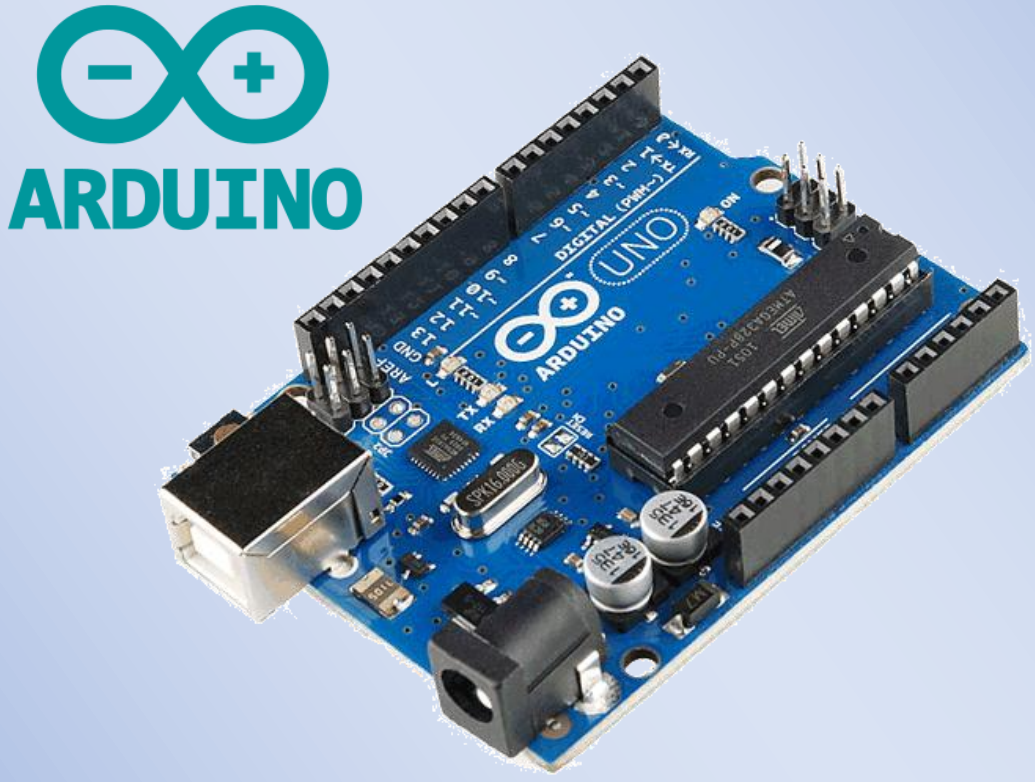
الأردوينو من البداية وحتى الاحتراف

(مستوى المبتدئ)



د.م. حسام الوفائي

مقدمة

يتناول كتاب الأردوينو من البداية وحتى الاحتراف – مستوى المبتدئ – ما يلي:

1-البنية الالكترونية للوحات الأردوينو ومزايا وخواص أشهر لوحات الأردوينو.

2-بيئة التطوير المتكاملة للأردوينو Arduino IDE التي من خلالها يتم برمجة لوحة الأردوينو.

3-مجموعة من التطبيقات العملية:

- التحكم بالثنائيات الضوئية LEDs.
- التحكم بالسبع قطع ضوئي 7-Segment.
- التحكم بمصفوفة ثنائيات ضوئية LED matrix.
- التحكم بشاشة الإظهار الكريستالية السائلة LCD.
- التحكم بلوحة المفاتيح keypad.
- قراءة إشارة الحساسات التشابهية.
- التحكم بالمحركات (المستمرة، الخطوية، السيرفو).
- تشغيل الأجهزة التي تعمل بجهد 220V.
- نقل البيانات وفق بروتوكول الاتصال التسلسلي UART.

لإبداء الملاحظات والمساعدة يرجى التواصل على البريد الالكتروني:

hussam.wafai@hotmail.com

حمص 2018/8/1

د.م. حسام الوفائي

الفهرس

الفصل الأول

لمحة عامة عن لوحة الأردوينو Arduino Board Overview

1-1-مقدمة

لوحة الأردوينو هي عبارة عن لوحة الكترونية تم تصميمها لتساعدك على بناء وإنشاء الدارات والمشاريع الالكترونية المختلفة بكل سهولة. تتسم هذه اللوحة بكونها قابلة للبرمجة programmable board، وهذا يعني أنه يتم برمجتها من قبل المستخدم لتؤدي الغرض المطلوب منها. لهذا كلما أتقنت برمجة لوحة الأردوينو كلما استطعت تنفيذ دارات احترافية أكثر. لغة البرمجة المستخدمة هي Arduino C، وهي لغة بسيطة، متوفرة بشكل مجاني، ومفتوحة المصدر open source بحيث أنه يمكنك الاطلاع على الشيفرة البرمجية للبرامج وتعديلها وتطويرها. في البداية لنلقي نظرة سريعة على قلب لوحة الأردوينو وهو المتحكم الصغري microcontroller، بعد ذلك سنبين في هذا الفصل البنية الالكترونية لأحد أشهر اللوحات المستخدمة وهي لوحة Arduino UNO، وأخيراً سنعرض مزايا بقية اللوحات.

1-2-المتحكم الصغري

المتحكم الصغري عبارة عن حاسوب صغير على شكل دارة متكاملة Integrated Circuit (IC)، يتكون من العناصر المدمجة التالية كما هو موضح في الشكل (1-1):

1-ذاكرة برنامج program memory: يخزن في هذه الذاكرة الشيفرة أو التعليمات البرمجية والتي تم كتابتها من قبل المبرمج. تحدد هذه التعليمات الآلية التي سيعمل من خلالها المتحكم الصغري، والنتائج التي سيتم الحصول عليها. على سبيل المثال قد تكون هذه التعليمات قراءة إشارة حساس حرارة، وتفحص فيما إذا زادت عن قيمة محددة، وتشغيل وإطفاء أجهزة، وهكذا. ذاكرة البرنامج تكون من نوع ROM أو EEPROM أو Flash، بحيث عند عدم تطبيق تغذية كهربائية على المتحكم لن تفقد الذاكرة بياناتها (أي برنامج عمل المتحكم). يمكن لذواكر EEPROM أو Flash حذف البرنامج وتحميل برنامج آخر.

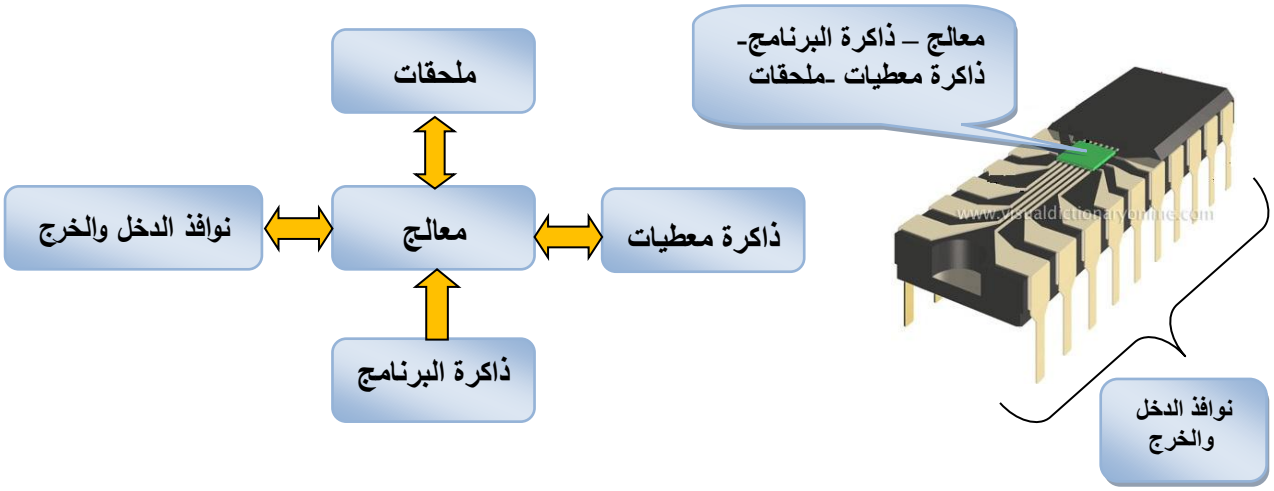
2-معالج processor: عبارة عن دارة الكترونية تعمل على تنفيذ التعليمات البرمجية المخزنة في ذاكرة البرنامج. فمثلاً عندما يحضر المعالج تعليمة قراءة إشارة حساس الحرارة فإنه يفعل الدارة التي يمكن من خلالها قراءة هذه الإشارة، ويعمل على مقارنة الإشارة الناتجة مع القيمة التي حددها المصمم في برنامجه، ليفعّل استناداً لذلك نافذة خرج محددة تكون متصلة مع جهاز ما بحيث يتم تشغيله أو إيقافه عن العمل.

3-ذاكرة معطيات من نوع RAM لتخزين بيانات مختلفة بشكل مؤقت. هذه الذاكرة تفقد بياناتها مع انقطاع التغذية الكهربائية عن المتحكم الصغري. يمكن على سبيل المثال تخزين قيمة درجة الحرارة

فيها الناتجة عن الحساس، وغير ذلك. قد يحتوي المتحكم الصغري أيضاً على ذاكرة معطيات أخرى من نوع EEPROM بحيث إذا قطعت التغذية الكهربائية عن المتحكم تبقى محتفظة ببياناتها، كتخزين كلمة سر على سبيل المثال.

4-نوافذ دخل وخرج: تسمح هذه النوافذ بأن يتصل المتحكم الصغري مع الدارات الالكترونية المحيطة به الخارجية، وهي التي تبدو ظاهرة في المتحكم ويتم التعامل معها. من الممكن على سبيل المثال وصل حساس درجة الحرارة معها، أو ثنائيات ضوئية، ومفاتيح الكترونية وغير ذلك.

