تقرير التنظيف/النسخ الاحتياطي بالمعلومات التالية:
○ التاريخ والوقت
○ الملفات المحذوفة
○ الملفات التي تم نسخها احتياطيًا
○ موقع النسخ الاحتياطي
○ التعليقات على أي مشاكل</td>
</tr>
<tr><td colspan="3">توثيق رمز المصدر</td></tr>
</table>

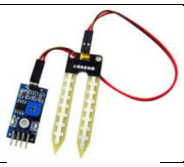
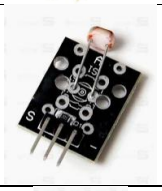
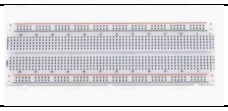
	**شرح التعليقات المضمنة وتعليقات الحظر وبنية تعليقات بايثون:** • استخدم سلاسل توثيق بايثون المكونة من ثلاث علامات اقتباس: def connect_wifi:() """ للاتصال بشبكة واي فاي محددة مسبقًا باستخدام وحدة ESP8266 القيمة المُعادة: bool: صحيح إذا نجح الاتصال، وإلا خطأ. """ pass
	**إنشاء وحفظ ملف README أساسي** • إنشاء وحفظ ملف README: ○ اسم المشروع ○ الوصف والغرض ○ تعليمات الإعداد ○ كيفية التشغيل ○ المؤلف ومعلومات الاتصال
	نظّم ملفات التوثيق • احفظ فيها جميع أدلة المشروع، والأدلة، وملفات README • استخدم التعليقات وسلاسل التوثيق في نصوص بايثون لشرح وظائف الكود • أضف أرقام الإصدارات أو التواريخ في أسماء ملفات النصوص والمستندات لتتبع التحديثات
تخزين أدوات الأجهزة	
	**التنظيف قبل التخزين** • امسح الأدوات بمنظف مناسب • أزل أي بقايا (مثل اللحام، الغبار، الزيوت) • تحقق من وجود أي تلف مرئي • أبلغ عن الأدوات المعيبة أو صنفها لإصلاحها أو التخلص منها • صنّف الأدوات إلى فئات محددة مسبقًا • استخدم فواصل أو حشوات إسفنجية للحماية • تأكد من تخزين أسلاك التوصيل، وأجهزة الاستشعار، والمكونات في أكياس أو حجرات مضادة للكهرباء الساكنة • أعد الأدوات إلى صناديق التخزين المخصصة لها والتي تحمل علامة • تأكد من أن الملصقات واضحة ومطابقتها لنظام الجرد (إن وجد) • ضع الأجهزة الحساسة للكهرباء الساكنة في حاويات مضادة للكهرباء الساكنة • افصل جميع البطاريات أو مصادر الطاقة عن المكونات النشطة
إعداد التقارير وفقًا لسجلات المهام	
	**توثيق أنشطة المهام** • إعداد التقارير لضمان إمكانية التتبع، والمساءلة، والصيانة، واستمرارية المشروع. • تسجيل أنواع مختلفة من معلومات المهام: سجلات الصيانة، وتقارير التكوين، وسجلات الأخطاء، ونتائج الاختبارات. • تضمين جميع المكونات الرئيسية في تقرير المهمة: التاريخ/الوقت، وأعضاء الفريق المشاركين، ووصف المهمة/أهدافها، والأساليب/الإجراءات، والأدوات/البرامج/المعدات المستخدمة، والمشكلات والحلول، والمخرجات/النتائج، وإجراءات المتابعة.

	• مراجعة وتحليل سجلات المهام (الحقيقية أو المحاكاة) لاستخراج البيانات الرئيسية، مع ضمان الوضوح، والاكتمال، والدقة الفنية. • إكمال نماذج التقارير (اليدوية والرقمية) بمعلومات موجزة، وموضوعية، وصحيحة فنيًا باستخدام المصطلحات المناسبة.

6.1.2 الادوات والمواد

الادوات	الصورة	الكمية [5]	الملاحظة
جهاز كمبيوتر شخصي مع شاشة ولوحة مفاتيح وفأرة (مثبت عليه نظام تشغيل ويندوز OS) أو كمبيوتر محمول مزود بمنفذ USB مثبت عليه (Python 3.8+، وThonny IDE، وبرامج تشغيل USB: برامج تشغيل CP210x / CH340) صلاحيات إدارية لتثبيت التعريفات/البرمجيات، مع بيانات اعتماد شبكة WiFi)		1 لكل محطة	
اتصال بالإنترنت		1 لكل محطة	
كتالوجات ومواصفات أجهزة إنترنت الأشياء ورقة مرجعية لقواعد مايكروبايثون		1 لكل محطة	
صناديق تخزين للمكونات		1 لكل محطة	
مجموعة أدوات أساسية (مفكات، كماشة، قواطع أسلاك)		1 لكل محطة	
حصيرة مضادة للكهرباء الساكنة آمنة		1 لكل محطة	
لوحة تطوير ESP8266 (مثل NodeMCU وWemos D1 Mini)		1	
كابل بيانات Micro USB		1	(تأكد من أنه كابل بيانات، وليس كابل شحن فقط)

[5] إن الكمية المشار إليها للورشة تعني إمكانية استخدام المنتج لوحدات مختلفة في الورشة ذاتها، مما يعني أنه لا يُمكن تسليم وحدتان تستخدمان المنتج في نفس الوقت.
تحذير! قبل شراء المنتج، يُرجى التحقق من وجوده في الورشة لتجنب أي تكرار غير ضروري. كما يجب مراجعة القائمة والتحقق منها من قِبل خبير مُختص.

وحدة استشعار DHT22/DHT11		1	
وحدة استشعار رطوبة التربة		1	
وحدة استشعار الإضاءة		1	
مصابيح LED		5	
مقاومات 470 أوم، 1 كيلو أوم و 10 كيلو أوم		10 لكل منها	
لوحة تجارب		1	
وحدة إمداد طاقة لوحة التجارب 3.3 فولت، 5 فولت		1	
أسلاك توصيل جامبر (أنثى-ذكر، ذكر-ذكر)		رزمة لكل منها	
بطارية ليثيوم أيون قابلة لإعادة الشحن 18650		1	
حامل بطارية ليثيوم أيون قابلة لإعادة الشحن 18650 مع سلك		1	

6.2 معايير التقييم- الوحدة 5

6.2.1 التقييم التكويني

اسم المتدرب:	التاريخ:	
القدرات/ناتج التعلّم	المعايير لوحدة الكفاية 5 - تنفيذ أنشطة نهاية العمل	نعم/لا
إجراء التنظيف والنسخ الاحتياطي	هل حدد المتدرب الملفات غير الضرورية (.pyc، .log، .temp، .old، .zip) في مجلد جهاز التطوير؟	
	هل قام المتدرب بحذف أو أرشفة الملفات غير الضرورية بشكل صحيح؟	
	هل أنشأ المتدرب مجلد نسخ احتياطي محلي باستخدام التسمية الصحيحة (IoT_Backup_<date>)؟	
	هل نسخ المتدرب جميع ملفات py. الأساسية وملفات التكوين إلى مجلد النسخ الاحتياطي؟	
	هل أنشأ المتدرب ملف README.txt يصف محتوى وإصدار النسخة الاحتياطية؟	
	هل أكمل المتدرب تقرير التنظيف/النسخ الاحتياطي، بما في ذلك التاريخ/الوقت، والملفات التي تم حذفها، والملفات التي تم نسخها احتياطيًا، وموقع النسخ الاحتياطي، والتعليقات على المشاكل؟	
توثيق رمز المصدر	هل استخدم المتدرّب التعليقات المضمنة (inline) وتعليقات الكتل (block comments) في نصوص بايثون البرمجية بشكل مناسب؟	
	هل طبّق المتدرب سلاسل توثيق بايثون للوظائف لوصف الغرض، والمدخلات، والمخرجات، والسلوك؟	
	هل أنشأ وحفظ المتدرب ملف أساسي README ، يتضمن اسم المشروع، والوصف/الغرض، وتعليمات الإعداد، وتعليمات التنفيذ، ومعلومات الاتصال بالمؤلف؟	
	هل نظم المتدرب جميع ملفات التوثيق في المجلد المخصص لها مع أسماء الملفات، وأرقام الإصدارات، والتواريخ الصحيحة؟	
تخزين أدوات الأجهزة	هل نظف المتدرب الأدوات قبل التخزين، وأزال البقايا وتحقق من وجود أي تلف واضح؟	
	هل بلّغ المتدرب عن الأدوات المعيبة، أو صنفها لإصلاحها أو التخلص منها؟	
	هل صنّف المتدرب الأدوات إلى فئات مُحددة مسبقًا واستخدم فواصل واقية أو حشوات إسفنجية؟	
	هل خزّن المتدرب أسلاك التوصيل، وأجهزة الاستشعار، والمكونات في أكياس أو حجرات مضادة للكهرباء الساكنة؟	
	هل أعاد المتدرب الأدوات إلى صناديق تخزين مُصنفة بملصقات واضحة تتوافق مع نظام الجرد؟	
	هل تأكد المتدرب من تخزين الأجهزة الحساسة للتفريغ الكهروستاتيكي في حاويات مضادة للكهرباء الساكنة؟	
	هل فصل المتدرب جميع البطاريات أو مصادر الطاقة عن المكونات النشطة قبل التخزين؟	
إعداد التقارير وفقًا لسجلات المهام	هل احتفظ المتدرب بتقارير لضمان إمكانية التتبع، والمساءلة، والصيانة، واستمرارية المشروع؟	
	هل سجّل المتدرب أنواعًا مختلفة من معلومات المهام، بما في ذلك سجلات الصيانة، وتقارير التكوين، وسجلات الأخطاء، ونتائج الاختبارات؟	
	هل أدرج المتدرب جميع المكونات الرئيسية في تقرير المهمة: التاريخ/الوقت، وأعضاء الفريق، وأهداف المهمة، والأساليب/الإجراءات، والأدوات/البرمجيات/المعدات، والمشكلات/الحلول، والمخرجات/النتائج، وإجراءات المتابعة؟	
	هل راجع المتدرب وحلّل سجلات المهام لاستخراج البيانات الرئيسية بوضوح واكتمال ودقة فنية؟	
	هل أكمل المتدرب نماذج التقارير (اليدوية والرقمية) بمعلومات موجزة، وموضوعية، وصحيحة فنيًا باستخدام المصطلحات المناسبة؟	
	هل أدرج المتدرب جميع المكونات الرئيسية في تقرير المهمة: التاريخ/الوقت، وأعضاء الفريق، وأهداف المهمة، والأساليب/الإجراءات، والأدوات/البرمجيات/المعدات، والمشكلات/الحلول، والمخرجات/النتائج، وإجراءات المتابعة؟	

6.2.2 التقييم الختامي أو التجميعي

اسم المتدرب:	**التاريخ:**	
اسم المقيّم:	**التاريخ:**	
يعتبر المتدرب مؤهلاً إذا أثبت أداء أفضل الممارسات في جميع المعايير التالية للتقييم المتعلقة بوحدة **الكفاية 5- تنفيذ أنشطة نهاية العمل**		

معيار التقييم	المؤهّل	نعم	لا (*)
إجراء التنظيف والنسخ الاحتياطي	يجب أن يكون المرشح قادرًا، في غضون 30 دقيقة أو أقل، على تحديد الملفات غير الضرورية وإزالتها، وتنظيم مكونات المشروع الأساسية، وإنشاء مجلد نسخ احتياطي بتسمية مناسبة يتضمن جميع البرامج النصية المهمة وملفات التكوين والوثائق. يجب حفظ النسخة الاحتياطية بنجاح في مكان آمن والتحقق من اكتمالها.	☐	☐
توثيق رمز المصدر	يجب أن يكون المرشح قادرًا، في غضون 30 دقيقة، على تطبيق التعليقات المضمنة، وكتابة سلاسل توثيق متسقة على مستوى الوظيفة، وإنشاء ملف README موجز يشرح غرض البرنامج النصي، واستخدامه، وتبعياته، والمدخلات/المخرجات المتوقعة، مع ضمان سهولة فهم الكود وصيانته من قبل مطور بايثون آخر.	☐	☐
تخزين أدوات الأجهزة	يجب أن يكون المرشح قادرًا، في غضون 15 دقيقة، على تنظيف جميع أدوات الأجهزة، وتنظيمها، وتخزينها، مع ضمان تحديد الأدوات التالفة أو المعيبة والإبلاغ عنها، ووضع كل أداة في مكانها المخصص، وحماية العناصر الحساسة للتفريغ الكهروستاتيكي، وترتيب منطقة التخزين واكتمال محتوياتها، وتوثيق أي تشوهات.	☐	☐
إعداد التقارير وفقًا لسجلات المهام	يجب أن يكون المرشح قادرًا، في غضون 30 دقيقة أو أقل، على تجميع وتقديم تقرير فني بناءً على مهمة مكتملة، باستخدام التنسيق المناسب والتأكد من استخراج جميع المعلومات المهمة بدقة من سجل المهمة.	☐	☐
المعيار (*)	الدليل على عدم الامتثال إن وجد (**)		
توقيع المقيّم:	التاريخ:		
توقيع المتدرب:	التاريخ:		

INTERNAL

german cooperation
DEUTSCHE ZUSAMMENARBEIT

Co-funded by the European Union

Implemented by
giz Deutsche Gesellschaft für Internationale Zusammenarbeit (GIZ) GmbH

Improving the Quality and Attractiveness of Vocational Education and Training in Lebanon for poor and vulnerable social groups (QuA-VET)

Curriculum for IoT Developer

Module: Developing Python applications for IoT devices

DATE: 23/07/2025

Introduction

The Multi-Donor Action "Improving the Quality and Attractiveness of Technical and Vocational Education and Training (TVET) in Lebanon for poor and vulnerable social groups" is jointly co-financed by the European Union and the Federal Ministry of Economic Cooperation and Development (BMZ). The joint action is implemented by GIZ as a specific Action within the wider BMZ project "Improving the Quality and Attractiveness of TVET in Lebanon (QuA-VET)." The QuA-VET project has developed this Curriculum as part of a series of curricula in the Agrofood and Internet of Things (IoT) in its overall effort to promote the use of standards across public and private vocational and technical training providers – schools, institutes, NGOs, and other training institutions in Lebanon.

A major objective of developing and adopting standards in the training standards is to set the best practice defining a curriculum is to promote consistency and quality, including training taking place outside the framework of the General Directorate for Technical and Vocational Education (DGVTE). Other objectives in adopting a modular approach are to introduce greater flexibility (both external and internal) into the VTE system; make VTE more attractive and raise its status; increase participation rates and/or reduce early dropout; and promote links with the labour market to meet industry needs.