5-ملحقات أخرى: بعض المتحكمات الصغرية تحتوي أيضاً ضمن بنيتها الداخلية على وحدات الكترونية إضافية: مثل المبدل التمثيلي الرقمي Analog Digital Converter (ADC)، مؤقتات Timers، وحدات اتصال تسلسلية I2C،UART، SPI، CAN، Ethernet ، وغير ذلك.

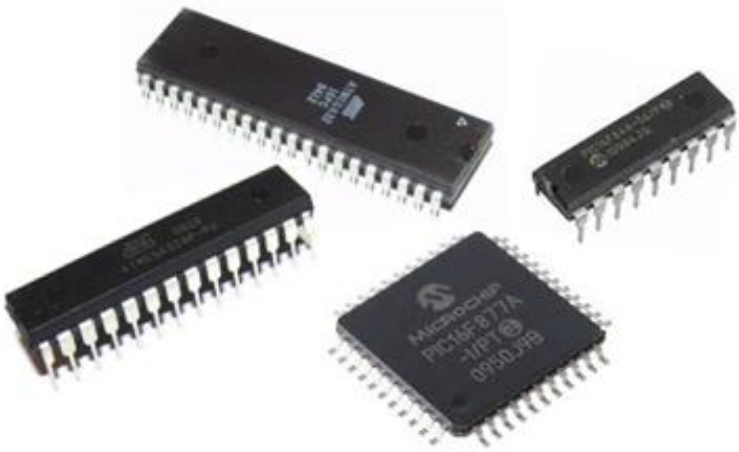


الشكل (1-1): مخطط صندوقي لبنية المتحكم الصغري.

تختلف المتحكمات الصغرية فيما بينها بسعة ذاكرة البرنامج، وذاكرة RAM، وبنية المعالج وسرعته، وبعدد نوافذ الدخل والخرج، وبالملحقات الداخلية، وتبعاً لذلك سيختلف سعرها، ويتم اختيار المتحكم المناسب حسب التطبيق العملي الذي يتم تنفيذه.

يوجد العديد من الشركات المصنعة للمتحكمات الصغرية أشهرها Microchip التي تنتج متحكمات تعرف بـ PIC، وشركة Atmel التي تنتج متحكمات تعرف بـ AVR. يوضح الشكل (1-2) نماذج مختلفة للمتحكمات الصغرية.

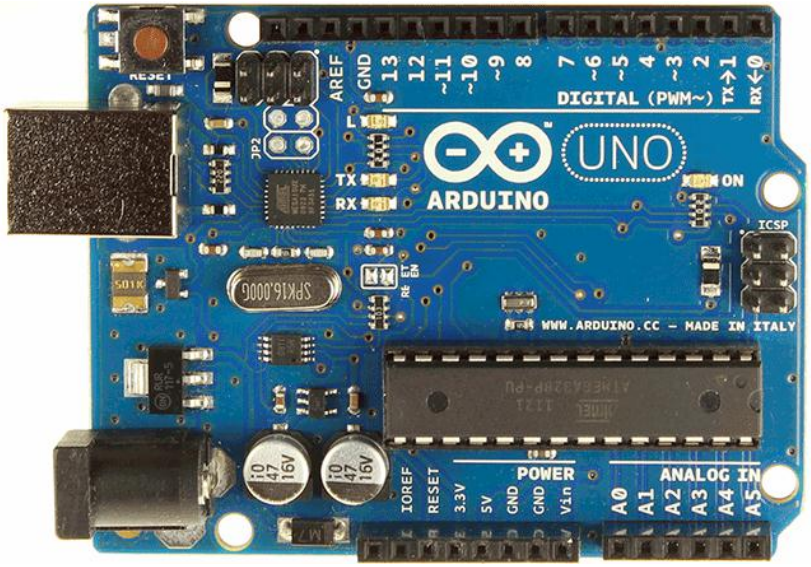
اللغة الأساسية التي يتم بها برمجة المتحكم الصغري هي لغة التجميع assembly، وهي عبارة عن مجموعة من التعليمات التي يفهمها المعالج ويستطيع فك تشفيرها وتنفيذها، ولكل معالج مجموعة التعليمات الخاصة به. لغة التجميع معقدة نسبياً ولا يتم اللجوء إليها غالباً، وإنما تستخدم لغات عالية المستوى كلغة C مثلاً، ويستخدم برنامج مترجم compiler يحول لغة C المفهومة إلى لغة التجميع.



الشكل (1-2): نماذج مختلفة للمتحكمات الصغرية.

1-3-لوحة الأردوينو أونو Arduino UNO

تعتبر لوحة الأردوينو أونو Arduino UNO أكثر اللوحات شهرةً واستخداماً، وتعد الأنسب للبدء بتعلم الأردوينو. يوجد لهذه اللوحة في الوقت الحاضر ثلاثة إصدارات معدلة: اللوحة الأولى هي اللوحة الأصلية Uno R1، اللوحة الثانية تم فيها إدخال بعض التعديلات على اللوحة الأصلية وتعرف بـ UNO R2، لاحقاً تم إجراء بعض التعديلات على اللوحة الثانية وتم الحصول على اللوحة المطورة الثالثة UNO R3. يوجد بعض الاختلاف البسيطة فيما بينها سنذكرها فيما بعد. يبين الشكل (1-3) لوحة الأردوينو أونو المطورة الثالثة Arduino UNO R3 التي تم في هذه الفقرة إيضاح بنيتها الالكترونية الداخلية بشكل مبسط بالإضافة إلى منافذها الخارجية.



الشكل (1-3): لوحة الأردوينو أونو المطورة الثالثة Arduino UNO R3.

1-3-1-المتحكم الصغري AVR ATmega328 الرئيسي

أهم عنصر في لوحات الأردوينو هو المتحكم الصغري الرئيسي، فهو الذي سيحدد خواصها ومزاياها الالكترونية. عندما يتم برمجة اللوحة يتم برمجة هذا المتحكم تحديداً. مع تطبيق التغذية

الكهربائية على اللوحة ينفذ المتحكم البرنامج المخزن فيه لتعمل اللوحة وفق المطلوب منها. المتحكم المستخدم في لوحة الأردوينو أونو هو AVR ATmega328 الموضح في الشكل (1-4).

الشكل (1-4): المتحكم الصغري ATmega328 الرئيسي في لوحة الأردوينو أونو.

لا نحتاج لتعلم هذا المتحكم بالتفصيل، وسنكتفي بذكر أهم خواص هذا المتحكم في الجدول (1-1).

الجدول (1-1): مزايا المتحكم ATmega328.

الخاصية	القيمة
نوع المعالج	8-bit AVR
تردد العمل الأعظمي	20 MHz
الأداء	يستطيع المتحكم تنفيذ مليون تعليمة في الثانية (MIPS) عند تردد عمل 1MHz.
سعة ذاكرة البرنامج	32 KB
سعة ذاكرة SRAM	2 KB
سعة ذاكرة EEPROM	1 KB
عدد الأرجل	28 pin
الملحقات الداخلية	مبدل تمثيلي رقمي ADC بدقة 10 bit، وعدد قنواته 6. ثلاث مؤقتات / عدادات. وحدة الاتصال التسلسلي USART. وحدة الاتصال التسلسلي I2C. وحدة الاتصال التسلسلي SPI. مقاطعات داخلية وخارجية.

يتصل المتحكم ATmega328 في لوحة الأردوينو أونو بكريستالة ترددها 16 MHz، لذلك فإن عدد التعليمات المنفذة في الثانية 16 مليون تعليمة.

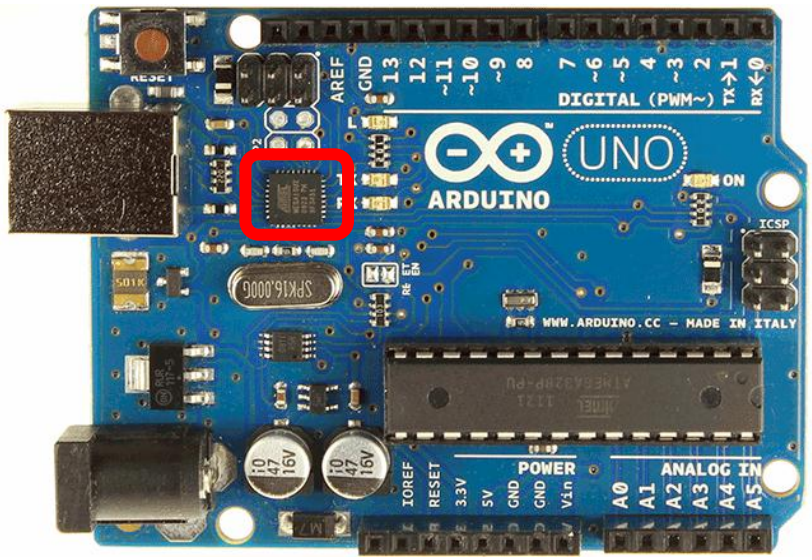
1-3-2-المتحكم الصغري ATmega16U2 الثانوي

لا يمكن للمتحكم الرئيسي ATmega328 أن يتبادل البيانات مع الحاسب عبر منفذ USB، لذلك تم إضافة المتحكم ATmega16U2 إلى لوحة الأردوينو أونو، الذي يوجد ضمن بنيته الداخلية وحدة اتصال تسلسلية USB. يبين الشكل(1-5) مكان توضع هذا المتحكم ضمن لوحة الأردوينو أونو. البيانات التي يتم تبادلها ما بين المتحكم الرئيسي ATmega328 والحاسب قد تكون عملية نقل للشيفرة البرمجية المكتوبة في الحاسب عن طريق برنامج Arduino cc إلى المتحكم، أو نقل معلومات أثناء تشغيل اللوحة مثل إرسال قيمة درجة الحرارة من اللوحة إلى الحاسب، وإرسال أوامر من الحاسب إلى اللوحة وغير ذلك، وهذا ما سيتم دراسته لاحقاً.