The process to design and deliver training modules can be seen as the process to design and deliver a curriculum. The model for curriculum design divides the process into four stages connected to one another, forming together what we might call the outcome - oriented (or competency-based) curriculum:

- Occupational standard: a document which sets out a norm of what people need to learn and how the quality and content of learning is assessed;
- Qualification standard: a document which sets out a norm of what people need to learn;
- Assessment standard: a document which sets out a norm of how the quality and content of learning is assessed, shortly to verify if the learner is competent to carry out the occupation;
- Education standard: a document which translates the qualification standard into the educational context;
- Training or learning standard: a document which plans teaching and learning activities, considering the environments and resources required, duration and learners' needs. It provides detail on each of the tasks and sub-tasks that make up the qualification.

This curriculum may be used for a variety of purposes, including:

- guiding the development of trainees' abilities
- accreditation of training programs
- assessing performance and career development
- supporting training providers who are developing or revising basic and advanced training programs
- developing criteria for prior learning assessment and recognition
- providing guidance to employers for recruiting, selecting, training and retaining employees

This curriculum sets the standards towards best practice in the occupation and is developed around a limited number of Competence Units (3 to 5) grouping together the abilities required to perform the tasks in line with the internationally recognized best practice of the sector. Each competence unit can be assessed independently from the others and can be acquired in different settings (formal, informal and non-formal). The assessment of competences will be carried out by a professional entity independent from the training provider assuring reliability, transparency and consistency in the assessment and in the certification of the qualification.

Regardless of the size of the project, the tasks must be performed in a manner that complies with Occupational Health and Safety (OH&S) guidelines and regulations.

Curriculum delivery

The curriculum is a set of standards that should be interpreted and delivered to trainees having in mind that at the completion of the training program they will be tested against the criteria set in the assessment standards that are reflecting best practices. Education and Training standards are the two documents that a trainer will use in the delivery of the curriculum. The education standards set the overall timeframe required for the learning of the competences set in each competence unit while the Training standards set the best practice adopted in carry out the tasks by a competent and qualified operator. The Training standards should be considered as a guideline for the trainer to assure that trainees will carry on the tasks as stated in the Training standards. The trainer will monitor the progress of the trainees in acquiring the abilities filling in the formative assessment for each trainee.

Necessary materials and facilities

For the appropriate delivery of the training module, it is necessary to provide for the necessary facilities (practical training workshops), materials and tools included that are described in the Training standards. Tools and materials must be the ones that are currently used in the trade. They should not be models or equipment oriented to make understand but tools and materials for practicing the tasks of the trade.

It is essential for learning the abilities of the trade that practical exercises resemble real working conditions as closely as possible. Hence, the tools and materials as well as the guidelines given regarding the practical training venue are a prere developing, testing and deploying prerequisite for demonstrating the working techniques by the trainer, followed by exercises executed by the students themselves. This also means that the number of tools and the material as stated in the Training standards should be respected to ensure real practical training. In addition, audio-visual material is required to help trainees visualize procedures and processes that can't be demonstrated or practiced.

Teaching and learning approaches

The instructor should choose a participative teaching-learning approach, providing as much possibility to the trainee to discover and practice as possible. The trainee should be encouraged to actively participate in the development of his/her skills through discussion as well as practical exercises in the real world of the workshop.

The desirable didactical approach of training is characterized as follows:

- Instruction is based on action orientation and project based work in groups
- Learning experiences are guided by frequent feedback using the formative assessment sheet
- Instruction is, to a considerable extent, field− centered − based on realistic work problem and situations;

Furthermore, the training standard serves as the basis for the development of a training plan, including:

- the timetable, with the allocation of the appropriate duration of training time for each content and task;
- the assignment of each content and task to appropriate training venue (school/training center classroom, workshop or company).

Assessment

Based on the Assessment standards individuals will then be assessed in order to determine if they have the requisite abilities for effective work performance. The assessment of trainees by a professional entity independent from the training provider will assure also feedback to the training provider on the "quality" of the training provided.

Table of Contents

1 IoT Developer: Developing Python applications for IoT devices

1.1 Occupational standard

ISCO-08 code: 2512 Software developers	**Occupational standard** IoT developer	
The occupation covers the process of developing, testing and Deploying Python-based Applications on IoT devices, with a focus on data acquisition, device monitoring, and control.		
Process	**Element**	**Description**
P1 Developing Python applications for IoT devices	Output	IoT application code, functional IoT device with deployed application, configuration documentation
	Sub-Process	Setting up the development environment Writing Scripts for Data Acquisition from IoT Sensors Developing programs for Device Control Establishing network connectivity using Python for cloud integration Performing end-of-work activities
	Input	New product or service, technology upgrades, industry requirements, customer demand, market trends, project requirements
	Resources	Sensors, actuators, embedded systems, power sources, network, IoT devices, Python software and libraries, cloud services

1.2 Competence standard

ISCO-08 code: 2512 Software developers		**Competence standard** IoT Developer
A competent person should be able to develop and test Python applications on IoT devices, for data acquisition and device control.		
Competence unit	**Ability / Competence** **Learning outcome**	**Knowledge**
CU1 Setting up the development environment	Set Up an IDE and Python Environment for MicroPython Configure development tools and drivers Establish microcontrollers communication Import external libraries and dependencies	MicroPython Device drivers External libraries Serial communication protocols USB/COM port handling
CU2 Writing IoT data acquisition scripts	Write Python scripts to collect data from IoT sensors using MicroPython Interface sensors with code to read and display their data Debug data acquisition errors Extract sensor data	Python scripts to collect data from various sensors Digital or analogue sensors Data acquisition methods Debugging tools Common error patterns
CU3 Developing programs to control devices	Write control programs in Python for actuators and peripheral devices Test control logic to actuate devices based on conditions Use GPIOs and PWM outputs Refine control logic using hardware connections while ensuring energy efficiency	Python Actuators GPIO, PWM, and conditional logic Libraries
CU4 Establishing network connectivity using Python for cloud integration	Configure network settings for IoT devices Write Python scripts to connect devices to cloud platforms Transmit sensor data securely to the cloud Troubleshoot basic network connectivity and data transmission issues	Network protocols used in IoT (Wi-Fi, MQTT, HTTP) Basics of Python for IoT communication Structure and function of IoT cloud platforms Data security principles for cloud transmission
CU5 Performing end-of-work activities	Conduct cleanup and backup Document source code Store hardware tools Report according to task records	Documentation techniques Version control systems and workflows using tools like Git Reports and logs

1.3 Assessment standard

ISCO-08 code: 2512 Software developers	**Assessment standard** IoT Developer
A competent person should be able to develop and test Python applications on IoT devices, for data acquisition and device control.	
Competence unit 1	Setting up the development environment
Assessment criteria and indicators	The candidate is considered competent if he/she can demonstrate a best practice performance in all the 4 learning outcomes / abilities within the allocated assessment time: YES/NO IDE installed and set up YES/NO Python installed YES/NO Development tools and drivers configured YES/NO Microcontroller communication established YES/NO External libraries and dependencies imported
Assessment procedure	The assessor observes the candidate installing and setting up the Integrated Development Environment (IDE) and installing Python and any required libraries. The candidate configures the development tools and installs the necessary drivers based on the setup instructions. Microcontroller communication is established by connecting and verifying interaction between devices. The candidate also imports external libraries and dependencies as listed in the development requirements Quality assurance is ensured through the use of a standardized checklist, and moderation of results after completion. Assessment time will be predefined and sufficient for all tasks, based on prior trial runs
Competence unit 2	Writing IoT data acquisition scripts
Assessment criteria and indicators	The candidate is considered competent if he/she can demonstrate a best practice performance in all the 4 learning outcomes / abilities: YES/NO Python scripts to collect data from IoT sensors written using MicroPython YES/NO Sensors interfaced with code YES/NO Data acquisition errors debugged YES/NO Sensor data extracted
Assessment procedure	The assessor observes the candidate writing Python scripts to collect data from IoT sensors using the provided requirements. The candidate interfaces the sensors with the code so that the sensors respond and transmit data. During the process, the candidate identifies and debugs any errors related to data acquisition. The candidate then extracts the sensor data for further use. Consistency of assessment across assessors and institutions is maintained through the use of a standardized marking checklist, clear performance criteria, and post-assessment moderation or calibration sessions to align scoring decisions.
Competence unit 3	Developing programs to control devices
Assessment criteria and indicators	The candidate is considered competent if he/she can demonstrate a best practice performance in all the 4 learning outcomes / abilities: YES/NO Control programs used for actuators and peripherals devices YES/NO Control logic tested to actuate devices based on conditions

	YES/NO GPIOs and PWM outputs used YES/NO Control logic refined based on test results and performance feedback
Assessment procedure	The assessor observes the candidate using control programs to operate actuators and peripheral devices based on task requirements. The candidate tests the control logic to actuate devices in response to specified conditions. General-purpose input/output (GPIO) pins and pulse-width modulation (PWM) outputs are used to manage device behavior. The candidate then refines the control logic based on the outcomes of the test procedures.
Competence unit 4	Establishing network connectivity using Python for cloud integration
Assessment criteria and indicators	The candidate is considered competent if he/she can demonstrate a best practice performance in all the 4 learning outcomes / abilities: YES/NO Network settings for IoT devices configured YES/NO Python scripts to connect devices to cloud platforms written YES/NO Sensor data securely transmitted to the cloud YES/NO Basic connectivity issues troubleshot
Assessment procedure	The assessor observes the candidate configuring network settings for IoT devices according to task requirements. The candidate writes Python scripts to connect IoT devices to cloud platforms as specified. Sensor data is securely transmitted to the cloud. The candidate then troubleshoots basic network connectivity and data transmission issues to ensure stable device-to-cloud communication.
Competence unit 5	Performing end-of-work activities
Assessment criteria and indicators	The candidate is considered competent if he/she can demonstrate a best practice performance in all the 4 learning outcomes / abilities: YES/NO Cleanup and backup conducted YES/NO Source code documented YES/NO Hardware tools stored YES/NO Task records reported
Assessment procedure	The assessor observes the candidate conducting cleanup and creating a backup of the project files as required. The candidate documents the source code by adding necessary comments and information to support understanding and future use. Hardware tools used during the task are collected and stored in their designated locations. The candidate completes the process by reporting task records using the provided format.

1.4 Education standard

ISCO-08 code: 2512 Software developers	**Education standard** IoT Developer
Occupational standard	Developing Python applications for IoT devices (ISCO-08 code: 2512)
Assessment standard	Developing Python applications for IoT devices (ISCO-08 code: 2512)
Overview	At the completion of the Training Module the participant will be able to develop, test and Deploy Python-based Applications on IoT devices, with a focus on data acquisition, device monitoring, and control as well as establish network connectivity using Python for cloud integration
Competences	Set Up an IDE and Python Environment for MicroPython Configure development tools and drivers Establish microcontrollers communication Import external libraries and dependencies Write Python scripts to collect data from IoT sensors Interface sensors with code to read and display their data Debug data acquisition errors Extract sensor data Write control programs in Python for actuators and peripheral devices Test control logic to actuate devices based on conditions Use GPIOs and PWM outputs Refine control logic using hardware connections Configure network settings for IoT devices Write Python scripts to connect devices to cloud platforms Transmit sensor data securely to the cloud Troubleshoot basic network connectivity and data transmission issues Conduct cleanup and backup Document source code Store hardware tools Report according to task records
Knowledge	MicroPython Thonny Device drivers External libraries Serial communication protocols USB/COM port handling Python scripts to collect data from various sensors Digital or analogue sensors Data acquisition methods Debugging tools Common error patterns Actuators GPIO, PWM, and conditional logic Network protocols used in IoT (Wi-Fi, MQTT, HTTP) Basics of Python for IoT communication Structure and function of IoT cloud platforms Data security principles for cloud transmission Documentation techniques Version control systems and workflows using tools like Git Reports and logs

<table>
<tr><td>Learning program</td><td colspan="5">During the classroom standard into lessons the following topics will be covered:
1. Introduction to IoT and Development Environments
2. Python for IoT Data Acquisition
3. Microcontroller and Sensor Communication
4. Programming Device Control with Python
5. Cloud Integration and Secure Data Transmission
6. End-of-Work Procedures and Documentation

During the workshop lessons the following practical activities will be executed:
1. Installing and configuring python, IDEs and drivers
2. Testing device connectivity
3. Developing and deploying Python scripts for real-time sensor data collection
4. Debugging scripts and validating sensor data extraction
5. Connecting circuits
6. Writing and testing control programs to operate actuators and peripherals
7. Refining control logic using hardware connections
8. Configuring network settings for device connectivity
9. Writing Python scripts to connect devices to cloud platforms
10. Testing and troubleshooting secure sensor data transmission

During on the job training the following job activities will be carried out
1. Configuring network settings for device connectivity (5 hours)
2. Testing device connectivity within actual workflows (3 hours)
3. Developing and deploying Python scripts for real-time sensor data collection (16 hours)
4. Debugging scripts in a live operational environment (6 hours)
5. Writing and testing control programs to operate workplace-specific actuators and peripherals (18 hours)
6. Refining control logic using hardware connections (2 hours)
7. Writing Python scripts to connect devices to cloud platforms (15 hours)
8. Testing and troubleshooting secure data transmission (5 hours)
9. Conducting cleanup and backup (3 hours)
10. Documenting source code and maintenance logs (4 hours)
11. Storing tools and completing reports (3 hours)
On-the-job training placements are coordinated with approved host organizations, following an agreed training plan. Progress is monitored through regular supervisor check-ins, activity logs signed by both trainee and supervisor, and periodic reviews by the training provider to ensure learning outcomes are met</td></tr>
<tr><td rowspan="4">Learning time frame</td><td>Competence code</td><td>Classroom lesson (hrs)</td><td>Workshop lessons (hrs)</td><td>On the job training (hrs)</td><td>Total (hrs)</td></tr>
<tr><td>CU 1</td><td>1</td><td>2</td><td>3</td><td>6</td></tr>
<tr><td>CU 2</td><td>2</td><td>8</td><td>22</td><td>32</td></tr>
<tr><td>CU 3</td><td>1</td><td>8</td><td>20</td><td>29</td></tr>
</table>

	CU 4	2	12	25	39
	CU 5	1	3	10	14
	Total (hrs)	7	33	80	120

2 Competence Unit 1: Setting up the development environment

2.1 Training Standard – Unit 1

2.1.1 Training guidelines

<table>
<tr>
<td>Ability / Learning outcome:

• Set Up an IDE and Python Environment for MicroPython
• Configure development tools and drivers
• Establish microcontrollers communication
• Import external libraries and dependencies

Training site:

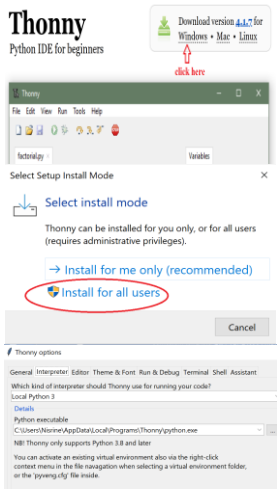
Indoor room of approximately 4 x 5 m equipped with

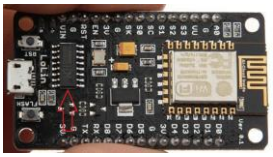
• Wide window
• Fire extinguisher
• Many electrical switches and sockets
• Fixed wooden high-chair furniture suitable for electronics lab activities
• Desks and benches where we can place PCs, tools, and materials for practical work
• Closet and drawers for organizing and storing electronic components and lab materials</td>
<td>Required Tools:

• PC & Monitor or Laptop with USB port
• Internet connection
• Anti-Static Mat ESD Safe</td>
<td>Required Materials and Consumables:

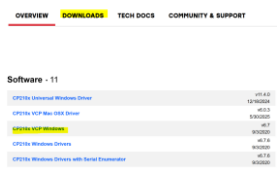
• ESP8266 Development Board (e.g., NodeMCU, Wemos D1 Mini)
• Micro USB data cable (ensure it is a data cable, not just charging)
• IoT device catalogues and specifications sheets
• MicroPython syntax reference sheet</td>
</tr>
<tr>
<td colspan="3">Set up an IDE and Python Environment for MicroPython</td>
</tr>
<tr>
<td><pre>C:\Users\Nisrine>python --version
Python 3.8.5</pre></td>
<td colspan="2">Download/ Install Python

• Download Python 3.8+ from Download Python | Python.org
• Follow the installer instructions and check "Add Python to PATH".
• Verify Python installation:
○ Open command prompt by pressing Win + R then type cmd and hit Enter.
○ Type: python --version.
○ If Python is installed, you will see something like: Python 3.8.5.
○ If it does not work, python may not be added to your system's path. You will need to repeat the Python installation process.</td>
</tr>
</table>

<table>
<tr><td> 
</td><td>Select and Install IDE

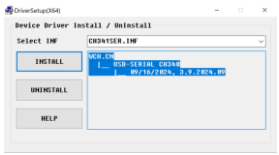
• Open your browser and go to https://thonny.org
• Download ‘Thonny’ IDE for windows
• Locate the downloaded file (thonny-4.x.x.exe) in your Downloads folder.
• Double click to Run the installer.
• Select ‘Install for all users’ and then accept the agreement.
• Follow the installer instructions with default settings.
• Check “Add Thonny to PATH” if prompted.
• Open the IDE to confirm it is working.
• Verify python installation: Go to tools > options > interpreter</td></tr>
<tr><td colspan="2">Configure development tools and drivers</td></tr>
<tr><td>  </td><td>Identify your USB-to-Serial chip

• Inspect your board physically, Look for markings on the small USB-to-Serial chip near the USB port:“CH340G” → needs CH340 driver
• “CP2102” → needs CP210x driver

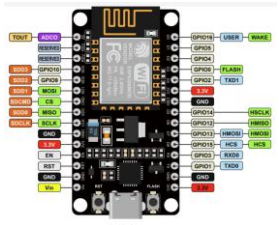
If unclear: Check your board’s data sheet.</td></tr>
<tr><td> 
</td><td>Download the driver

• Download CH340 driver from the website: https://www.wch.cn/downloads/CH341SER_ZIP.html
• Browse the silicon labs official website to download CP210x driver from: https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers
• Click on downloads tab and choose: <u>CP210x Windows Drivers</u>

Tip: It is better to download both drivers if you are unsure, it won’t cause conflicts.</td></tr>
</table>

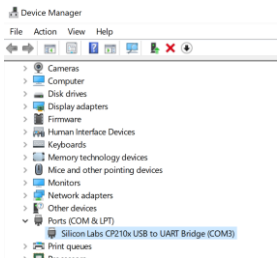
<table>
<tr><td>  </td><td>Install the driver

• Install CH340:
○ Unzip the downloaded .zip file.
○ Run SETUP.EXE (Windows).
○ Follow the on-screen instructions → Click Install.
○ You should see “Driver install successful.”

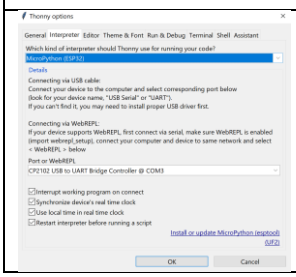
• Install CP210x:
○ Run the .exe installer.
○ Accept terms → Click Next.
○ Complete the installation.</td></tr>
<tr><td colspan="2">Establish microcontrollers communication</td></tr>
<tr><td>  </td><td>Examine the ESP8266 board to get familiar with its layout:

• Identify the main components:
ESP8266 chip (the small black square/rectangle).
USB connector (for programming and power).
Status LEDs (power and communication).

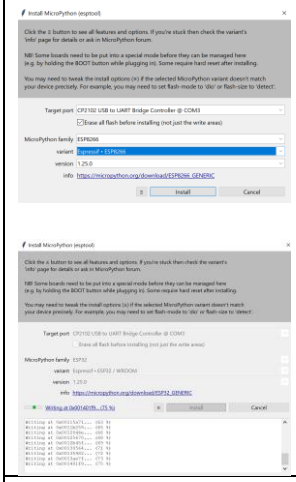
• Locate the pinout:
Power pins: 3.3V, GND.
GPIO pins: general-purpose input/output for sensors, actuators, etc.
Special pins: Reset (RST), Enable (EN), Analog input (ADC).</td></tr>
<tr><td>  </td><td>Connect the microcontroller

• Plug the ESP8266 into the PC with the Micro USB data cable.
• Ensure the ESP32 board is not placed on a metallic surface during operation to avoid short circuits by placing it on the Anti-Static Mat ESD Safe
• Confirm the board’s LED turns on (power indicator).
• Wait for Windows to detect the device.</td></tr>
<tr><td>  </td><td>Verify the COM Port

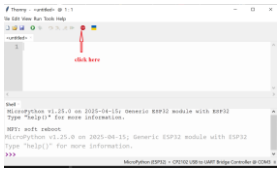
• Open Device Manager: Press Win + X → Device Manager.
• Expand “Ports (COM & LPT)”.Look for: USB-SERIAL CH340 (COMx) for CH340 or Silicon Labs CP210x USB to UART Bridge (COMx) for CP210x. you’ll need the com port in Thonny later.

If no COM port appears:
• Try a different USB port.
• Try a different Micro USB cable (must be a data cable).
• Reinstall the driver.
• Restart the PC if needed.Try another ESP8266 board.</td></tr>
<tr><td>  </td><td>Configure Thonny

• Open Thonny.
• Go to Tools > Options > Interpreter
• In Interpreter: Select MicroPython (ESP8266) or ESP32
• In Port: Select your identified COM port and click OK.</td></tr>
</table>

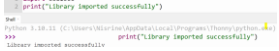
<table>
<tr><td>  </td><td>Flash MicroPython Firmware

• Open Thonny.
• Go to Tools > Options > Interpreter
• Click on ‘Install or update MicroPython (esptool)
• Select the target port.
• Select the MicroPython family: ESP8266
• Select Variant: Espressif ESP8266
• Select the latest version
• Click install
• Wait for flashing to complete and reset your ESP8266.

Note that if the flashing completes and the board resets without error, your firmware is installed.</td></tr>
<tr><td>  </td><td>Test Communication with microcontroller

• Click Stop/Restart Backend.

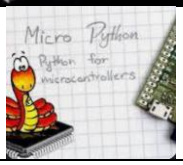
You should see: MicroPython v1.x.x on 202x-xx-xx; ESP module with ESP8266</td></tr>
<tr><td colspan="2">Import external libraries and dependencies</td></tr>
<tr><td>  </td><td>Prepare and upload the library

• Download the library from MicroPython GitHub drivers (example of library: ssd1306.py)
• Save it to your computer (e.g., Downloads folder).
• Open Thonny.
• Connect your ESP8266.
• Confirm REPL appears in the bottom shell. This is where you can directly type and run MicroPython commands on your ESP8266 once communication is successful.
• Go to File > Open... > This Computer.
• Open ssd1306.py.
• Go to File > Save As... > MicroPython device.
• Save as ssd1306.py.
• Verify upload: In Thonny’s file browser (left sidebar), confirm ssd1306.py appears under your ESP8266 files.</td></tr>
<tr><td>  </td><td>Test Import

• In Thonny’s REPL (bottom console area), type the following:
import ssd1306
print("Library imported successfully")
• Press Enter.
o If no errors appear and you see the message:
Library imported successfully
then the ssd1306 library has been successfully imported.
o If you get an error like ModuleNotFoundError, it means the library is not yet installed or uploaded to your device.</td></tr>
</table>

2.1.2 Tools and materials

Tools	Picture	Quantity[1]	Comment
PC & Monitor & keyboard & mouse (Windows OS installed) Or Laptop with USB port With installed (Python 3.8+, Thonny IDE, and USB Drivers: CP210x / CH340 Drivers and Administrative privileges to install drivers/software		1 per station	
Internet connection With WiFi network credentials	 	1 per classroom	
Anti-Static Mat ESD Safe		1 per station	

Materials and Consumables	Picture	Quantity	Comment
ESP8266 Development Board (e.g., NodeMCU, Wemos D1 Mini)		1 per station	
Micro USB data cable		1 per station	(ensure it is a data cable, not just charging)
IoT device catalogues and specifications sheets MicroPython syntax reference sheet		1 per station	

[1] The quantity expressed for workshop means that the item can be used for different modules in the same workshop, this imply that two modules that are using the item could not be delivered at the same time. Warning! before purchasing the item it should be verified if it already exists in the workshop to avoid unnecessary duplications. Also, the list should be reviewed and verified by an experienced professional.

2.2 Assessment standards – Unit 1

2.2.1 Formative assessment

Trainee Name:	**Date:**	
Ability / learning outcome	**Criteria for competence unit 1 – Setting up the development environment**	**Yes / No**
Set Up an IDE and Python Environment for MicroPython	Does the candidate download and install Python 3.8+ from the official Python.org website?	
	Does the candidate follow the installer instructions and check “Add Python to PATH”?	
	Does the candidate verify the Python installation by opening Command Prompt and checking the version with python --version?	
Configure development tools and drivers	Does the candidate identify the USB-to-Serial chip on the ESP8266 board (CH340G or CP2102)?	
	Does the candidate download the correct driver (CH340 or CP210x) from the official websites?	
	Does the candidate install the selected driver successfully?	
Establish microcontrollers communication	Does the candidate identify the ESP8266 components (chip, USB connector, LEDs)?	
	Does the candidate locate the power, GPIO, and special pins on the board?	
	Does the candidate connect the ESP8266 to the PC using a micro-USB data cable?	
	Does the candidate ensure the board is placed on a safe, non-conductive surface (e.g., ESD mat) to avoid short circuits?	
	Does the candidate verify that the board powers up (LED indicator turns on)?	
	Does the candidate open Device Manager, locate the correct COM port, and identify it as CH340 or CP210x?	
	If the COM port does not appear, does the candidate troubleshoot (try another port, cable, reinstall driver, or restart PC)?	
	Does the candidate open Thonny and select MicroPython (ESP8266/ESP32) as the interpreter?	
	Does the candidate select the identified COM port in Thonny and confirm the connection?	
	Does the candidate access Tools > Options > Interpreter in Thonny and choose “Install or update MicroPython (esptool)”?	
	Does the candidate select the correct port, family (ESP8266), and variant (Espressif ESP8266)?	
	Does the candidate install the latest MicroPython version and wait for the flashing process to complete successfully?	
	Does the candidate reset the board and confirm that firmware installation succeeded?	
	Does the candidate restart the backend in Thonny and confirm that the REPL displays the correct MicroPython version and ESP8266 module information?	
Import external libraries and dependencies	Does the candidate download an external library (e.g., ssd1306.py) from the MicroPython GitHub drivers?	
	Does the candidate open the library in Thonny and save it to the MicroPython device with the correct filename?	
	Does the candidate confirm that the library file appears in the ESP8266 file system within Thonny?	
	Does the candidate test the import of the uploaded library in Thonny’s REPL using import ssd1306?	
	Does the candidate verify the success of the import by printing “Library imported successfully”?	
	If an error occurs (e.g., ModuleNotFoundError), does the candidate troubleshoot the issue by re-uploading the library?	