لا يتصل متحكم ATmega328 بشكل مباشر مع منفذ USB كما ذكرنا، وإنما يتواصل مع متحكم ATmega16U2 بشكل تسلسلي وفق بروتوكول UART، وبدوره يتصل ATmega16U2 مع الحاسب عبر منفذ USB. بنفس الطريقة يتصل الحاسب مع ATmega328 عن طريق ATmega16U2. يمكن اعتبار متحكم ATmega16U2 بمثابة مبدل USB إلى Serial (UART) والعكس، ويظهر كمنفذ COM افتراضي virtual على الحاسب.

يتصل متحكم ATmega16U2 مع خط إعادة إقلاع المتحكم الرئيسي ATmega328 لأنه عندما يراد برمجة المتحكم الرئيسي لابد من إعادة تشغيله، وهذا ما يقوم به هذا المتحكم. كذلك يتصل متحكم ATmega16U2 مع كريستالة ترددها 16MHz تقع إلى الأسفل منه في الشكل (1-5).

لابد من الإشارة إلى أنه في لوحتي الأردوينو أونو للنوعين original وR2 يستخدم المتحكم ATmega8U2 بدلاً من المتحكم ATmega16U2.

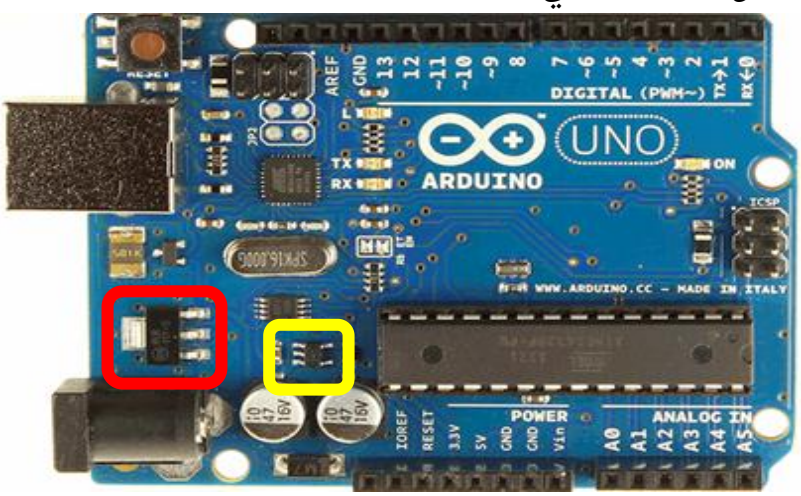


الشكل (1-5): المتحكم الصغري ATmega16U2 .

1-3-3-منظم جهد 5V، ومنظم جهد 3.3V

تعمل لوحة الأردوينو أونو بجهد 5V+، لذلك تحتاج إلى منظم جهد 5V+ عندما يتم تغذيتها كهربائياً من منبع خارجي. منظم جهد 5V+ المستخدم هو 1117ST50T3G. تيار الخرج الأعظمي

لهذا المنظم يزيد عن 1 A. توفر اللوحة أيضاً جهداً مقداره 3.3 V يمكن أن يستخدم مع دارات أخرى. منظم جهد 3.3V+ المستخدم هو LP2985-33BVR. تيار الخرج الأعظمي لهذا المنظم يصل إلى 150 mA. يوضح الشكل (1-6) منظمي الجهد 1117ST50T3G و LP2985-33BVR.

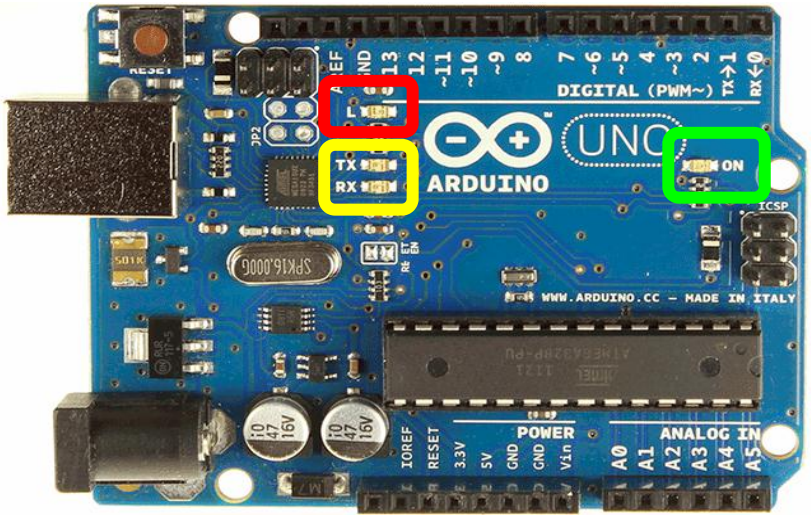


الشكل (1-6): المنظم 1117ST50T3G المحاط بالمربع الأحمر، ومنظم الجهد LP2985-33BVR المحاط بالمربع الأصفر.

1-3-4-مجموعة من الثنائيات الضوئية

تحتوي لوحة الأردوينو أونو على أربعة ثنائيات ضوئية كما هو موضح في الشكل (1-7).

- ثنائي ضوئي ON: يشير إلى أن اللوحة تم تطبيق جهد كهربائي 5V+ عليها وهي جاهزة للعمل. لون هذا الثنائي أخضر.
- ثنائي ضوئي L: هذا الثنائي متصل مع المخرج الرقمي 13. يضيء عند تطبيق 1 منطقي (5V+) على هذا المخرج. لون هذا الثنائي أصفر.
- ثنائيان ضوئيان RX، TX: يومض هذان الثنائيان عندما تنتقل البيانات ما بين متحكم ATmega16U2 ومنفذ USB إلى الحاسب فقط. لون هذين الثنائيين أصفر.



الشكل (1-7): الثنائي الضوئي ON محاط بالمربع الأخضر، الثنائي ضوئي L محاط بالمربع الأحمر، الثنائيان الضوئيان RX،TX المحاطان بالمربع الأصفر.

1-3-5-منفذ USB

يوضح الشكل (1-8) منفذ USB للوحة الأردوينو أونو. لهذا المنفذ عدة استخدامات:

- برمجة اللوحة عن طريق الحاسب.
- تبادل البيانات ما بين لوحة الأردوينو (المتحكم ATmega328) ومنفذ USB للحاسب.
- تغذية اللوحة بجهد 5V+ عند وصل هذا المنفذ مع الحاسب.

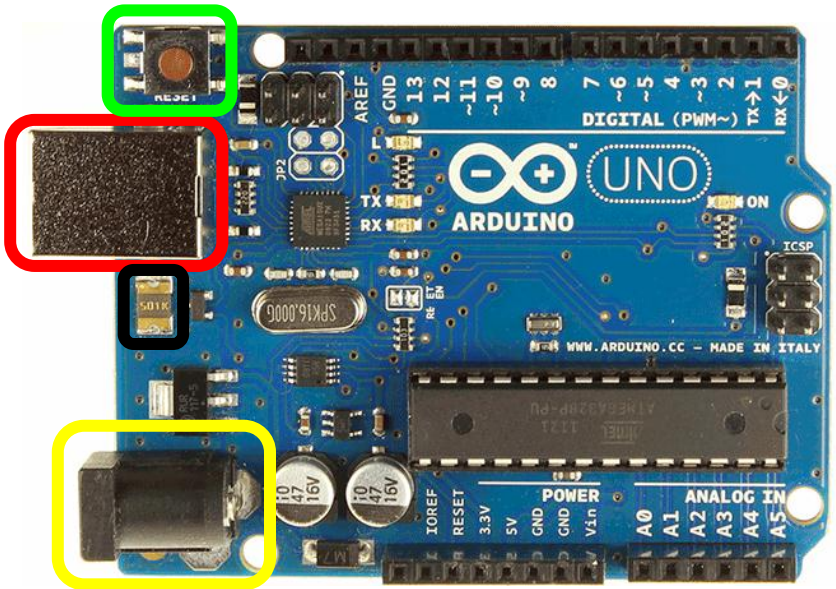
بجانب هذا المنفذ يوجد فيوز Fuse لحماية منفذ USB للحاسب من القصر والتيار الزائد على الرغم من أن معظم الحواسيب لها دارة حماية داخلية.. يقطع الفيوز الاتصال بشكل آلي عندما يزيد التيار عن 500 mA.

1-3-6-مقبس الطاقة

يمكن تأمين مصدر طاقة بديل لمنفذ USB عن طريق وصل محول AC-to-DC إلى مقبس الطاقة power jack الموضح في الشكل (1-8). قطر المقبس 2.1 mm، وفي مركزه يطبق الطرف الموجب للمحول. مجال جهد المحول الموصى به يمتد من 7V وحتى 12V.