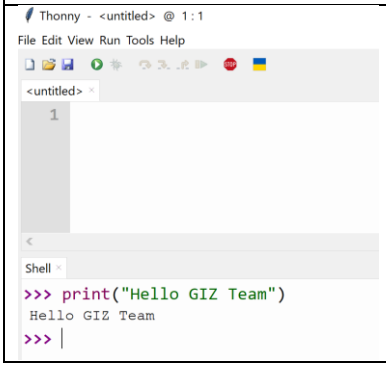
2.2.2 Summative assessment

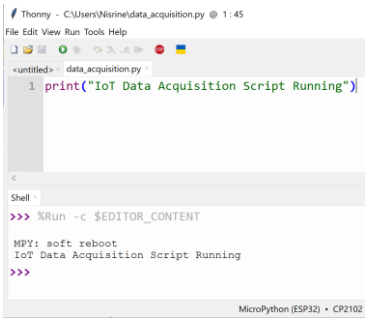
Trainee name:	**Date:**		
Assessor name:	**Date:**		
The candidate is considered competent if he demonstrates a best practice performance in all the 5 following criteria for the assessment related to the competence unit 1 - **Setting up the development environment**			
Criteria for assessment	**Qualifier**	**YES**	**NO (*)**
Install and Set up the required IDE	The candidate should complete the installation and initial setup of the required IDE (e.g., Thonny) in less than 15 minutes, ensuring the IDE opens without errors and recognizes the connected microcontroller.	☐	☐
Install Python	The candidate should complete the installation of Python in less than 15 minutes, ensuring that Python is added to the system PATH and can be verified by running python --version or python3 --version in the terminal without errors.	☐	☐
Install and configure drivers and development tools	The candidate should complete the installation and configuration of the required USB-to-Serial drivers and development tools in less than 15 minutes, ensuring the IoT device or microcontroller is recognized by the system and the development environment without errors.	☐	☐
Establish Microcontroller communication	The candidate should establish communication between the microcontroller and the development environment in less than 20 minutes, ensuring the device is recognized, the correct communication port is selected, and a test command is executed successfully in the REPL or console without errors.	☐	☐
Import external libraries	The candidate should complete the upload and import of required external libraries to the microcontroller in less than 15 minutes, ensuring that the library files are correctly placed on the device, the import command executes without errors in the REPL or script, and functionality is verified.	☐	☐
Criteria (*)	**Evidence of non-compliance if any (**)**		
Assessor signature:	**Date:**		
Trainee signature:	**Date:**		

3 Competence Unit 2: Writing IoT data acquisition scripts

3.1 Training standards – Unit 2

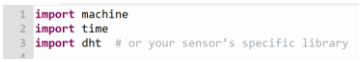
3.1.1 Training guidelines

Ability / Learning outcome: • Write Python scripts to collect data from IoT sensors using MicroPython • Interface sensors with code to read and display their data • Debug data acquisition errors • Extract sensor data **Training site:** Indoor room of approximately 4 x 5 m equipped with: • Wide window • Fire extinguisher • Many Electrical switches and sockets to accommodate several PCs • Fixed wooden high-chair furniture suitable for electronics lab activities • Desks and benches where we can place PCs, tools, and materials for practical work • Closet and drawers for organizing and storing electronic components and lab materials A soil area (or soil containers) inside the room for testing soil moisture sensors	**Required Tools:** • PC & Monitor or Laptop with USB port • Internet connection • Anti-Static Mat ESD Safe • Multimeter • Power Bank • 5V DC Adapter micro USB • Storage boxes for components • Basic tool kit (screwdrivers, pliers, wire cutters)	**Required Materials and Consumables:** • ESP8266 Development Board (e.g., NodeMCU, Wemos D1 Mini) • Micro USB data cable • DHT22/DHT11 sensor module • Soil Moisture sensor module • Ambient light sensor module • 18650 Li-Ion rechargeable battery + holder *1 • LEDs • Resistors • Push buttons module • switches • Breadboard • Jumper wires (female-male, male-male)
Write Python scripts to collect data from IoT sensors using MicroPython		
	**Open Thonny IDE:** • Open Thonny IDE by double-clicking the Thonny icon on your desktop or search for “Thonny” in your start menu to open the IDE. • Connect your ESP8266 or ESP32 to your computer using a USB cable. • Ensure Thonny detects the device (check the bottom-right corner for “MicroPython (ESP8266)” and the correct COM port). • Check that the REPL is working by typing: *print("Hello MicroPython")* and confirm the message appears without errors.	



Create a .py script

- Click on File > New, or press Ctrl + N. An empty script editor will open in Thonny, ready for your IoT code.
- Save the script:
  - Click File > Save As.
  - Choose where to save: On your computer for local editing and later upload or directly onto the microcontroller if you run the script immediately.
  - Name your file clearly, for example: data_acquisition.py
- Prepare for coding at the top of your script:
  - Import necessary libraries (machine, time, sensor libraries).
  - Initialize your sensor.
  - Write your data acquisition loop.
- Test saving and running:
  - Write a test line in your script:
  - print("IoT Data Acquisition Script Running")
  - Click Run (F5) to ensure your script executes, and the message appears in the Shell/REPL.



Use import machine, import time, and sensor-specific libraries

- Identify required libraries for IoT data acquisition:

Core libraries:
  - machine for GPIO and hardware control.
  - time for timing and delays.

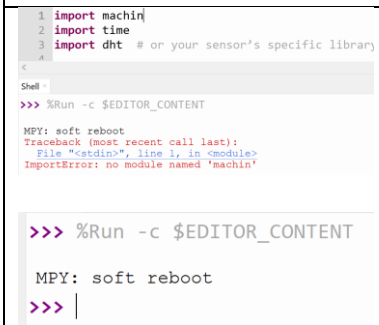
Sensor-specific libraries:
Example: dht for DHT11/DHT22 sensors.

- Open the IoT Script:
  - Open Thonny IDE.
  - Open the data_acquisition.py script or create a new .py file for testing imports.
- Write Import Statements at the Top of the Script:

import machine
import time
import dht # or your sensor's specific library

- Save the script.

Tip: Press Enter after typing each line so it runs correctly in Python.
The # symbol is used for comments, anything after # is ignored by the Python interpreter and is only there to help you understand the code.

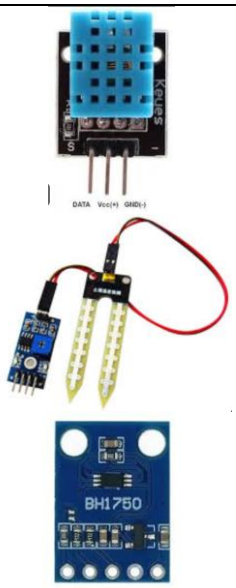


Test import statements

- Run the script using Run (F5). If no errors appear, the imports are successful.

If an error like ModuleNotFoundError appears:

- Check that the library is uploaded to the microcontroller in Thonny,
- Open the library file from your computer and use File > Save As... > MicroPython device, then confirm it appears in the device's file list.
- Confirm the correct spelling of the library name.



Review the 3 sensors and connection requirements

- Identify:

Sensor type (DHT11, soil moisture, light sensor).
GPIO pin used for the sensor signal.
Whether the pin should be set as INPUT (reading data) or OUTPUT (sending signals).

- Use the sensor's datasheet or lab wiring diagram to confirm pin numbers and voltage compatibility.

```
import dht  # or your sensor's specific library
sensor_pin = machine.Pin(5, machine.Pin.IN)  # GPIO5 as input
sensor = dht.DHT11(sensor_pin)  # Create a sensor object linked to GPIO5
print("GPIO pin and sensor initialized successfully")
```

```
Shell
>>> %Run -c $EDITOR_CONTENT

MPY: soft reboot
GPIO pin and sensor initialized successfully

>>>
```

Write initialization code for DHT 11 sensor

- Initialize the GPIO Pin of the DHT11 sensor in the code
  - Use the machine.Pin class to set up the pin.
    Example for an input pin: sensor_pin = machine.Pin(5, machine.Pin.IN) # GPIO5 as input
- Initialize the DHT 11 sensor using the Library
  - Use the initialization function from your sensor library.
    Example:
    import dht #press enter
    sensor = dht.DHT11(sensor_pin) # Create a sensor object linked to GPIO5
- Test Initialization
  - Add a simple print test to confirm no errors:
  - print("GPIO pin and sensor initialized successfully")
- Run the script in Thonny:
  - If the message displays without errors, initialization is successful.
  - If errors appear, check wiring, pin numbers, or library imports.

Note: When writing your full script, follow a clear structure:
  - Import libraries
  - Initialize pins and sensors
  - Loop to read, print, and store data
  - Handle errors

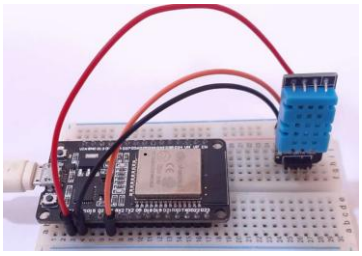
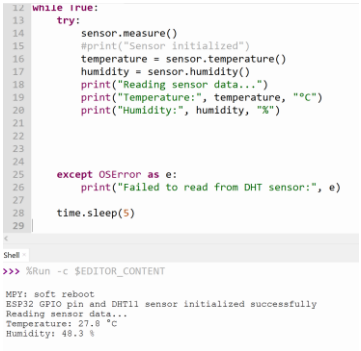
```
from machine import ADC
# Initialize ADC on pin A0
soil_sensor = ADC(0)

while True:
    moisture_value = soil_sensor.read()  # Returns 0-1023
    print("Soil moisture:", moisture_value)
```

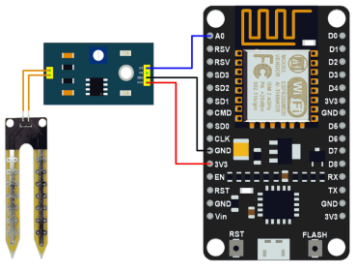
Write initialization code for the soil moisture

The soil moisture sensor usually comes with:
  - Probe (two metal pins that go into the soil).
  - Signal board (with power pins and output).

Pinout:
AO (Analog Out) → Connects to ADC pin of ESP8266 (A0).
DO (Digital Out) → Optional, can be connected to a GPIO pin for digital threshold.

- Initialize the Analog Digital Conversion Library
  from machine import ADC
- Map AO pin of sensor to A0 of ESP8266:

	soil_sensor = ADC(0) # ADC pin is fixed to A0 on ESP8266 • Read the moisture value moisture_value = soil_sensor.read() • Add a simple print test to confirm no errors: print("Soil moisture:", moisture_value) • Run the script in Thonny: ○ If the message displays without errors, initialization is successful. ○ If errors appear, check wiring, pin numbers, or library imports.
	**Write initialization code for the light sensor** The light intensity sensor outputs an analog signal, so we use the ADC pin (A0) of the ESP8266. *Initialize the ADC Pin of the Light Sensor in the code:* *from machine import ADC* *# Initialize ADC on pin A0* *light_sensor = ADC(0) # ADC(0) is the only ADC pin on ESP8266* • Test Initialization *value = light_sensor.read() # Returns value 0–1023* *print("Light sensor initialized, value:", value)* • Run the script in Thonny: If a value between **0 and 1023** is displayed, initialization is successful.
Interface sensors with code to read and display their data	
	**Connect the DHT 11 sensor to the ESP8266** • Identify: Sensor type (e.g., DHT11, soil moisture sensor, light sensor). Pin connections and GPIO pin numbers. Sensor voltage and communication requirements. • Connect the circuit consisted of the DHT11 module and the Esp8266 or ESP32 development module: ○ Connect the Vcc of the DHT11 module to the Vcc (3.3V) of the ESP8266 dev module using the red wire ○ Connect the ground (0V) of the DHT11 module to the ground of the ESP8266 dev module using the black wire Connect the data of the DHT11 module to the D1 (GPIO5) of the ESP8266 dev module using the orange wire.
	**Write a code to read the sensor (**<u>Example the temperature and humidity sensor DHT11 or DHT 22)</u> • Create an instance of your sensor linked to its GPIO pin *import dht* *sensor = dht.DHT11(machine.Pin(5)) #or DHT22* (The DHT stands for Digital Humidity and Temperature sensor) • Read data from the sensor by using the following method from the sensor's library: For DHT11: try: *sensor.measure()* *temperature = sensor.temperature()* *humidity = sensor.humidity()* • Display Sensor Data in the REPL by using the print() function to display the sensor's data clearly. Example:

	print("Temperature:", temperature, "°C") *print("Humidity:", humidity, "%")* • Embed in a Loop to read Continuously (infinite loop) • Wrap the reading and displaying code inside a while True loop*:* *while True:* *sensor.measure()* *temperature = sensor.temperature()* *humidity = sensor.humidity()* *print("Temperature:", temperature, "°C")* *print("Humidity:", humidity, "%")* *time.sleep(5)* Tip: We need to allow sensors time to stabilize between readings. That's why we Use: time.sleep(5) # Wait 5 seconds before the next reading to slow down the loop. This enables continuous monitoring in the REPL. • Save the script and click **Run (F5)**. • If errors occur, guide learners to: ○ Check wiring and initialization code. ○ Check the exact sensor library method names. • Save the script and run it in Thonny
	**Connect the soil moisture sensor to the ESP8266** • Identify: ○ Sensor type: Soil Moisture Sensor (analog output). ○ Pin connections: VCC, GND, AO (analog out). ○ Sensor voltage: 3.3V supply. ○ Communication: Analog signal → ESP8266 ADC pin (A0). • Connect the circuit consisting of the soil moisture sensor module and the ESP8266 development board: • Connect the VCC of the soil moisture module to the 3.3V (Vcc) of the ESP8266 dev module using a red wire. • Connect the GND of the soil moisture module to the GND of the ESP8266 dev module using a black wire. • Connect the AO (Analog Out) of the soil moisture module to the A0 (ADC pin) of the ESP8266 dev module using an orange wire.
`` import time from machine import ADC # Initialize the Soil Moisture Sensor (AO → A0) soil_sensor = ADC(0) print("Soil moisture sensor initialized successfully") while True: # Read the raw value (0 - 1023) value = soil_sensor.read() # Display raw value print("Soil moisture value:", value) time.sleep(5) # Delay 5 seconds before the next reading ```	**Write a code to read the soil moisture sensor** • Initialize the sensor object in Code, The soil moisture sensor provides an analog output, so we use the ESP8266's ADC pin (A0). *from machine import ADC* *soil_sensor = ADC(0) # Create a soil sensor object linked to A0* • Read Data from the Sensor, Use the read() method from the ADC class to collect data. *value = soil_sensor.read() # Returns 0–1023* • Display Sensor Data in the REPL, Use the print() function to display the sensor's data. *print("Soil moisture value:", value)* • Embed in a Loop for Continuous Reading (infinite loop) • Wrap the reading and displaying code inside a while True loop: *import time* *from machine import ADC* *# Initialize the Soil Moisture Sensor (AO → A0)*

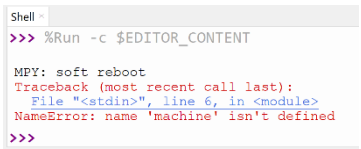
soil_sensor = ADC(0)
print("Soil moisture sensor initialized successfully")
while True:
Read the raw value (0 - 1023)
value = soil_sensor.read()
Display raw value
print("Soil moisture value:", value)
time.sleep(5) # Delay 5 seconds before the next reading

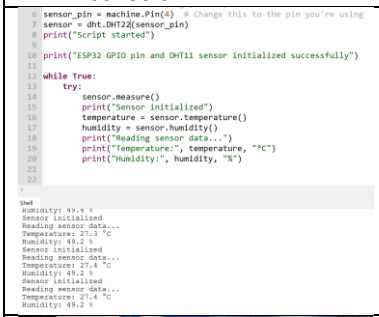
The delay gives the sensor time to stabilize and prevents spamming the REPL.

- Run and test the code
  - Save the script and click Run (F5) in Thonny.
  - If errors occur, guide learners to:
    - Check wiring (VCC → 3.3V, GND → GND, AO → A0).
    - Check that the correct ADC method (read()) is being used.
    - Make sure only A0 is used for analog input (ESP8266 has only one ADC).

Debug data acquisition errors

When working with microcontrollers and sensors, errors can arise from different sources. Some are hardware-related, such as loose or broken connections, using the wrong GPIO pins, incorrect pin wiring, or an insufficient power supply to the sensor. Others stem from the code itself, including syntax mistakes like missing colons or typos, incorrect GPIO pin numbers, or missing and wrongly imported libraries. Finally, there are runtime errors that occur during execution. These may include issues such as *ModuleNotFoundError*, *OSError* when attempting to read from a sensor, or simply no data being displayed in the REPL console.

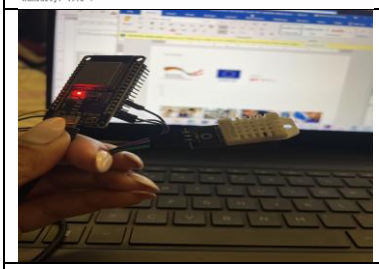




Use print statements for debugging

- Add print statements to identify where the script fails:

print("Script started")
print("Sensor initialized")
print("Reading sensor data...")



Verify the hardware connections

- Check if the Sensor wires are connected to correct GPIO pins.
- Check if the Power (VCC) and Ground (GND) are securely connected using the multimeter.
- Check if the microcontroller is properly connected and detected by the computer

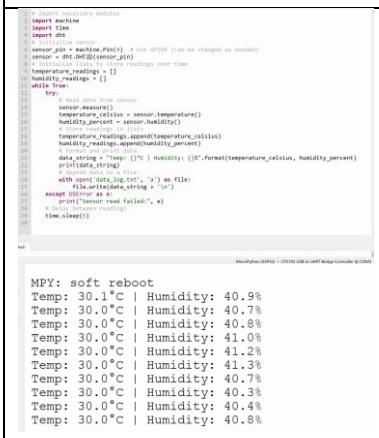
```
1 import machine
2 import time
3 import dht
```

Verify GPIO pin configurations in code & library imports

- Ensure the GPIO pin numbers in the code match the physical connections.
- Verify the correct mode (e.g., machine.Pin.IN for input).
- Check that necessary libraries are imported without typos:

import machine
import time
import dht

- Ensure external libraries (e.g., dht.py, ssd1306.py) are uploaded to the microcontroller if required.

	Handle sensor-specific runtime errors • Fix OSError or read failures by: ○ Adding appropriate delays using time.sleep(). ○ Wrapping sensor reads in try-except blocks: *sensor.measure()* *temp = sensor.temperature()* *print(temp)* *except OSError as e:* *print("Sensor read error:", e)*
	**Document the Issue and Fix** • Keep a simple debugging log: ○ Error observed ○ Fix applied
Extract sensor data	
 MPY: soft reboot Temp: 30.1°C | Humidity: 40.9% Temp: 30.0°C | Humidity: 40.7% Temp: 30.0°C | Humidity: 40.8% Temp: 30.0°C | Humidity: 41.0% Temp: 30.0°C | Humidity: 41.2% Temp: 30.0°C | Humidity: 41.3% Temp: 30.0°C | Humidity: 40.7% Temp: 30.0°C | Humidity: 40.3% Temp: 30.0°C | Humidity: 40.4% Temp: 30.0°C | Humidity: 40.8%	**Store data in variables** • Use clear, descriptive variable names: *temperature_celsius = sensor.temperature()* *humidity_percent = sensor.humidity()*
 import time from machine import ADC # Initialize the Soil Moisture Sensor (AO → A0) soil_sensor = ADC(0) # Create a list to store moisture readings moisture_readings = [] print("Collecting soil moisture data...") while True: # Read current value (0-1023) value = soil_sensor.read() # Store reading in the list moisture_readings.append(value) # Display reading print("Soil moisture value:", value) print("All readings so far:", moisture_readings) # Wait 5 seconds before the next reading time.sleep(5)	**Collect data over time** • Demonstrate how to store data in lists for monitoring trends: *temperature_readings = []* *humidity_readings = []* *temperature_readings.append(temperature_celsius)* *humidity_readings.append(humidity_percent)*

<table>
<tr>
<td>

```
import time
from machine import ADC

# Initialize the Soil Moisture Sensor (AO → A0)
soil_sensor = ADC(0)

# Create a list to store moisture readings
moisture_readings = []

print("Collecting soil moisture data...")

while True:
    # Read current value (0-1023)
    value = soil_sensor.read()

    # Store reading in the list
    moisture_readings.append(value)

    # Display reading
    print("Soil moisture value:", value)
    print("All readings so far:", moisture_readings)

    # Wait 5 seconds before the next reading
    time.sleep(5)
```

```
Soil Moisture Value: 420
Soil Moisture Value: 715
Soil Moisture Value: 310
```

</td>
<td>

Format data for clarity

- Create readable output strings:

data_string = "Temp: {}°C | Humidity: {}%".format(temperature_celsius, humidity_percent)
print(data_string)

</td>
</tr>
<tr>
<td>

```
import time
from machine import ADC

# Initialize the Soil Moisture Sensor (AO → A0)
soil_sensor = ADC(0)

print("Logging soil moisture data...")

while True:
    # Read soil moisture value
    value = soil_sensor.read()

    # Create readable string
    data_string = "Soil Moisture Value: {}".format(value)
    print(data_string)

    # Write data to a log file
    with open('data_log.txt', 'a') as file:
        file.write(data_string + '\n')

    time.sleep(5)  # Log every 5 seconds
```

</td>
<td>

Prepare Data for Storage or Transmission

- Write data to files

with open('data_log.txt', 'a') as file:
file.write(data_string + '\n')

It will append each reading to data_log.txt on the ESP8266's internal filesystem. Each new value is saved on a new line.

Note: The ESP8266 has limited storage. For longer-term logging, you'd typically move data to an SD card or send it over Wi-Fi (MQTT/HTTP)

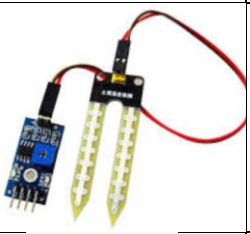
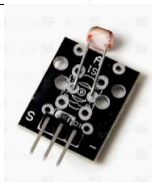
</td>
</tr>
</table>

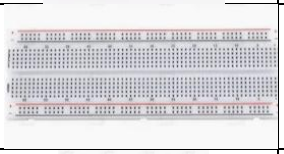
3.1.2 Tools and materials

Tools	Picture	Quantity[2]	Comment
PC & Monitor & keyboard & mouse (Windows OS installed) Or Laptop with USB port With installed (Python 3.8+, Thonny IDE, and USB Drivers: CP210x / CH340 Drivers and Administrative privileges to install drivers/software		1 per station	
Internet connection With WiFi network credentials		1 per station	
Multimeter		1 per station	
Anti-Static Mat ESD Safe		1 per station	
Power Bank		1 per station	
5V DC Adapter micro-usb		1 per station	
Storage drawer cabinet for components		1 per station	

[2] The quantity expressed for workshop means that the item can be used for different modules in the same workshop, this imply that two modules that are using the item could not be delivered at the same time. Warning! before purchasing the item it should be verified if it already exists in the workshop to avoid unnecessary duplications. Also, the list should be reviewed and verified by an experienced professional.

Tools	Picture	Quantity[2]	Comment
Basic tool kit (screwdrivers, pliers, wire cutters)		1 per station	

Materials and consumables	Picture	Quantity (per station)	Comment
ESP8266 Development Board (e.g., NodeMCU, Wemos D1 Mini)		1	
Micro USB data cable		1	Data cable, not just charging
DHT22/DHT11 sensor module		1	
Soil Moisture sensor module		1	
Light sensor module		1	
LED		5	
Resistors 470 ohm, 1kohm and 10 kohm		10 each	
Push button module		2	

Materials and consumables	**Picture**	**Quantity (per station)**	**Comment**
ON/OFF Switch		2	
Breadboard		1	
Jumper wires (female-male and male- male)		1 pack of each	
18650 Li-Ion rechargeable battery		1	
18650 Li-Ion rechargeable battery holder with wire		1	

3.2 Assessment standards – Unit 2

3.2.1 Formative assessment

Trainee Name:	**Date:**	
Ability / learning outcome	**Criteria for competence unit 2 - Writing IoT data acquisition scripts**	**Yes / No**
Write Python scripts to collect data from IoT sensors using MicroPython	Does the candidate open the Thonny IDE correctly and connect the ESP8266/ESP32 via USB?	
	Does the candidate verify that Thonny detects the device and that the REPL is functional?	
	Does the candidate create a new .py script and save it with a clear filename?	
	Does the candidate import necessary libraries (machine, time, and sensor-specific) at the top of the script?	
	Does the candidate write an initial test line (print("IoT Data Acquisition Script Running")) and run the script successfully in Thonny?	
	Does the candidate identify and correctly import the required core and sensor-specific libraries?	
	Does the candidate test the import statements and handle errors like ModuleNotFoundError if they occur?	
	Does the candidate review sensor types, GPIO pins, and pin modes before coding?	
Interface sensors with code to read and display their data	Does the candidate write initialization code for the DHT11 sensor, including GPIO pin setup?	
	Does the candidate initialize the DHT11 sensor object using the sensor library and test it?	
	Does the candidate write initialization code for the soil moisture sensor using ADC and test reading values?	
	Does the candidate write initialization code for the light sensor using ADC and test reading values?	
	Does the candidate connect the DHT11 sensor to the ESP8266 with correct VCC, GND, and data pins?	
	Does the candidate create code to read the DHT11 sensor data and display temperature and humidity in the REPL?	
	Does the candidate embed sensor reading and displaying in a continuous loop with appropriate delays?	
	Does the candidate connect the soil moisture sensor to the ESP8266 correctly and write code to read and display analog values continuously?	
Debug data acquisition errors	Does the candidate use print statements effectively to identify where the script may fail?	
	Does the candidate verify hardware connections, ensuring sensors are connected to the correct GPIO pins and power lines?	
	Does the candidate verify GPIO pin configurations in code match the physical wiring and correct library imports?	
	Does the candidate handle sensor-specific runtime errors, e.g., using try-except for OSError or other read failures?	
	Does the candidate document observed errors and applied fixes in a debugging log?	
Extract sensor data	Does the candidate store sensor readings in descriptive variables (e.g., temperature_celsius, humidity_percent)?	
	Does the candidate collect multiple readings over time and store them in lists for monitoring trends?	
	Does the candidate format data for clear output using strings before printing?	
	Does the candidate save data to a file (data_log.txt) on ESP8266 or prepare it for transmission?	
	Does the candidate ensure the saved data is appended and formatted for later use?	

3.2.2 Summative assessment

Trainee name:	**Date:**	
Assessor name:	**Date:**	

The candidate is considered competent if he demonstrates a best practice performance in all the following criteria for the assessment related to the competence unit 2 - **Writing IoT data acquisition scripts**

Criteria for assessment	**Qualifier**	**YES**	**NO (*)**
Script setup and initialization	The candidate should be able to, in 20 minutes or less, create and save a MicroPython script in Thonny IDE, import the required libraries, initialize the GPIO pins with correct modes, and initialize the connected sensor using the appropriate library commands without errors.	☐	☐
Data acquisition and display	The candidate should be able to, in 20 minutes or less, implement a while loop in the MicroPython script to continuously read sensor data using the correct methods, display the extracted data clearly in the REPL using print() statements, and include an appropriate delay between readings to ensure stable operation.	☐	☐
Hardware connection and verification	The candidate should be able to, in 10 minutes or less, correctly connect the IoT sensor to the microcontroller according to the wiring diagram, verify physical connections and power, and confirm that the microcontroller detects the sensor setup for further data acquisition.	☐	☐
Code initialization, reading, and display	The candidate should be able to, in 15 minutes or less, write and execute a MicroPython script that initializes the GPIO pins and sensor, reads data using the appropriate library methods, and displays live sensor data clearly in the REPL without errors.	☐	☐
Debug data acquisition errors	The candidate should be able to, in 10 minutes or less, identify errors occurring during sensor data acquisition, apply systematic debugging steps including checking hardware connections, verifying GPIO configurations, inspecting library imports, using print statements, and applying try-except handling, and confirm that the script executes without errors while displaying correct live sensor data in the REPL.	☐	☐
Extract sensor data	The candidate should be able to, in 10 minutes or less, extract sensor data by reading it using the appropriate library methods, store the data in clearly named variables, format the data for clarity, and display it in the REPL to confirm correct and stable readings suitable for analysis, logging, or further IoT processing.	☐	☐

Criteria (*)	**Evidence of non-compliance if any (**)**

Assessor signature:	**Date:**	
Trainee signature:	**Date:**	

4 Competence Unit 3: Developing programs to control devices

4.1 Training standards – Unit 3

4.1.1 Training guidelines

<table>
<tr>
<td>Ability / Learning outcome:

• Write control programs in Python for actuators and peripheral devices
• Test control logic to actuate devices based on conditions
• Use GPIOs and PWM outputs
• Refine control logic using hardware connections

Training site:

Indoor room of approximately 4 x 5m equipped with:
• Wide window
• Fire extinguisher
• Many Electrical switches and sockets for the PCs
• Fixed wooden high-chair furniture suitable for electronics lab activities
• Desks and benches where we can place PCs, tools, and materials for practical work
• Closet and drawers for organizing and storing electronic components and lab materials

A soil area (or soil containers) inside the room for testing soil moisture sensors</td>
<td>Required Tools:

• PC & Monitor
• Internet connection
• Multimeter
• Power Bank
• DC Adapter 12V
• 5V DC Adapter micro USB
• Storage boxes for components
• Basic tool kit (screwdrivers, pliers, wire cutters)
• Battery 12V , 3 A or more
• Anti-Static Mat ESD Safe</td>
<td>Required Materials and Consumables:

• ESP8266 Development Board (e.g., NodeMCU, Wemos D1 Mini)
• Micro USB data cable (ensure it is a data cable, not just charging)
• DHT22/DHT11 sensor module
• Water level sensor module
• Soil Moisture sensor module
• Ambient light sensor module
• 18650 Li-Ion rechargeable battery + holder
• RGB LEDs
• LEDs
• Resistors
• Buzzer
• Push buttons module
• switches
• Servo motor
• DC motor
• H-bridge module
• Relays module
• Pump
• Breadboard
• Jumper wires (female-male, male-male)
• breadboard Power Supply Module 3.3V 5V</td>
</tr>
<tr>
<td colspan="3">Write control programs in Python for actuators and peripheral devices</td>
</tr>
<tr>
<td></td>
<td colspan="2">Open the development environment

• Launch Thonny IDE.
• Connect the ESP8266 development board via USB to the PC.
• Confirm the ESP8266 is connected, and the COM port is recognized.
• Open a new Python file for the control program.</td>
</tr>
</table>

```
import machine
import time
led = machine.Pin(2, machine.Pin.OUT)
relay = machine.Pin(5, machine.Pin.OUT)
```

Import required libraries

Machine library is used For GPIO and PWM control. Time library is used for delays and timing control.