1-3-7-زر إعادة التشغيل

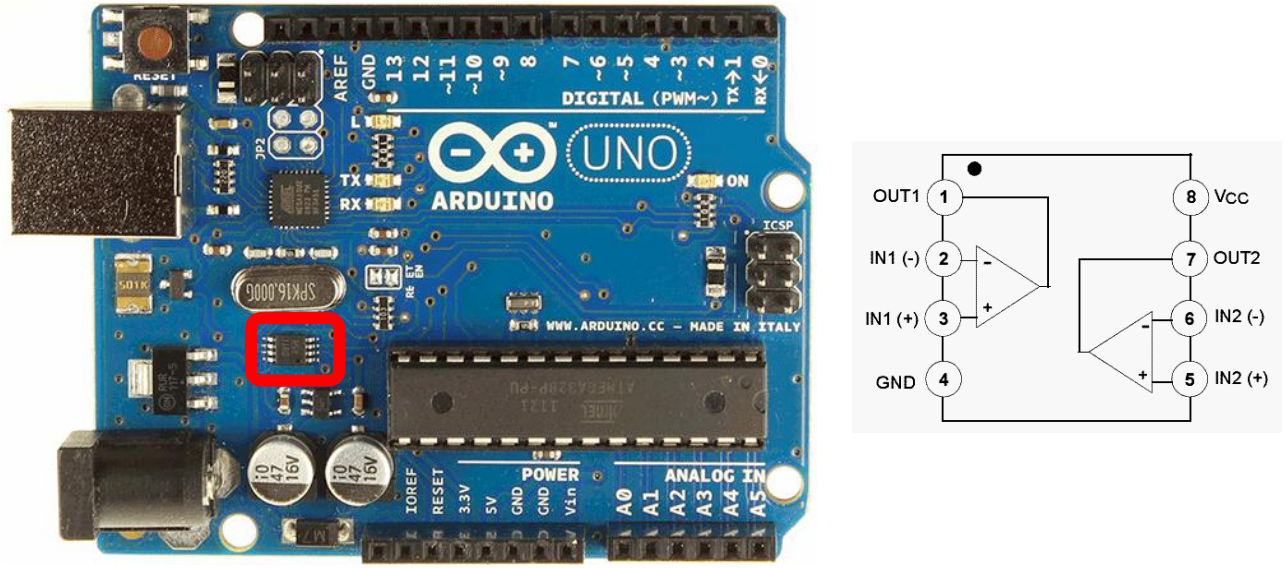
يوضح الشكل (1-8) زر إعادة إقلاع لوحة الأردوينو أونو لبدء تنفيذ الأوامر من البداية من جديد.



الشكل (1-8): منفذ USB المحاط باللون الأحمر، وبجانبه فيوز الحماية محاط باللون الأسود، مقبس الطاقة الخارجية المحاط باللون الأصفر، وزر إعادة التشغيل المحاط باللون الأخضر.

1-3-8-دارة متكاملة LM358

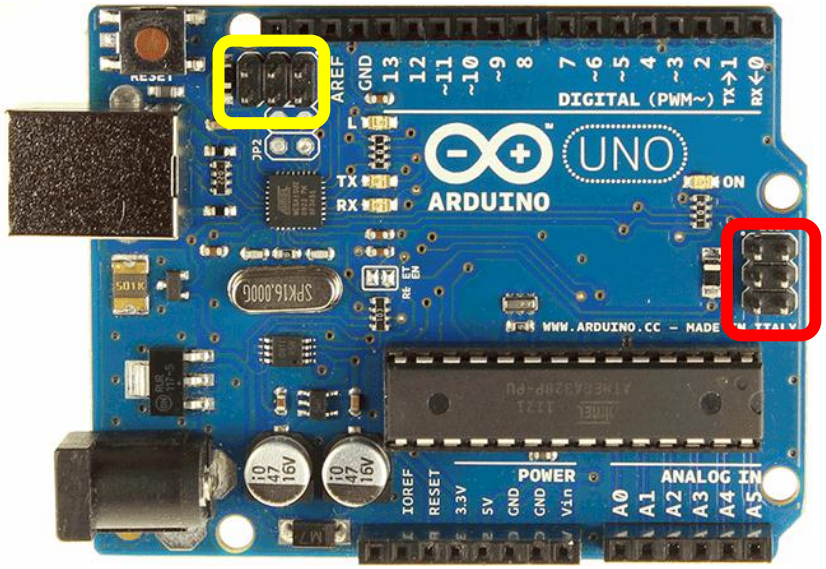
تتضمن لوحة الأونو أيضاً دارة متكاملة LM358 كما هو موضح في الشكل (1-9). هذه الدارة عبارة عن مكبري عمليات. يستخدم أحدهما كمقارن جهدي لاختيار تغذية اللوحة من منفذ USB أو من منفذ Vin، ويستخدم الآخر كعازل buffer ما بين المخرج 13 والثنائي الضوئي L.



الشكل (1-9): الدارة المتكاملة LM358 في لوحة الأردوينو أونو.

1-3-9-منفذ برمجة تسلسلي In-Circuit Serial Programming (ICSP)

ذكرنا سابقاً أنه من الممكن برمجة المتحكم الرئيسي ATmega328 عن طريق منفذ USB. يوجد طريقة أخرى يتم فيها برمجته من خلال منفذ يعرف بـ ICSP الموضح في الشكل (1-10). كذلك يوجد للمتحكم ATmega16U2 منفذ ICSP تتم برمجته من خلاله. تتم عمليه البرمجة ICSP من خلال استخدام مبرمجة خارجية أو لوحة أردوينو أخرى يتم وصلها معه.



الشكل (1-10): منفذ برمجة ICSP للمتحكم ATmega328 محاط باللون الأحمر، و منفذ برمجة ICSP للمتحكم ATmega16U2 محاط باللون الأصفر.

10-3-1-منافذ الاستطاعة

يوضح الشكل (1-11) خمسة منافذ استطاعة في لوحة الأردوينو أونو وهي مرتبة من اليمين إلى اليسار على الشكل التالي:

المنفذ Vin: يمكن من خلاله تطبيق مصدر تغذية خارجي للوحة (بطارية مثلاً) بدلاً من منفذ USB ومقبس الطاقة. يتراوح الجهد المطبق ما بين 7V و 12V تقريباً.

منفذان GND على التتالي: أرضي لوحة الأردوينو. يمكن الاستفادة منهما عند وصل اللوحة مع دارات أخرى.

منفذ 5V+: تعطي لوحة الأردوينو من خلال هذا المخرج جهداً مقداره 5V+، بالتالي من الممكن استخدامه لتغذية الدارات الخارجية المربوطة مع اللوحة.

منفذ 3.3V: تعطي لوحة الأردوينو أيضاً من خلال هذا المخرج جهداً مقداره 3.3V+، بالتالي من الممكن استخدامه لتغذية الدارات الخارجية المربوطة مع اللوحة. أعظم تيار يقدمه هذا المنفذ 50mA.

يوجد إلى الأيسر من منافذ الاستطاعة ثلاثة منافذ أخرى مرتبة على الشكل التالي:

منفذ RESET: من خلال هذا المنفذ يتم إعادة تشغيل لوحة الأردوينو، وذلك من خلال تطبيق 0V (إشارة منخفضة) عليه. يمكن وصل مفتاح لحظي button، طرف منه يتصل مع هذا المنفذ، والطرف آخر يتصل مع الأرضي، عند الضغط على هذا المفتاح يتم تطبيق إشارة 0V على هذا المنفذ، ويعاد تشغيل اللوحة. يعمل المفتاح في هذه الحالة بشكل مشابه لزر إعادة التشغيل الموجود على اللوحة. يمكن الاستفادة من هذا المنفذ عندما يتم وصل لوحة الأردوينو مع لوحة تعرف بالغطاء shield (لوحة خاصة يتم تركيبها فوق الأردوينو الأصلي لتوسيع عمله) مانعةً الوصول لزر إعادة التشغيل على لوحة الأردوينو، ليكون البديل هذا المنفذ.

منفذ IOREF: يقدم هذا المنفذ الجهد المرجعي الذي يعمل فيه المتحكم الصغري. يستخدم هذا المنفذ من قبل ألواح الأغطية shield لاختيار مصدر الطاقة المناسب أو تمكين محول جهد على المخارج للعمل مع 5V أو 3.3V.

منفذ غير مستخدم حالياً: ومن الممكن أن يستخدم لاحقاً للتطوير.

الشكل (1-11): منافذ الاستطاعة من اليمين إلى اليسار Vin، GND، GND، 5V، 3.3V. يضاف إليها منفذ RESET، IOREF، وفي الآخر منفذ غير مستخدم حالياً.

1-3-11-منافذ دخل تشابهية

تسمح هذه المنافذ بإدخال إشارات تشابهية، والتي تنتج عادة عن الحساسات التشابهية. يتم تحويل الإشارة التشابهية إلى إشارة رقمية باستخدام مبدل ADC في متحكم ATmega328. دقة تمييز المبدل 10bit، والجهد المرجعي افتراضياً 5V، ويمكن تغييره عن طريق المنفذ AREF. عدد هذه المنافذ التشابهية 6 هي A0، A1، A2، A3، A4، A5 كما هو موضح في الشكل (1-12).

الشكل (1-12): المداخل التشابهية الستة في لوحة الأردوينو أونو.

1-3-12-منافذ دخل وخرج رقمية

يوضح الشكل (1-13) منافذ الدخل والخرج الرقمية مرتبة من اليمين إلى اليسار بالأرقام من 0 وحتى 13. والتي لها عدة وظائف كما يلي:

يمكن لهذه الأرجل كلها أن تعمل كمخارج رقمية (أي تخرج إشارة 0V أو إشارة 5V) تبعاً للشيفرة البرمجية. أو أن تعمل كمداخل رقمية (أي إدخال إشارة 0V أو إشارة 5V). بالتالي يتم على هذه النوافذ وصل عناصر ودارت الكترونية أخرى مثل الثنائيات الضوئية، شاشة الإظهار LCD التي تستقبل بتات رقمية، أو لوحة مفاتيح، أو محرك، وغير ذلك كما سنرى. لكل رجل مقاومة سحب داخلية internal pull-up (غير مفعلة افتراضياً).

المنافذ 3،5، 6، 9، 10، 11 : يمكن لكل منها أن تولد إشارة تعديل عرض النبضة Pulse Width Modulation (PWM). يرمز إليها في اللوحة على شكل ~. إشارة PWM هي إشارة مربعة دورية يتم التحكم بعرض النبضة المرتفعة (1 منطقي)، وسيتم شرحها لاحقاً.

المنفذان 0،1 يعملان كواجهة اتصال للبروتوكول التسلسلي UART الذي سيتم دراسته فيما بعد. المنفذ 1 للإرسال، والمنفذ 0 للاستقبال. يتصل هذان المنفذان في نفس الوقت مع منافذ متوافقة للمتحكم ATmega16U2 بحيث يمكن إرسال البيانات التسلسلية إلى الحاسب عبر منفذ USB.

المنفذان 2،3 : تستخدم كمقاطعات خارجية

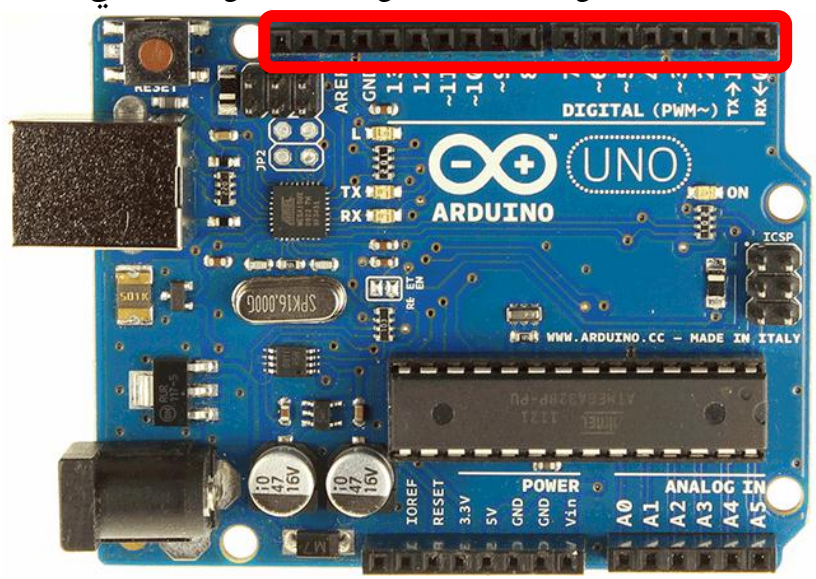
المنافذ 10،11، 12، 13 : تعمل كواجهة اتصال للبروتوكول التسلسلي SPI.

يضاف لهذه المنافذ إلى اليسار منها في الشكل (1-13) المنافذ التالية:

منفذ GND: أرضي لوحة الأردوينو.

منفذ AREF : الجهد المرجعي للمداخل التشابهية. سيتم إيضاح المقصود منه لاحقاً.

المنفذان الأخيران SDA،SCL : يعملان كواجهة اتصال للبروتوكول التسلسلي I2C.



الشكل (1-13): المداخل والمخارج الرقمية المرقمة بالترتيب من اليمين إلى اليسار. يضاف إلى الأيسر منها منافذ GND ، AREF، SDA، SCL.

يمكن تلخيص مزايا وخواص لوحة الأردوينو أونو بالجدول (1-2).

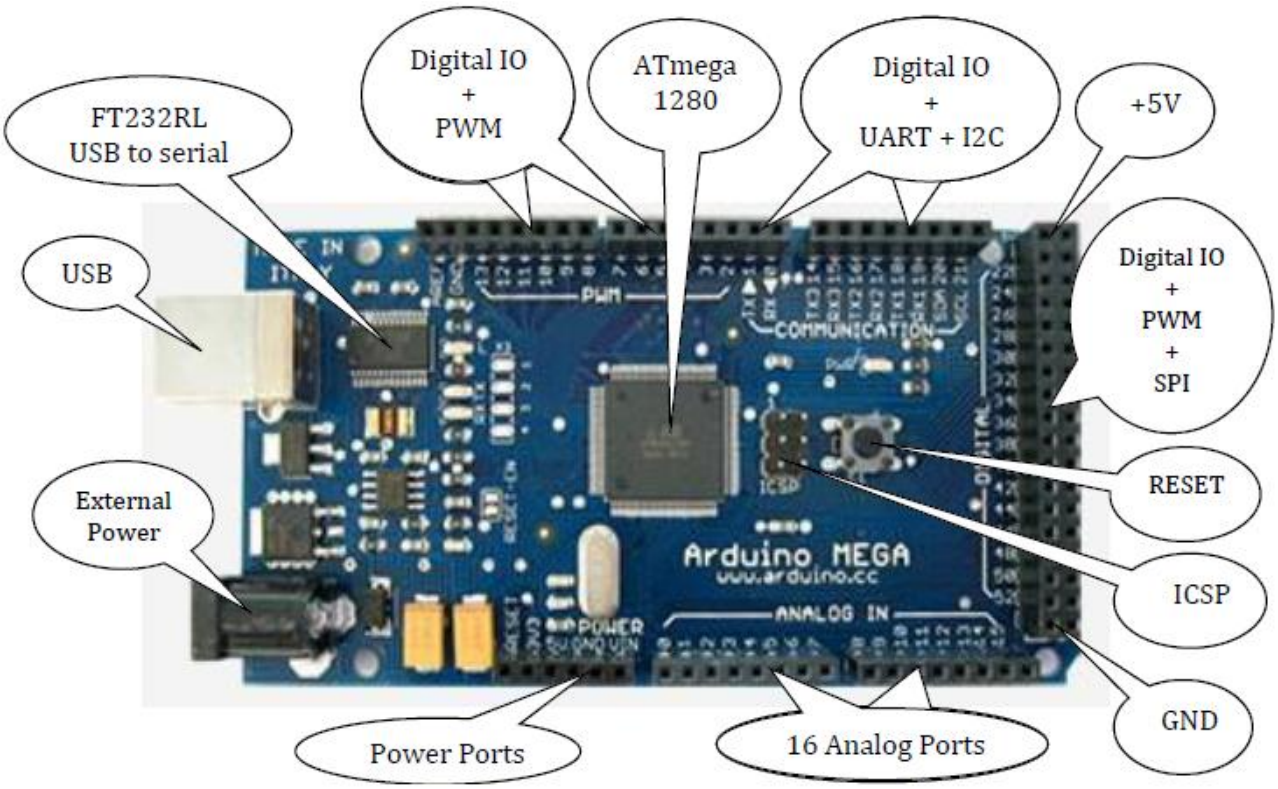
الجدول (1-2): مزايا وخواص لوحة الأردوينو أونو

المتحكم	ATmega328
سعة ذاكرة البرنامج	32KB (يستخدم 0.5 KB لمحمل الإقلاع bootloader)
سعة ذاكرة SRAM	2KB
سعة ذاكرة EEPROM	1KB
سرعة المعالج	16MHz
جهد عمل اللوحة	+5V
جهد الدخل عبر مقبس الطاقة أو مدخل Vin الموصى به	7-12V
عدد منافذ الدخل أو الخرج الرقمية	14
التيار الأعظمي للمنفذ الرقمي (دخل أو خرج) الموصى به	20 mA
عدد أرجل PWM	6
عدد أرجل الدخل التشابهية	6
عدد أرجل الخرج التشابهية	لا يوجد
دقة تمييز المبدل ADC	10 bit
الثنائي الضوئي L المدمج	مع المنفذ 13
بروتوكولات الاتصال التسلسلية التي تدعمها	SPI، I2C، UART
طرق البرمجة	عن طريق منفذ USB أو مبرمجة خارجية (ICSP)
الطول	68.6 mm
العرض	53.4 mm
الوزن	25 gr

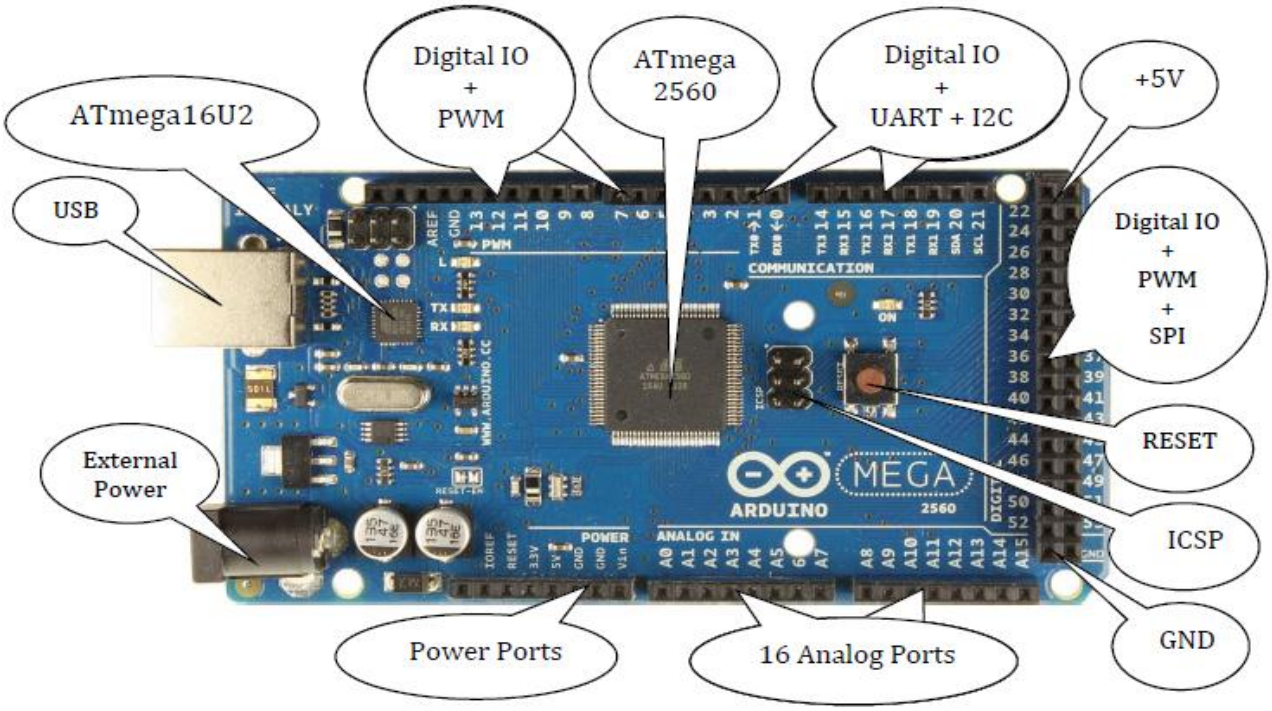
1- 4-لوحات Arduino MEGA - Arduino MEGA 2560 - Arduino MEGA ADK