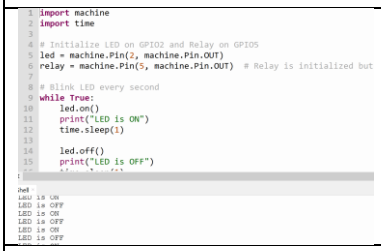
- Start the code by typing:
  import machine
  import time
- Click on File then select save as. Save the script as control_actuator.py for clarity.
- Identify and Initialize GPIO Pins
- Set the GPIO pins that will control actuators as outputs. Example:
  led = machine.Pin(2, machine.Pin.OUT)
  For a relay or motor driver:
  relay = machine.Pin(5, machine.Pin.OUT)
- Emphasize using clear, descriptive variable names for each actuator.

```
import machine
import time

# Initialize LED on GPIO2 and Relay on GPIO5
led = machine.Pin(2, machine.Pin.OUT)
relay = machine.Pin(5, machine.Pin.OUT)  # Re

# Blink LED every second
while True:
    led.on()
    print("LED is ON")
    time.sleep(1)

    led.off()
    print("LED is OFF")
    time.sleep(1)
```

Write basic control logic

- Start by turning the actuator ON and OFF:
  led.on() # Turns LED on
  time.sleep(1) # Waits 1 second
  led.off() # Turns LED off
  time.sleep(1)
- Test by creating a simple blinking loop:
  while True:
  led.on()
  print("LED is ON")
  time.sleep(1)
  led.off()
  print("LED is OFF")
  time.sleep(1)

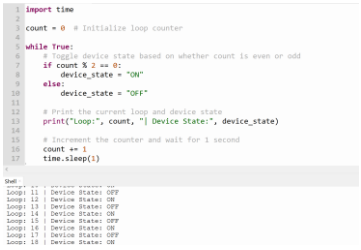
Tip: To stop infinite loops during execution in Thonny, use Ctrl+ C.



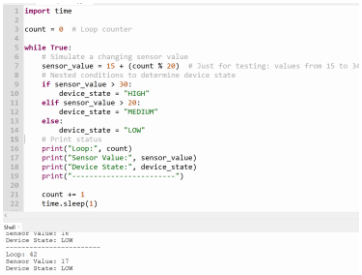
Save and upload the script

- Save the script with a clear name.
- Run the script from Thonny to observe the actuator response.

Test control logic to actuate devices based on conditions

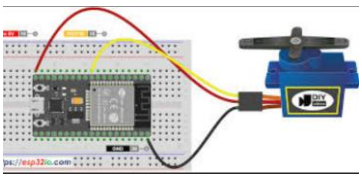
<table>
<tr>
<td> 
</td>
<td>Create the script for logical testing

• Open Thonny IDE and create test_control_logic.py.
• Import necessary libraries:
import time
• Initialize a simulated “device state”:
device_state = "OFF"
• Write a loop to test logic:
count = 0
while True:
if count % 2 == 0:
device_state = "ON"
else:
device_state = "OFF"
print("Loop:", count, "Device State:", device_state)
count += 1
time.sleep(1)
• Execute the script.
• Observe the REPL output
• Add comments for clarity

Syntax:
if condition:
action
else:
alternative action</td>
</tr>
<tr>
<td> 
</td>
<td>Modify conditions for practice

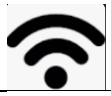
• Add nested conditions (e.g., multiple states).
• Simulate a sensor value:
sensor_value = 25
if sensor_value > 20:
device_state = "ON"
else:
device_state = "OFF”

Tip: It is better to use descriptive and clear variable names example temperature_celsius.</td>
</tr>
<tr>
<td colspan="2">Use GPIOs and PWM outputs</td>
</tr>
</table>

	Write GPIO control code • Open Thonny IDE and create a new script (gpio_pwm_control.py). • Import libraries: *import machine* *import time* • Initialize a GPIO pin as output: *led = machine.Pin(2, machine.Pin.OUT)* • Control the pin to turn an LED ON and OFF: *led.on()* *time.sleep(1)* *led.off()* *time.sleep(1)* • Use print() statements to indicate the action if hardware is not connected.
	Write PWM control code • Explain PWM as a way to simulate analog output by varying the duty cycle of digital signals • Initialize PWM on a GPIO pin: pwm = machine.PWM(machine.Pin(2)) pwm.freq(1000) # Set frequency to 1kHz • Change duty cycle to control intensity or speed: for duty in range(0, 1024, 200): pwm.duty(duty) print("Duty cycle:", duty) time.sleep(0.5) • Run the script and observe outputs in the REPL. • Try one more example: Dimming an LED Using PWM *import machine* *import time* *pwm = machine.PWM(machine.Pin(2))* *pwm.freq(1000) # 1kHz frequency* *while True:* *for duty in range(0, 1024, 100): # Increase brightness* *pwm.duty(duty)* *print("Duty:", duty)* *time.sleep(0.2)* *for duty in range(1023, -1, -100): # Decrease brightness* *pwm.duty(duty)* *print("Duty:", duty)* *time.sleep(0.2)*
Refine control logic using hardware connections	
	**Set up the hardware** • Connect ESP8266 to Laptop ○ Use a USB Data cable (not a charge only cable) to connect the ESP8266 board to your laptop. • Open Thonny • Go to Tools > Options > Interpreter: ○ Select MicroPython (ESP8266) ○ Set the correct COM port (usually auto-detected)

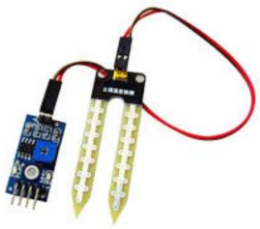
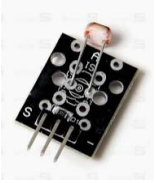
	• Connect the circuit on the Breadboard o GPIO Output Example: Connect GPIO5 (D1) → first terminal of the resistor (220Ω) Connect the second terminal of the resistor to the positive terminal of the LED (long pin) Connect the second pin of the LED (short leg) to the GND (0V) o PWM Example: Connect GPIO14 (D5) → servo signal wire Servo VCC → 3.3V or external 5V (depending on model) Servo GND → GND o Input Device Example: Push button: One pin to GPIO4 (D2), other to GND (Use internal pull-up in code)
	Write initial control logic • Open Thonny to write a MicroPython script with GPIO and PWM control logic. Example snippet: *from machine import Pin, PWM* *from time import sleep* *led = Pin(5, Pin.OUT) # GPIO5* *servo = PWM(Pin(14), freq=50) # GPIO14 for PWM* *button = Pin(4, Pin.IN, Pin.PULL_UP)* *while True:* *if button.value() == 0:* *led.on()* *servo.duty(40) # Example position* *else:* *led.off()* *servo.duty(75)* *sleep(0.1)* • Save the file as main.py in Thonny • Click Run or press F5 to upload the code • Confirm that the ESP8266 executes the logic automatically on reboot Note that the code will auto-run again whenever the board is reset or powered on.
	Refinement process • Observe the hardware's behavior: o Is the LED reacting as expected? o Is the servo reaching the right angle? o Are transitions smooth? • Add delays to debounce buttons • Calibrate servo angles • Use print statements in Thonny REPL for live debugging • Adjust logic thresholds or duty cycle values • Modify the code and redeploy. • Recheck behavior with each change. • Use comments to document changes or logic explanations. • Have multiple versions (e.g v1, v2 ,…) to track development progress.

4.1.2 Tools and materials

Tools	Picture	Quantity[3]	Comment
PC & Monitor & keyboard & mouse (Windows OS installed) Or Laptop with USB port Python 3.8+, Thonny IDE , and USB Drivers: CP210x / CH340 Drivers Administrative privileges to install drivers/software WiFi network credentials		1 per station	
Internet connection	 	1 per class	
Multimeter		1 per station	
Power Bank		1 per station	
5V DC Adapter micro-usb		1 per station	
12V DC Adapter		1 per station	

[3] The quantity expressed for workshop means that the item can be used for different modules in the same workshop, this imply that two modules that are using the item could not be delivered at the same time. Warning! before purchasing the item it should be verified if it already exists in the workshop to avoid unnecessary duplications. Also, the list should be reviewed and verified by an experienced professional. Also, the list should be reviewed and verified by an experienced professional.

Tools	Picture	Quantity[3]	Comment
Storage drawer cabinet for components		1 per station	
Battery 12 V , 3 A or more + clips		1 per classroom	
Anti-Static Mat ESD Safe		1 per station	

Materials and consumables	**Picture**	**Quantity per station**	**Comment**
ESP8266 Development Board (e.g., NodeMCU, Wemos D1 Mini)		1	
Micro USB data cable		1	(ensure it is a data cable, not just charging)
DHT22/DHT11 sensor module		1	
Water level sensor module		1	
Soil Moisture sensor module		1	
Light sensor module		1	
RGB LED		1	
LED		5	
Resistors 470 ohm, 1kohm and 10 kohm		10 each	

Buzzer		1	
Push button module		2	
ON/OFF Switch		2	
Servo Motor		1	
DC Motor		1	
Relay Module		1	
H-Bridge Module		1	
Pump 12V		1 per classroom	
Breadboard		1	
Breadboard Power Supply Module 3.3V 5V		1	

Jumper wires (female-male and male- male)		1 pack of each	

4.2 Assessment standards – Unit 3

4.2.1 Formative assessment

Trainee Name:	**Date:**	
Ability / learning outcome	**Criteria for competence unit 3 - Developing programs to control devices**	**Yes / No**
Write control programs in Python for actuators and peripheral devices	Does the candidate open Thonny IDE and connect the ESP8266 board correctly via USB?	
	Does the candidate confirm the ESP8266 is detected and the correct COM port is recognized?	
	Does the candidate create a new Python file for the control program and save it with a clear name (e.g., control_actuator.py)?	
	Does the candidate import the required libraries (machine and time) at the top of the script?	
	Does the candidate identify and initialize GPIO pins for actuators using correct syntax and descriptive variable names?	
	Does the candidate write basic control logic to turn an actuator ON and OFF with correct use of time.sleep()?	
	Does the candidate create and test a simple blinking loop to verify actuator response?	
	Does the candidate save and upload the script to observe the actuator functioning?	
Test control logic to actuate devices based on conditions	Does the candidate create a Python script for testing control logic (test_control_logic.py)?	
	Does the candidate import necessary libraries and initialize simulated device states?	
	Does the candidate write a loop to test logical conditions and display output in the REPL?	
	Does the candidate use if-else statements correctly to switch device states based on conditions?	
	Does the candidate modify and test nested conditions or simulate sensor values for practice?	
	Does the candidate use descriptive variable names to improve code readability and maintainability?	
Use GPIOs and PWM outputs	Does the candidate initialize GPIO pins as outputs and control devices (e.g., LEDs) using on() and off()?	
	Does the candidate implement print statements to indicate actions when hardware is not connected?	
	Does the candidate initialize PWM on GPIO pins and set frequency correctly?	
	Does the candidate vary the duty cycle to control intensity or speed of actuators?	
	Does the candidate write and test a loop to gradually increase and decrease PWM output (e.g., LED dimming)?	
	Does the candidate observe PWM outputs in the REPL and confirm correct behavior?	
Refine control logic using hardware connections	Does the candidate connect the ESP8266 board to the laptop using a proper USB data cable and select the correct MicroPython interpreter in Thonny?	
	Does the candidate assemble hardware circuits for GPIO outputs, PWM devices (e.g., servo), and input devices (e.g., push button) correctly?	
	Does the candidate write control logic in Python that reacts to input devices (e.g., button presses) and controls actuators accordingly?	
	Does the candidate test and observe hardware behavior to ensure actuators respond as expected?	
	Does the candidate refine the control logic by adding delays, calibrating actuators, adjusting thresholds, and redeploying the code?	
	Does the candidate use print statements for live debugging and document changes using comments?	
	Does the candidate maintain multiple script versions to track development progress and improvements?	

4.2.2 Summative assessment

Trainee name:	**Date:**		
Assessor name:	**Date:**		
The candidate is considered competent if he demonstrates a best practice performance in all the following criteria for the assessment related to the competence unit 3 - **Developing programs to control devices**			
Criteria for assessment	**Qualifier**	**YES**	**NO (*)**
Write control programs in MicroPython for actuators and peripheral devices	The candidate should be able to, in 15 minutes or less, create and save a MicroPython script in Thonny IDE, import the required libraries, initialize GPIO pins for actuators or peripheral devices, write and execute control logic to switch the devices ON and OFF within a loop, and verify correct operation through REPL output without errors.	☐	☐
Test conditional control Logic	The candidate should be able to, in 15 minutes or less, create and save a MicroPython script in Thonny IDE, write a while loop with a condition that changes a simulated device state variable, display the device state in the REPL using print() statements, execute the script, and verify that the device state changes correctly according to the condition logic without errors.	☐	☐
Configure GPIO pins to control digital devices using MicroPython and troubleshoot common GPIO errors	The candidate should be able to, in 10 minutes or less, to initialize at least one GPIO pin as output and demonstrate digital control by toggling the state (HIGH/LOW) and to troubleshoot common GPIO errors and implement necessary corrections.	☐	☐
Generate and modify PWM signals to control analog behavior	The candidate should be able to, in 10 minutes or less, to set up a PWM output on a GPIO pin and adjust duty cycle or frequency to create observable hardware response and to troubleshoot common PWM errors and implement necessary corrections.	☐	☐
Refine control logic using hardware connections	The candidate should be able to, in 30 minutes or less, connect a sensor and an actuator to the ESP8266, upload a control program, run the code in both simulation and hardware, verify the behavior, analyze the results, and refine the control logic to ensure proper system response.	☐	☐
Criteria (*)	**Evidence of non-compliance if any (**)**		
Assessor signature:	**Date:**		

Trainee signature:	**Date:**	

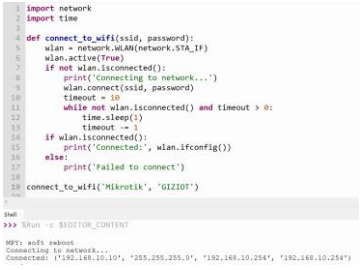
5 Competence Unit 4: Establishing network connectivity using Python for cloud integration

5.1 Training standards – Unit 4

5.1.1 Training guidelines

Ability / Learning outcome:	**Required Tools:**	**Required Materials and Consumables:**
• Configure network settings for IoT devices • Write Python scripts to connect devices to cloud platforms • Transmit sensor data securely to the cloud • Troubleshoot basic network connectivity and data transmission issues **Training site:** Indoor room of approximately 4 x 5 m equipped with: • Wide window • Fire extinguisher • Many Electrical switches and sockets for the PCs • Fixed wooden high-chair furniture suitable for electronics lab activities • Desks and benches where we can place PCs, tools, and materials for practical work • Closet and drawers for organizing and storing electronic components and lab materials A soil area (or soil containers) inside the room for testing soil moisture sensors	• PC & Monitor • Internet connection • Multimeter • Power Bank • DC Adapter 12V • 5V DC Adapter micro USB • Storage boxes for components • Basic tool kit (screwdrivers, pliers, wire cutters) • Anti-Static Mat ESD Safe	• ESP8266 Development Board (e.g., NodeMCU, Wemos D1 Mini) • Micro USB data cable (ensure it is a data cable, not just charging) • DHT22/DHT11 sensor module • Water level sensor module • Soil Moisture sensor module • Ambient light sensor module • 18650 Li-Ion rechargeable battery + holder • RGB LEDs • LEDs • Resistors • Buzzer • Push buttons module • switches • Ultrasonic distance sensor module • Breadboard • Jumper wires (female-male , male-male) • breadboard Power Supply Module 3.3V 5V