تعتبر لوحات الأردوينو ميغا MEGA من اللوحات ذات الحجم الكبير، حيث توفر منافذاً رقمية يصل عددها إلى 54 أي ما يعادل 4 أضعاف لوحة الأردوينو أونو تقريباً، ومنافذ دخل تشابهية عددها 16 منفذاً. تتشابه اللوحات الثلاثة في شكلها كما هو موضح في (1-14)، (1-15)، (1-16)، وتختلف في بعض الأمور. يستخدم الأردوينو Arduino MEGA متحكماً صغرياً ATmega1280 لذلك فإن سعة ذاكرة البرنامج 128KB، بينما تستخدم اللوحتان Arduino MEGA 2560 و Arduino MEGA ADK متحكماً صغرياً ATmega2560 لذلك نجد أن سعة ذاكرة البرنامج لهما 256KB. تمتاز لوحة Arduino MEGA ADK أيضاً بوجود دارة متكاملة MAX3421E التي توفر واجهة مستضيف USB

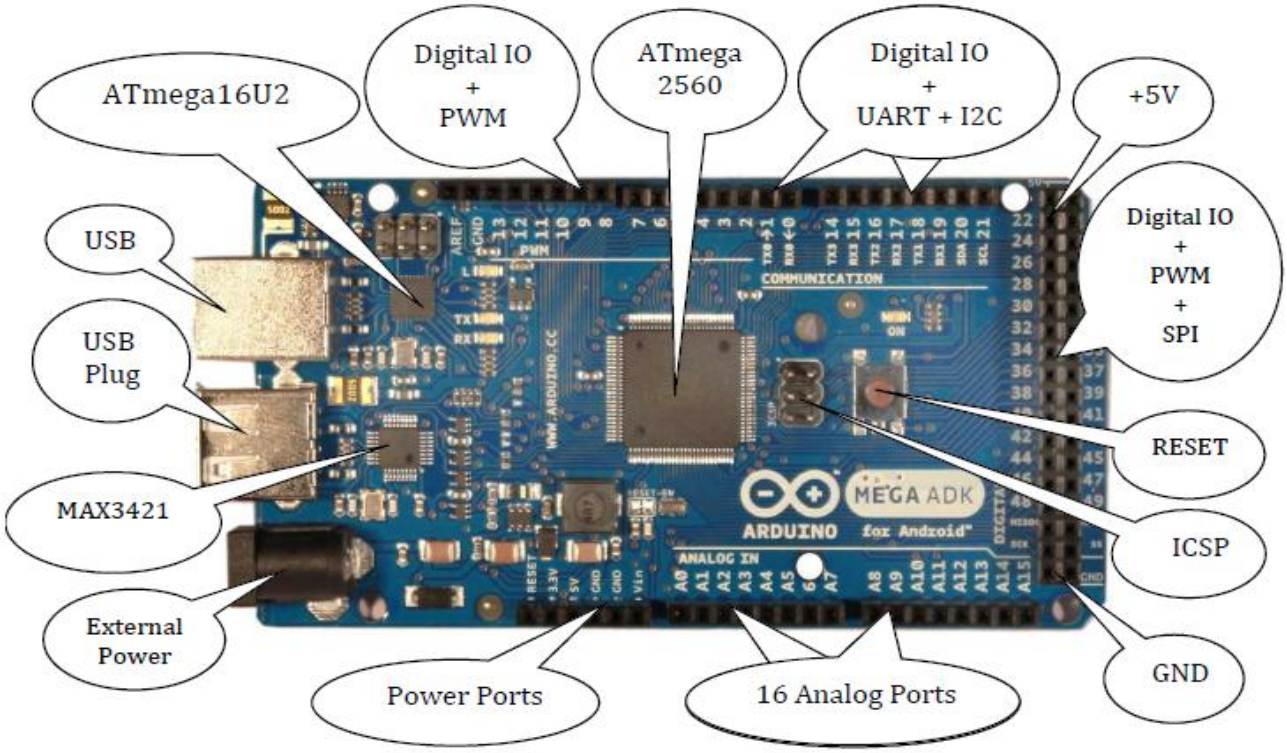
(USB Host)، مما يسمح لهذه اللوحة بالاتصال والتفاعل مع أي جهاز له منفذ USB مثل الهواتف الخلوية الداعمة لنظام الأندرويد android، والكاميرات، ومتحكمات ألعاب مثل PS3 وWiimote، بالإضافة إلى الفارة ولوحة المفاتيح. يقارن الجدول (1-3) ما بين خواص ومزايا لوحات الأردوينو الثلاثة.



الشكل (1-14): لوحة Arduino MEGA التي تحتوي على المتحكم ATmega1280، وبجانبه منفذ البرمجة ICSP وزر إعادة التشغيل Reset، والدارة المتكاملة FT232RL التي تعمل كمبدل USB-serial، و16 منفذ دخل تشابهي، و32 منفذ رقمي التي يعمل بعضها كوحدات اتصال UART أوI2C أو SPI، والبعض الآخر كمولد إشارة PWM.



الشكل (1-15): لوحة Arduino MEGA 2560 التي تحتوي على المتحكم ATmega2560، وبجانبه منفذ ICSP وزر إعادة التشغيل Reset، والمتحكم ATmega16U2، و16 منفذ تشابهي، و32 منفذ رقمي التي يعمل بعضها كوحدات اتصال UART أوI2C أو SPI، والبعض الآخر كمولد إشارة PWM.



الشكل (1-16): لوحة Arduino MEGA ADK التي تحتوي على المتحكم ATmega2560، وبجانبه منفذ ICSP وزر إعادة التشغيل Reset، والمتحكم ATmega16U2 ، و16 منفذ دخل تشابهي، و32 منفذ رقمي التي يعمل بعضها كوحدات اتصال UART أوI2C أو SPI، والبعض الآخر كمولد إشارة PWM، والدارة المتكاملة MAX3421 التي تعمل كمستضيف لمنفذ USB plug.

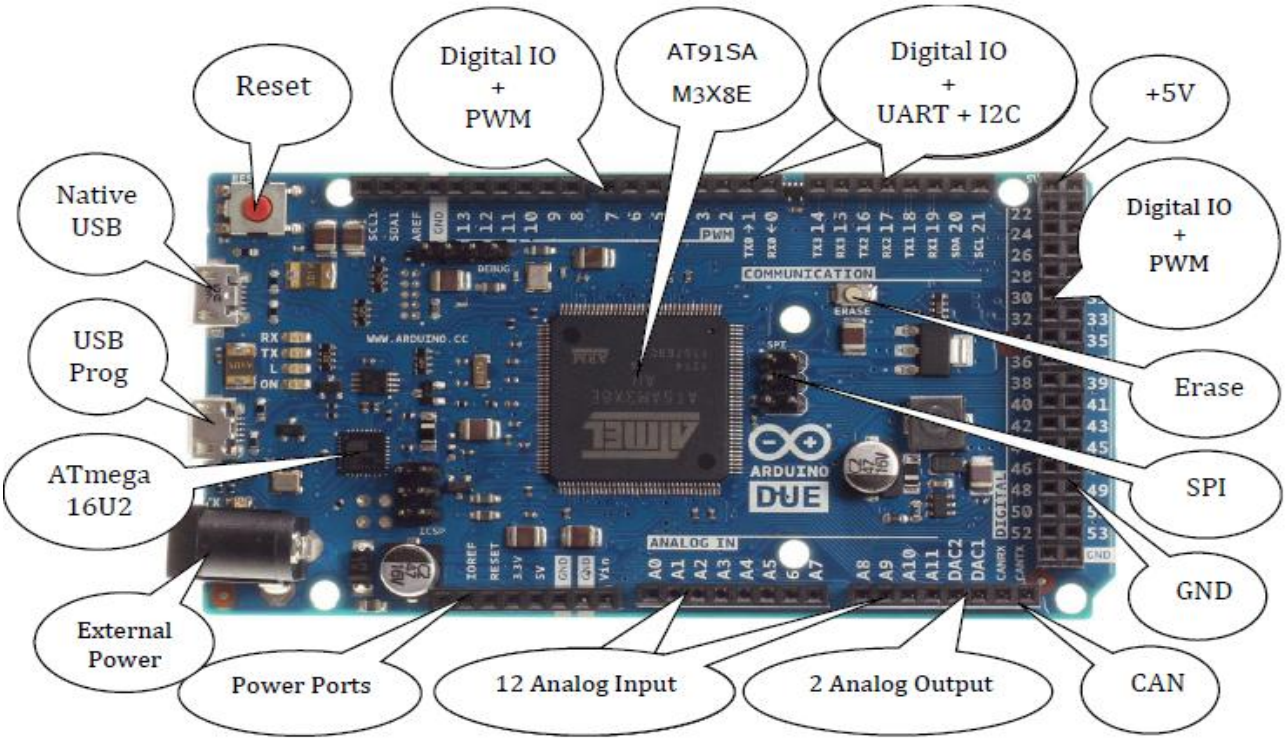
الجدول (1-3): مقارنة ما بين مزايا لوحات الأردوينو **MEGA** و **MEGA 2560** و **MEGA ADK**