Configure network settings for IoT devices



Configure Wi-Fi Settings Using MicroPython

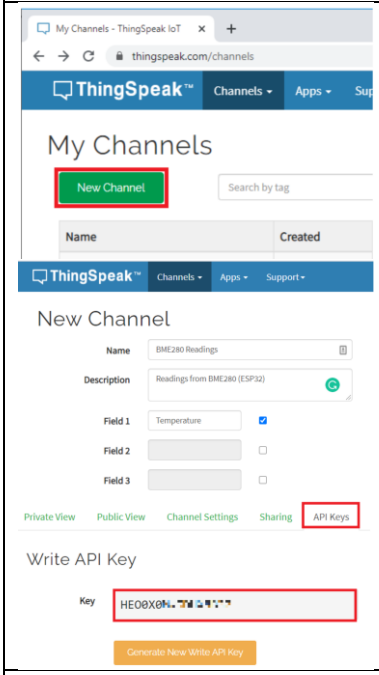
- Make sure the device is within range of the Wi-Fi router
- Ensure your network allows 2.4 GHz Wi-Fi (ESP8266/ESP32 do not support 5 GHz)
- Write a Basic Wi-Fi Connection Script

```
import network
import time

def connect_to_wifi(ssid, password):
    wlan = network.WLAN(network.STA_IF)
    wlan.active(True)
    if not wlan.isconnected():
        print('Connecting to network...')
```

```
    wlan.connect(ssid, password)
    timeout = 10
    while not wlan.isconnected() and timeout > 0:
      time.sleep(1)
      timeout -= 1
  if wlan.isconnected():
    print('Connected:', wlan.ifconfig())
  else:
    print('Failed to connect')

connect_to_wifi('Your_SSID', 'Your_PASSWORD')
```

- Save the file as boot.py or wifi_connect.py.
- Upload to ESP using Thonny's file manager.
- Restart the board and monitor output on the Thonny shell.

Write Python scripts to connect devices to cloud platforms



Setup the Cloud Account (ThingSpeak)

- Enter: https://thingspeak.mathworks.com in your browser
- Click the "Get Started For Free" button to create a ThingSpeak account.
- Open Channels -> select My Channels -> Create a new channel.
- Type a name for your channel and add a description.
  - If you want to publish multiple readings (like humidity and pressure), you can enable many fields.
- Click the Save Channel button to create and save your channel.

API

To send values from the ESP8266 or ESP32 to ThingSpeak, you need the Write API Key. Open the "API Keys" tab and copy the Write API Key and Channel ID to a safe place because you'll need it in a moment.

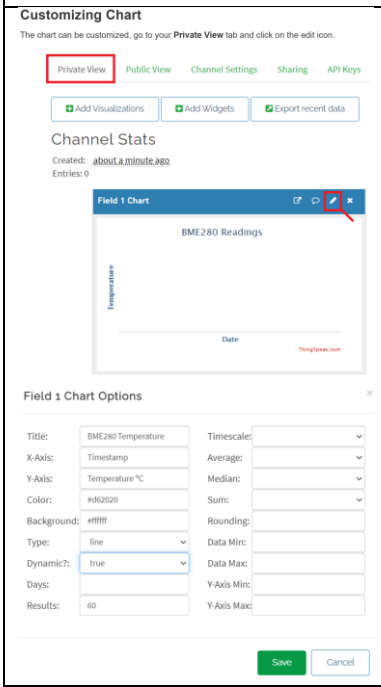
Important note: Do not share your API key publicly as it gives access to your data.

Write MicroPython Script: HTTP Post to ThingSpeak

- Type the following code

```
import network
import urequests
import time

def connect_wifi(ssid, password):
  wlan = network.WLAN(network.STA_IF)
  wlan.active(True)
  if not wlan.isconnected():
    wlan.connect(ssid, password)
    while not wlan.isconnected():
      time.sleep(1)
```

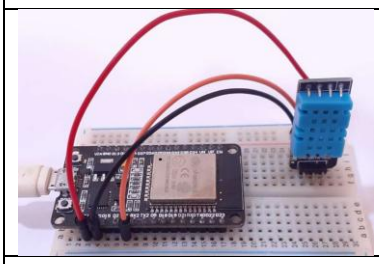
```
    print("Connected:", wlan.ifconfig())

def send_to_thingspeak(api_key, value):
    url = "https://api.thingspeak.com/update?api_key={}&field1={}".format(api_key, value)
    response = urequests.get(url)
    print("Server response:", response.text)
    response.close()

# Example usage
connect_wifi('YourSSID', 'YourPassword')
while True:
    sensor_value = 42  # Replace with actual sensor reading
    send_to_thingspeak('YOUR_API_KEY', sensor_value)
    time.sleep(15)
```



Customize the Chart

- Go to your Private View tab
- Click on the edit icon.
    - Give a title to your chart,
    - Customize the background color, x and y axis, and much more
    - When you're done, press the "Save" button.

Transmit sensor data securely to the cloud



Wire the DHT11 or DHT22 to ESP8266 or ESP32

- Insert the DHT11 sensor onto the breadboard.
- Connect the VCC pin of the sensor to the 3.3V pin on the ESP.Connect GND to GND.
- Connect the data pin of the sensor to GPIO2 on the ESP.

Verify sensor readings using a simple Python script on Thonny

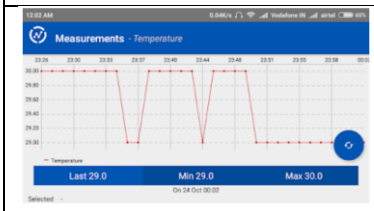
- Type the following script:

```
import dht
import machine
import time
```

	sensor = dht.DHT11(machine.Pin(2)) *while True:* *sensor.measure()* *temp = sensor.temperature()* *hum = sensor.humidity()* *print('Temp:', temp, 'C Humidity:', hum, '%')* *time.sleep(2)*
	Secure Wi-Fi Connection Setup • Use network module in MicroPython. *import network* *sta_if = network.WLAN(network.STA_IF)* *sta_if.active(True)* *sta_if.connect('YOUR_SSID', 'YOUR_PASSWORD')* *while not sta_if.isconnected():* *pass* *print('Connected:', sta_if.ifconfig())*
	Write a basic script for connecting to Wi-Fi, reading the DHT sensor, and sending data to ThingSpeak • Type the following script: *import network* *import urequests* *import utime* *import dht* *from machine import Pin* *# Wi-Fi credentials* *ssid = 'YOUR_SSID'* *password = 'YOUR_PASSWORD'* *# Connect to Wi-Fi* *sta = network.WLAN(network.STA_IF)* *sta.active(True)* *sta.connect(ssid, password)* *while not sta.isconnected():* *print("Connecting to Wi-Fi...")* *utime.sleep(1)* *print("Connected, IP:", sta.ifconfig()[0])* *# Sensor setup* *sensor = dht.DHT11(Pin(2)) # GPIO2 is D4* *# ThingSpeak API* *api_key = 'YOUR_API_KEY'*

```
url = 'https://api.thingspeak.com/update'

# Loop
while True:
    sensor.measure()
    temp = sensor.temperature()
    hum = sensor.humidity()
    print(f'Temp: {temp}°C | Humidity: {hum}%')

    # Send data
    full_url = f"{url}?api_key={api_key}&field1={temp}&field2={hum}"
    try:
        response = urequests.get(full_url)
        print("Sent to ThingSpeak:", response.text)
        response.close()
    except:
        print("Failed to send data.")

    utime.sleep(15)
```

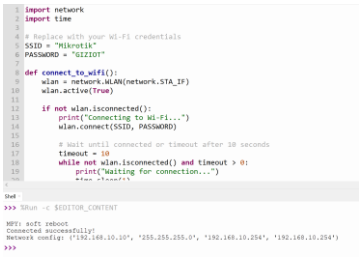
- Save the code as main.py.
- Upload it to the ESP using Thonny.
- Run it and monitor the REPL output for logs and errors.



Verify on cloud

- Log in to ThingSpeak.com.
- Open your channel dashboard.
- View real-time temperature and humidity graphs updating every 15 seconds.

Troubleshoot basic network connectivity and data transmission issues



```
import network
import time

# Replace with your Wi-Fi credentials
SSID = "Mikrotik"
PASSWORD = "GIZIOT"

def connect_to_wifi():
    wlan = network.WLAN(network.STA_IF)
    wlan.active(True)

    if not wlan.isconnected():
        print("Connecting to Wi-Fi...")
        wlan.connect(SSID, PASSWORD)

        # Wait until connected or timeout after 10 seconds
        timeout = 10
        while not wlan.isconnected() and timeout > 0:
            print("Waiting for connection...")
```

```
>>> %Run -c $EDITOR_CONTENT

MPY: soft reboot
Connected successfully!
Network config: ('192.168.10.10', '255.255.255.0', '192.168.10.254', '192.168.10.254')
>>>
```

Read device status

- Access serial monitor output
- Review connection status messages:
  - “Connecting to Wi-Fi…”
  - “Connection successful” + IP address
  - “Connection failed” or authentication error
- Demonstrate how to check IP and network info: Use sta_if.ifconfig() in MicroPython to retrieve and interpret IP configuration
- Test this script that attempts Wi-Fi connection:

```
import network
import time
```

	# Replace with your Wi-Fi credentials *SSID = "Your_SSID"* *PASSWORD = "Your_PASSWORD"* *def connect_to_wifi():* *wlan = network.WLAN(network.STA_IF)* *wlan.active(True)* *if not wlan.isconnected():* *print("Connecting to Wi-Fi...")* *wlan.connect(SSID, PASSWORD)* *# Wait until connected or timeout after 10 seconds* *timeout = 10* *while not wlan.isconnected() and timeout > 0:* *print("Waiting for connection...")* *time.sleep(1)* *timeout -= 1* *if wlan.isconnected():* *print("Connected successfully!")* *print("Network config:", wlan.ifconfig())* *else:* *print("Failed to connect.")* *connect_to_wifi()*
	Handle errors in Python scripts • Implement error logging by adding print statements or log errors for diagnosis • Sample code demonstration: Show a simple script with error handling for HTTP requests *import urequests* *try:* *response =* *urequests.get('https://api.thingspeak.com/update?api_key=YOUR_API_KEY&field1=25')* *if response.status_code == 200:* *print("Data sent successfully")* *else:* *print("Failed with status code:", response.status_code)* *response.close()* *except Exception as e:* *print("Error occurred:", e)*
	Debug data transmission failures • Check HTTP responses: When sending data, review the returned status code. For example: 200 – Data sent successfully 403 – Access forbidden; check API key or permissions 404 – Resource not found; verify URL or endpoint 500 – Server error; try again later or check server logs • Verify data on cloud platforms by logging in to the platform dashboard and confirm whether the data has been received, stored, or rejected.

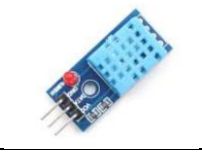
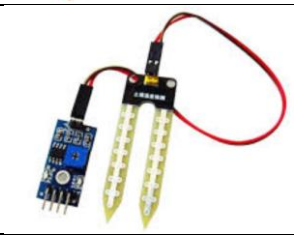
	• Manage API rate limits: If multiple requests are sent in a short time, pause between requests to avoid exceeding the allowed limit. • Test network connectivity by pinging the Wi-Fi network or send simple test requests to confirm internet access before sending data.

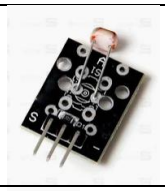
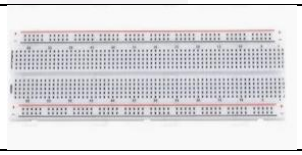
5.1.2 Tools and materials

Tools	Picture	Quantity[4]	Comment
PC & Monitor & keyboard & mouse (Windows OS installed) Or Laptop with USB port Python 3.8+, Thonny IDE , and USB Drivers: CP210x / CH340 Drivers Administrative privileges to install drivers/software WiFi network credentials		1 per station	
Internet connection		1 per station	
Multimeter		1 per station	
Power Bank		1 per station	
5V DC Adapter micro-usb		1 per station	
12V DC Adapter		1 per station	

[4] The quantity expressed for workshop means that the item can be used for different modules in the same workshop, this imply that two modules that are using the item could not be delivered at the same time. Warning! before purchasing the item it should be verified if it already exists in the workshop to avoid unnecessary duplications. Also, the list should be reviewed and verified by an experienced professional. Also, the list should be reviewed and verified by an experienced professional.