	MEGA	**MEGA 2560**	**MEGA ADK**
المتحكم	ATmega1280	ATmega2560	ATmega2560
سعة ذاكرة البرنامج	8KB) 128KB لمحمل الإقلاع)	8KB) 256KB لمحمل الإقلاع)	8KB) 256KB لمحمل الإقلاع)
سعة ذاكرة SRAM	8 KB	8 KB	8 KB
سعة ذاكرة EEPROM	4 KB	4 KB	4 KB
سرعة المعالج	16 MHz	16 MHz	16 MHz
جهد عمل اللوحة	+5 V	+5 V	+5 V
جهد الدخل عبر مقبس الطاقة أو مدخل Vin	7-12V	7-12V	7-12V
عدد المنافذ الرقمية	54	54	54
التيار الأعظمي للمنفذ الرقمي (دخل أو خرج) الموصى به	40 mA) 20 mA كحد أعظمي)	40 mA) 20 mA كحد أعظمي)	40 mA) 20 mA كحد أعظمي)
عدد أرجل PWM	15	15	15
عدد أرجل الدخل التشابهية	16	16	16
عدد أرجل الخرج التشابهية	لا يوجد	لا يوجد	لا يوجد
دقة تمييز المبدل ADC	10 bit	10 bit	10 bit
الثنائي الضوئي L المدمج	مع المنفذ 13	مع المنفذ 13	مع المنفذ 13
بروتوكولات الاتصال التسلسلية التي تدعمها	UART : 4 SPI : 1 I2C : 1	UART : 4 SPI : 1 I2C : 1 USB Host : 1	UART : 4 SPI : 1 I2C : 1
طرق البرمجة	عن طريق منفذ USB أو مبرمجة خارجية (ICSP)	عن طريق منفذ USB أو مبرمجة خارجية (ICSP)	عن طريق منفذ USB أو مبرمجة خارجية (ICSP)
الطول	101.52 mm	101.52 mm	101.52 mm
العرض	53.4 mm	53.3 mm	53.3 mm
الوزن	25 gr	37 gr	36 gr

1-5-لوحة الأردوينو Arduino Due

تشبه لوحة Arduino Due لوحة Arduino MEGA ADK من حيث الشكل، وتختلف عنها في البنية الالكترونية. المتحكم الصغري المستخدم في هذه اللوحة هو AT91SAM3X8E الذي يعتمد على بينة ARM بـ 32 بت، مع سرعة معالج تصل إلى84 MHz. تعمل اللوحة بجهد مقداره 3.3 V. يوجد منفذي خرج تشابهيين، ومنفذين للاتصال وفق بروتوكول CAN، وزر مسح erase لمسح ذاكرة

البرنامج. كذلك يوجد منفذين micro USB أحدهما للبرمجة والاتصال مع الحاسب، والآخر للاتصال مع أجهزة محيطية، ويمكن استخدامه للبرمجة. لا تتم برمجة المتحكم من خلال منفذ ICSP. يبين الشكل (1-17) لوحة Arduino Due، والجدول (1-4) مزايا وخواص هذه اللوحة.



الشكل (1-17): لوحة الأردوينو DUE. منفذ USB Prog للبرمجة والاتصال مع الحاسب، والمنفذ Native USB للاتصال مع الوحدات المحيطية غالباً.

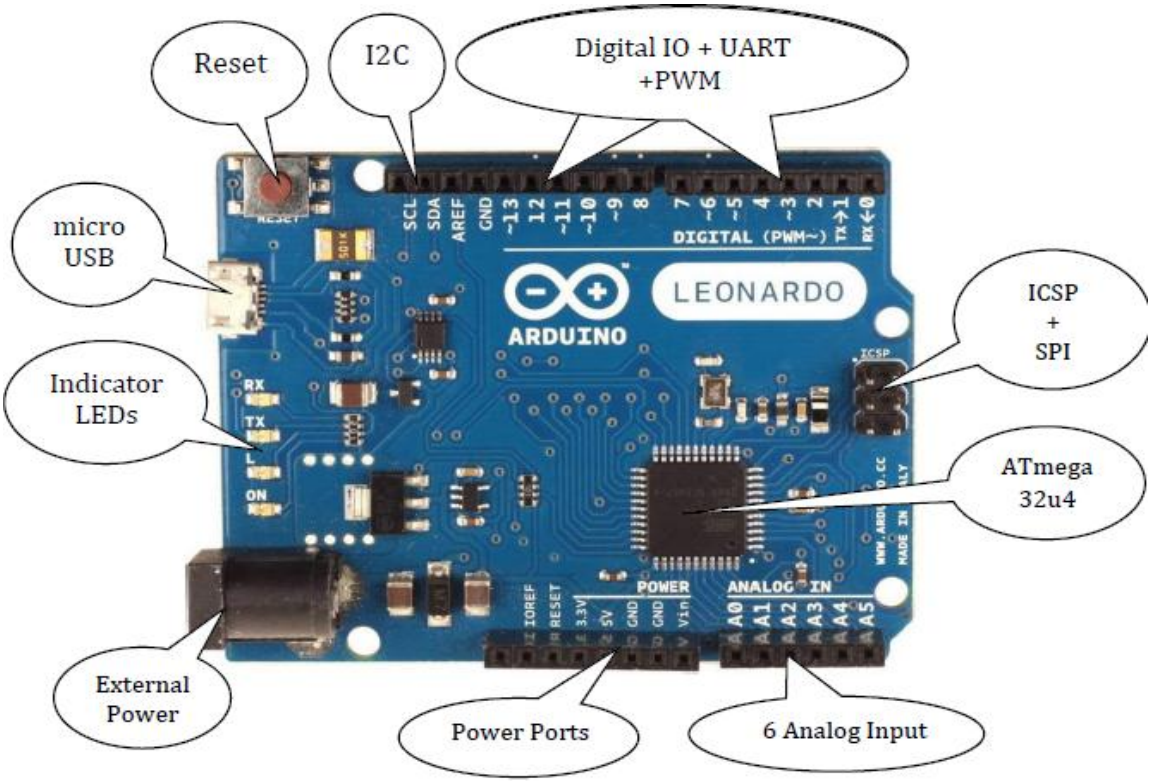
الجدول (1-4): مزايا وخواص لوحة الأردوينو DUE.

المتحكم	AT91SAM3X8E
سعة ذاكرة البرنامج	512KB
سعة ذاكرة SRAM	96KB
سرعة المعالج	84 MHz
جهد عمل اللوحة	+5V
جهد الدخل عبر مقبس الطاقة أو مدخل Vin الموصى به	7-12V
عدد المنافذ الرقمية	54
التيار الأعظمي للمنفذ الرقمي الموصى به	منفذ خرج: 3 mA أو 15 mA. اعتماداً على المنفذ المستخدم. منفذ دخل: 6 mA أو 9 mA. اعتماداً على المنفذ المستخدم.
عدد أرجل PWM	12
عدد أرجل الدخل التشابهية	12
عدد أرجل الخرج التشابهية	2
دقة تمييز المبدل ADC	12 Bit

دقة تمييز المبدل DAC	12 bit
الثنائي الضوئي L المدمج	مع المنفذ 13
بروتوكولات الاتصال التسلسلية	UART : 4 SPI : 1 I2C : 2 USB Host : 1
طرق البرمجة	Programming USB Port. Native USB Port.
أعظم تيار لمنفذ التغذية 3.3 V	800 mA
أعظم تيار لمنفذ التغذية 5 V	800 mA
الطول	101.52 mm
العرض	53.3 mm
الوزن	36 gr

6-1- لوحة الأردوينو ليوناردو Arduino Leonardo

تشبه لوحة الأردوينو ليوناردو لوحة الأونو من حيث الشكل، وتختلف عنها بالمتحكم الصغري الرئيسي المستخدم وهو ATmega32u4 الذي يحتوي على وحدة اتصال USB مدمجة، لذلك لا يحتاج إلى متحكم ثانوي، مما يسمح للحاسب بالاتصال مع الليوناردو كجهاز واجة إنسان Human interface device(HID) كالفارة أو لوحة المفاتيح، بالتالي فهو مناسب للمشاريع التي تتطلب ذلك. يمكن أيضاً للحاسب أن يتصل مع اللوحة كمنفذ COM افتراضي. يوضح الشكل (1-18) لوحة الأردوينو ليوناردو، والجدول (1-5) مزايا اللوحة.



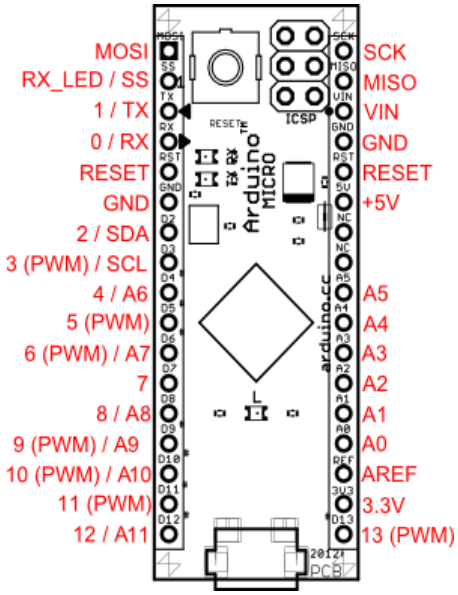
الشكل (1-18): لوحة الأردوينو ليوناردو.

الجدول (1-5): مزايا وخواص لوحة الأردوينو ليوناردو

المتحكم	ATmega32u4
سعة ذاكرة البرنامج	32KB (4 KB لمحمل الإقلاع bootloader)
سعة ذاكرة SRAM	2.5KB
سعة ذاكرة EEPROM	1KB
سرعة المعالج	16MHz
جهد عمل اللوحة	+5V
جهد الدخل عبر مقبس الطاقة أو مدخل Vin الموصى به	7-12V
عدد منافذ الدخل أو الخرج الرقمية	20
التيار الأعظمي للمنفذ الرقمي الموصى به	20 mA
عدد أرجل PWM	7
عدد أرجل الدخل التشابهية	12
عدد أرجل الخرج التشابهية	لا يوجد
دقة تمييز المبدل ADC	10 bit
الثنائي الضوئي L المدمج	مع المنفذ 13
بروتوكولات الاتصال التسلسلية	SPI، I2C، USB،UART
طرق البرمجة	عن طريق منفذ USB أو مبرمجة خارجية (ICSP)
الطول	68.6 mm
العرض	53.4 mm
الوزن	20 gr

1-7-لوحة الأردوينو ميكرو Arduino micro

يعتبر الأردوينو ميكرو لوحة مصغرة للوحة ليوناردو، ولها نفس المزايا والخواص المذكورة في الجدول (1-5) باستثناء أبعاد اللوحة ووزنها. تم تصميمها بحيث يمكن استخدامها بسهولة مع لوحة التجارب breadboard. يوضح الشكل (1-19) لوحة الأردوينو ميكرو، والجدول (1-6) يبين مزايا وخواص اللوحة.