Tools	Picture	Quantity[4]	Comment
Storage drawer cabinet for components		1 per station	
Basic tool kit (screwdrivers, pliers, wire cutters)		1 per station	
Anti-Static Mat ESD Safe		1 per station	

Materials and consumables	Picture	Quantity	Comment
ESP8266 Development Board (e.g., NodeMCU, Wemos D1 Mini)		1	
Micro USB data cable		1	(ensure it is a data cable, not just charging)
DHT22/DHT11 sensor module		1	
Water level sensor module		1	
Soil Moisture sensor module		1	

Materials and consumables	Picture	Quantity	Comment
Light sensor module		1	
RGB LED		1	
LED		5	
Resistors 470 ohm, 1kohm and 10 kohm		10 each	
Buzzer		1	
Push button module		2	
ON/OFF Switch		2	
Breadboard		1	
Breadboard Power Supply Module 3.3V 5V		1	
Jumper wires (female-male and male- male)		1 pack of each	
18650 Li-Ion rechargeable battery		2	
18650 Li-Ion rechargeable battery holder with wire		2	

5.2 Assessment standards – Unit 4

5.2.1 Formative assessment

Trainee Name:	**Date:**	
Ability / learning outcome	**Criteria for competence unit 4 - Establishing network connectivity using Python for cloud integration**	**Yes / No**
Configure network settings for IoT devices	Does the candidate ensure the IoT device is within Wi-Fi range and that the network supports 2.4 GHz?	
	Does the candidate write a basic MicroPython script to connect the device to Wi-Fi?	
	Does the candidate correctly use network.WLAN (network.STA_IF) to activate the Wi-Fi interface?	
	Does the candidate implement a connection loop that waits for successful Wi-Fi connection with a timeout?	
	Does the candidate display connection status and IP configuration in the REPL?	
	Does the candidate save the Wi-Fi script correctly (e.g., boot.py or wifi_connect.py) and upload it to the device?	
	Does the candidate restart the board and monitor the Thonny shell output for connection confirmation?	
Write Python scripts to connect devices to cloud platforms	Does the candidate create a cloud account (e.g., ThingSpeak) and configure a channel with fields for sensor data?	
	Does the candidate retrieve and securely store the Write API Key and Channel ID?	
	Does the candidate write a MicroPython script to connect to the cloud via HTTP requests (urequests)?	
	Does the candidate import necessary libraries (network, urequests, time) and establish Wi-Fi connectivity in the script?	
	Does the candidate correctly format the HTTP request URL to send data to the cloud channel?	
	Does the candidate implement a loop to periodically send sensor or test data to the cloud?	
	Does the candidate customize the cloud platform's chart display and verify the data fields are correctly mapped?	
Transmit sensor data securely to the cloud	Does the candidate wire the DHT11/DHT22 sensor correctly to the ESP8266/ESP32 (VCC, GND, data pin to correct GPIO)?	
	Does the candidate verify sensor readings using a Python script in Thonny?	
	Does the candidate set up a secure Wi-Fi connection using the network module in MicroPython?	
	Does the candidate write a script that reads sensor data and sends it to the cloud using the appropriate API key?	
	Does the candidate implement errors in handling failed HTTP requests when transmitting data?	
	Does the candidate monitor the REPL output for successful transmission logs and sensor readings?	
	Does the candidate verify that data is correctly displayed on the cloud platform in real-time?	
Troubleshoot basic network connectivity and data transmission issues	Does the candidate read and interpret serial monitor output to identify connection status messages?	
	Does the candidate use sta_if.ifconfig() to check IP and network configuration?	
	Does the candidate implement and test Wi-Fi connection scripts with proper handling of connection failures and timeouts?	
	Does the candidate log errors or use print statements to diagnose HTTP request failures?	
	Does the candidate check HTTP response codes (e.g., 200, 403, 404, 500) and interpret them correctly?	
	Does the candidate verify data reception on the cloud platform and confirm whether it is stored, rejected, or delayed?	

	Does the candidate manage API rate limits and test network connectivity to ensure reliable data transmission?	

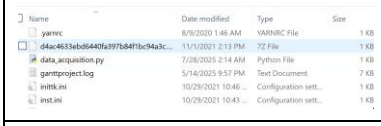
5.2.2 Summative assessment

Trainee name:	**Date:**		
Assessor name:	**Date:**		
The candidate is considered competent if he demonstrates a best practice performance in all the following criteria for the assessment related to the competence unit 4 - Establishing network connectivity using Python for cloud integration			
Criteria for assessment	**Qualifier**	**YES**	**NO (*)**
Configure network settings for IoT devices	The candidate should be able to, in 20 minutes or less, configure the network settings of an IoT device by updating the SSID and password in the script or configuration file, deploying the changes to the device, and verifying that the device successfully connects to the specified Wi-Fi network with a valid IP address.	☐	☐
IoT cloud integration	The candidate should be able to, in 60 minutes or less, successfully connect an IoT device to a cloud platform by writing a Python script that configures Wi-Fi setting	☐	☐
Hardware and Network Configuration	The candidate should be able to, in 30 minutes or less, correctly connect the sensor hardware to the microcontroller and establish a stable Wi-Fi connection	☐	☐
Secure Data Transmission and Cloud Integration	The candidate should be able to, in 30 minutes or less, write, deploy, and run a Python script that reads sensor data and securely transmits it to the cloud platform, handling errors and verifying data receipt.	☐	☐
Troubleshoot basic network connectivity and data transmission issues	The candidate should be able to, in 30 minutes or less, identify and resolve basic network connectivity and data transmission issues by debugging Python scripts, verifying Wi-Fi and cloud platform settings, and ensuring successful sensor data upload and confirmation on the cloud dashboard.	☐	☐
Criteria (*)	**Evidence of non-compliance if any (**)**		
Assessor signature:	**Date:**		
Trainee signature:	**Date:**		

6 Competence Unit 5: Performing end-of-work activities

6.1 Training standards – Unit 5

6.1.1 Training guidelines

<table>
<tr>
<td>Ability / Learning outcome:
Conduct cleanup and backup
Document source code
Store hardware tools
Report according to task records
Training site:
Indoor room of approximately 4 x 5m equipped with:
Wide window
Fire extinguisher
Many Electrical switches and sockets for the PCs
Fixed wooden high-chair furniture suitable for electronics lab activities
Desks and benches where we can place PCs, tools, and materials for practical work
Closet and drawers for organizing and storing electronic components and lab materials</td>
<td>Required Tools:
PC & Monitor or Laptop with USB port
Windows OS installed
Python 3.8+
IDE: (Thonny/VS Code/uPyCraft)
USB Drivers: CP210x / CH340 Drivers
WiFi network credentials
Internet connection
Administrative privileges to install drivers/software
Anti-Static Mat ESD Safe
Storage drawer cabinet for components
Basic tool kit (screwdrivers, pliers, wire cutters)</td>
<td>Required Materials and Consumables:
ESP8266 Development Board (e.g., NodeMCU, Wemos D1 Mini)
Micro USB data cable (ensure it is a data cable, not just charging)
DHT22/DHT11 sensor module
Soil Moisture sensor module
Ambient light sensor module
18650 Li-Ion rechargeable battery + holder
LEDs
Resistors
Breadboard
Jumper wires (female-male , male-male)
breadboard Power Supply Module 3.3V 5V</td>
</tr>
<tr><td colspan="3">Conduct cleanup and backup</td></tr>
<tr>
<td colspan="2">  </td>
<td>Cleanup on the development PC
Open the folder where scripts are saved.
Check for .pyc, .log, .temp, .old, or .zip files.
Remove (delete) or archive unnecessary files.</td>
</tr>
<tr>
<td colspan="2"></td>
<td>Create a local backup folder
Create a folder named IoT_Backup_<date>.
Copy essential .py files and configurations into it.
Add a README.txt to describe the content and version.</td>
</tr>
<tr>
<td colspan="2">  </td>
<td>Log the cleanup and backup on your memory stick
Fill out a cleanup/backup report with:
Date & time
Files removed
Files backed up
Backup location
Comments on any issues</td>
</tr>
<tr><td colspan="3">Document source code</td></tr>
</table>

<table>
<tr><td>

```
def connect_wifi():
    """
    Connects to a predefined Wi-Fi network using ESP8266 module.
    Returns:
        bool: True if connection succeeds, False otherwise.
    """
    pass
```

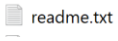
</td><td>

Demonstrate Inline and Block Comments and Python comment syntax:

- Use Python's triple-quote docstrings:

```
def connect_wifi():
    """
    Connects to a predefined Wi-Fi network using ESP8266 module.
    Returns:
        bool: True if connection succeeds, False otherwise.
    """
    pass
```

</td></tr>
<tr><td>



</td><td>

Create and save a Basic README File

- Create and save README:
  - Project name
  - Description and purpose
  - Setup instructions
  - How to run
  - Author and contact

</td></tr>
<tr><td></td><td>

Organize documentation files

- Store all project guides, manuals, and README files there.
- Use comments and docstrings in Python scripts to explain code functionality.
- Include version numbers or dates in filenames for scripts and documents to track updates.

</td></tr>
<tr><td colspan="2">

Store hardware tools

</td></tr>
<tr><td>

</td><td>

Cleanup before storage

- Wipe down tools with appropriate cleaning material
- Remove any residues (e.g., solder, dust, oils)
- Check for visible damage
- Report or tag defective tools for repair or disposal
- Sort tools into predefined categories
- Use dividers or foam inserts for protection
- Ensure storage of jumper wires, sensors and components in anti-static bags or compartments
- Return tools to their designated labeled storage bins
- Ensure labels are clear and match inventory system (if available)
- Place ESD-sensitive devices in anti-static containers
- Disconnect all batteries or power sources from active components

</td></tr>
<tr><td colspan="2">

Report according to task records

</td></tr>
<tr><td>

</td><td>

Document task activities

- Maintain reporting for traceability, accountability, maintenance, and project continuity.
- Record different types of task information: maintenance logs, configuration reports, error logs, and test results.
- Include all key components in a task report: date/time, team members involved, task description/objectives, methods/procedures, tools/software/equipment used, issues and resolutions, outcomes/results, and follow-up actions.

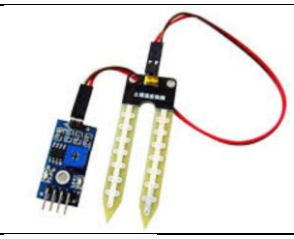
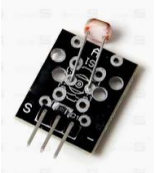
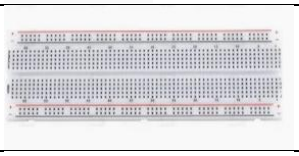
</td></tr>
</table>

	• Review and analyze task records (real or simulated) to extract key data, ensuring clarity, completeness, and technical accuracy. • Complete reporting templates (manual and digital) with concise, objective, and technically correct information using proper terminology.

6.1.2 Tools and materials

Tools	Picture	Quantity[5]	Comment
PC & Monitor & keyboard & mouse (Windows OS installed) Or Laptop with USB port Python 3.8+, Thonny IDE , and USB Drivers: CP210x / CH340 Drivers Administrative privileges to install drivers/software WiFi network credentials		1 per station	
Internet connection		1 per station	
IoT device catalogs and specifications sheets MicroPython syntax reference sheet		1 per station	
Storage drawer cabinet for components		1 per station	
Basic tool kit (screwdrivers, pliers, wire cutters)		1 per station	
Anti-Static Mat ESD Safe		1 per station	

[5] The quantity expressed for workshop means that the item can be used for different modules in the same workshop, this imply that two modules that are using the item could not be delivered at the same time. Warning! before purchasing the item it should be verified if it already exists in the workshop to avoid unnecessary duplications. Also, the list should be reviewed and verified by an experienced professional. Also, the list should be reviewed and verified by an experienced professional.

Tools	Picture	Quantity[5]	Comment
ESP8266 Development Board (e.g., NodeMCU, Wemos D1 Mini)		1	
Micro USB data cable		1	(ensure it is a data cable, not just charging)
DHT22/DHT11 sensor module		1	
Soil Moisture sensor module		1	
Light sensor module		1	
LED		5	
Resistors 470 ohm, 1kohm and 10 kohm		10 each	
Breadboard		1	
Breadboard Power Supply Module 3.3V 5V		1	
Jumper wires (female-male and male- male)		1 pack of each	
18650 Li-Ion rechargeable battery		2	
18650 Li-Ion rechargeable battery holder with wire		2	

Tools	Picture	Quantity[5]	Comment
			

6.2 Assessment standards – Unit 5

6.2.1 Formative assessment

Trainee Name:	**Date:**	
Ability / learning outcome	**Criteria for competence unit 5 - Performing end-of-work activities**	**Yes / No**
Conduct cleanup and backup	Does the trainee identify unnecessary files (.pyc, .log, .temp, .old, .zip) in the development PC folder?	
	Does the trainee delete or archive unnecessary files correctly?	
	Does the trainee create a local backup folder with the correct naming convention (IoT_Backup_<date>)?	
	Does the trainee copy all essential .py files and configuration files into the backup folder?	
	Does the trainee create a README.txt describing the content and version of the backup?	
	Does the trainee complete a cleanup/backup report including date/time, files removed, files backed up, backup location, and comments on issues?	
Document source code	Does the trainee use inline and block comments in Python scripts appropriately?	
	Does the trainee implement Python docstrings for functions to describe purpose, inputs, outputs, and behavior?	
	Does the trainee create and save a basic README file including project name, description/purpose, setup instructions, execution instructions, and author contact?	
	Does the trainee organize all documentation files in the designated folder with proper filenames, version numbers, and dates?	
Store hardware tools	Does the trainee clean tools before storage, removing residues and checking for visible damage?	
	Does the trainee report or tag defective tools for repair or disposal?	
	Does the trainee sort tools into predefined categories and use protective dividers or foam inserts?	
	Does the trainee store jumper wires, sensors, and components in anti-static bags or compartments?	
	Does the trainee return tools to labeled storage bins with clear labels matching the inventory system?	
	Does the trainee ensure ESD-sensitive devices are stored in anti-static containers?	
	Does the trainee disconnect all batteries or power sources from active components before storage?	
Report according to task records	Does the trainee maintain reporting for traceability, accountability, maintenance, and project continuity?	
	Does the trainee record different types of task information, including maintenance logs, configuration reports, error logs, and test results?	
	Does the trainee include all key components in a task report: date/time, team members, task objectives, methods/procedures, tools/software/equipment, issues/resolutions, outcomes/results, and follow-up actions?	
	Does the trainee review and analyze task records to extract key data with clarity, completeness, and technical accuracy?	
	Does the trainee complete reporting templates (manual and digital) with concise, objective, and technically correct information using proper terminology?	

6.2.2 Summative assessment

Trainee name:	**Date:**		
Assessor name:	**Date:**		
The candidate is considered competent if he demonstrates a best practice performance in all the following criteria for the assessment related to the competence unit 5 - Performing end-of-work activities			
Criteria for assessment	**Qualifier**	**YES**	**NO (*)**
Conduct cleanup and backup	The candidate should be able to, in 30 minutes or less, identify and remove unnecessary files, organize essential project components, and create a properly named backup folder that includes all critical scripts, configuration files, and documentation. The backup should be successfully saved in a secure location and verified for completeness.	☐	☐
Document source code	The candidate should be able to apply inline comments, write consistent function-level docstrings, and create a concise README that explains the script's purpose, usage, dependencies, and expected inputs/outputs, ensuring the code is understandable and maintainable by another Python developer within 30 minutes.	☐	☐
Store hardware tools	The candidate should be able to clean, organize, and store all hardware tools within 15 minutes, ensuring damaged or faulty tools are identified and reported, each tool is placed in its designated location, ESD-sensitive items are protected, the storage area is tidy and complete, and any anomalies are documented.	☐	☐
Report according to task records	The candidate should be able to, in 30 minutes or less, compile and submit a technical report based on a completed task, using the appropriate format and ensuring that all critical information is extracted accurately from the task record.	☐	☐
Criteria (*)	**Evidence of non-compliance if any (**)**		
Assessor signature:	**Date:**		
Trainee signature:	**Date:**		