الشكل (1-19): لوحة الأردوينو ميكرو

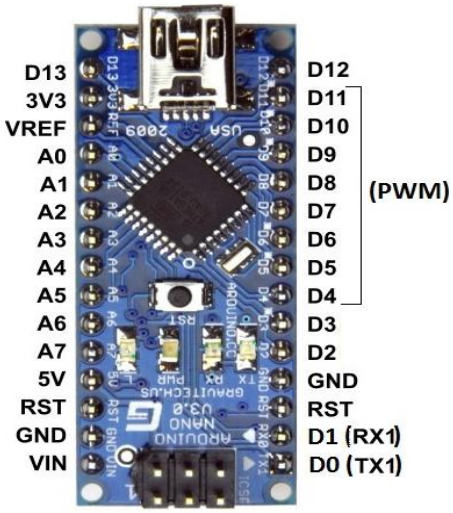
الجدول (1-6): مزايا وخواص لوحة الأردوينو ميكرو

المتحكم	ATmega32u4
سعة ذاكرة البرنامج	32KB (4 KB لمحمل الإقلاع bootloader)
سعة ذاكرة SRAM	2.5KB
سعة ذاكرة EEPROM	1KB
سرعة المعالج	16MHz
جهد عمل اللوحة	+5V
جهد الدخل عبر مقبس الطاقة أو مدخل Vin الموصى به	7-12V
عدد منافذ الدخل أو الخرج الرقمية	20
التيار الأعظمي للمنفذ الرقمي الموصى به	20 mA
عدد أرجل PWM	7
عدد أرجل الدخل التشابهية	12
عدد أرجل الخرج التشابهية	لا يوجد
دقة تمييز المبدل ADC	10 bit
الثنائي الضوئي L المدمج	مع المنفذ 13
بروتوكولات الاتصال التسلسلية	SPI، I2C، USB،UART
طرق البرمجة	عن طريق منفذ USB أو مبرمجة (ICSP)
الطول	48 mm
العرض	18 mm
الوزن	13 gr

8-1-لوحة الأردوينو نانو Arduino Nano

تعتبر لوحة النانو لوحة مصغرة للوحة الأونو ويمكن استخدامها بسهولة مع لوحات التجارب.

يوضح الشكل (1-20) لوحة الأردوينو نانو، والجدول (1-7) يبين مزايا وخواص اللوحة.



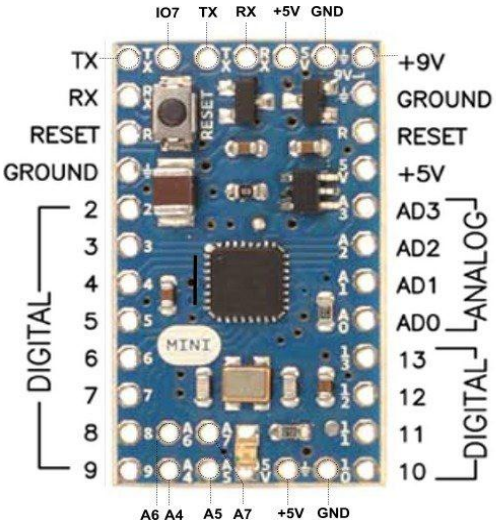
الشكل (1-20): لوحة الأردوينو نانو.

الجدول (1-7): مزايا وخواص لوحة الأردوينو نانو

المتحكم	ATmega328
سعة ذاكرة البرنامج	32KB (2 KB لمحمل الإقلاع bootloader)
سعة ذاكرة SRAM	2KB
سعة ذاكرة EEPROM	1KB
سرعة المعالج	16MHz
جهد عمل اللوحة	+5V
جهد الدخل عبر مدخل Vin الموصى به	7-12V
عدد منافذ الدخل أو الخرج الرقمية	22
التيار الأعظمي للمنفذ الرقمي الموصى به	20 mA
عدد أرجل PWM	6
عدد أرجل الدخل التشابهية	8
عدد أرجل الخرج التشابهية	لا يوجد
دقة تمييز المبدل ADC	10 bit
الثنائي الضوئي L المدمج	مع المنفذ 13
بروتوكولات الاتصال التسلسلية	SPI، I2C،UART
طرق البرمجة	عن طريق منفذ USB أو مبرمجة (ICSP)
الطول	45 mm
العرض	18 mm
الوزن	7 gr

9-1- لوحة الأردوينو Arduino Mini

تشبه هذه اللوحة لوحة النانو السابقة، ولها نفس المزايا باستثناء عدم وجود محول USB-serial في اللوحة، مما يستدعي استخدام ذلك المحول لربط اللوحة مع الحاسب للبرمجة أو تبادل المعلومات. يوضح الشكل (1-21) لوحة الأردوينو Mini، والجدول (1-8) مزايا اللوحة.



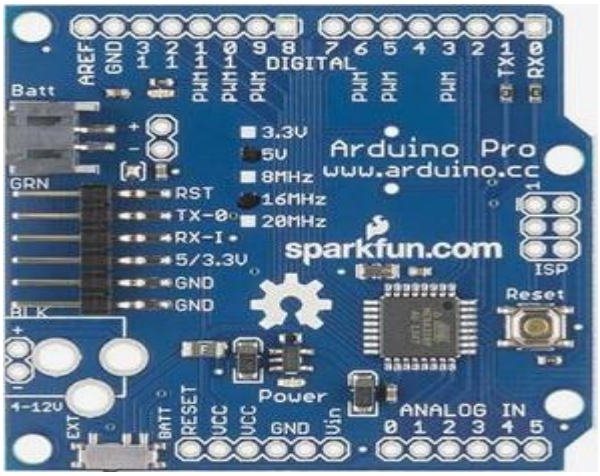
الشكل (1-21): لوحة الأردوينو Mini.

الجدول (1-8): مزايا وخواص لوحة الأردوينو Mini.

المتحكم	ATmega328
سعة ذاكرة البرنامج	32KB (2 KB لمحمل الإقلاع bootloader)
سعة ذاكرة SRAM	2KB
سعة ذاكرة EEPROM	1KB
سرعة المعالج	16 MHz
جهد عمل اللوحة	+5V
جهد الدخل عبر مدخل Vin الموصى به	7-9 V
عدد منافذ الدخل أو الخرج الرقمية	14
التيار الأعظمي للمنفذ الرقمي الموصى به	20 mA
عدد أرجل PWM	6
عدد أرجل الدخل التشابهية	8
عدد أرجل الخرج التشابهية	لا يوجد
دقة تمييز المبدل ADC	10 bit
طرق البرمجة	يمكن عن طريق منفذ USB ولكن تحتاج لمحول USB-Serial أو عن طريق مبرمجة (ICSP)
الطول	30 mm
العرض	18 mm
الوزن	7 gr

10-1 – لوحة Arduino Pro

تم تصميم لوحة Pro ليتم تركيبها بشكل شبه دائم في المشاريع، حيث تأتي المنافذ من دون رؤوس محمولة مسبقاً، مما يسمح من لحمها بأسلاك خارجية، أو أنماط مختلفة من الموصلات. لا يوجد محول USB-serial، مما يستدعي استخدام ذلك المحول لربط اللوحة مع الحاسب للبرمجة أو تبادل المعلومات. يوضح الشكل (1-22) لوحة الأردوينو Pro، والجدول (1-9) مزايا اللوحة.



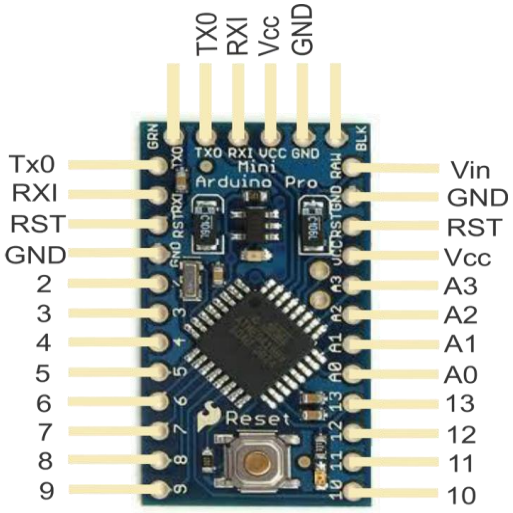
الشكل (1-22): لوحة Arduino Pro.

الجدول (1-9): مزايا وخواص لوحة Arduino Pro

المتحكم	ATmega328
سعة ذاكرة البرنامج	32KB (2 KB لمحمل الإقلاع bootloader)
سعة ذاكرة SRAM	2KB
سعة ذاكرة EEPROM	1KB
سرعة المعالج	8MHz(3.3V version) or 16MHz(5V version)
جهد عمل اللوحة	3.3V(3.3V version) or 5V(5V version)
جهد الدخل عبر مدخل Vin الموصى به	3.3-12V(3.3V version) or 5-12V(5V version)
عدد منافذ الدخل أو الخرج الرقمية	14
التيار الأعظمي للمنفذ الرقمي	20 mA
عدد أرجل PWM	6
عدد أرجل الدخل التشابهية	6
عدد أرجل الخرج التشابهية	لا يوجد
دقة تمييز المبدل ADC	10 bit
الثنائي الضوئي L المدمج	مع المنفذ 13
بروتوكولات الاتصال التسلسلية	SPI، I2C،UART
طرق البرمجة	يمكن عن طريق منفذ USB ولكن تحتاج لمحول USB-Serial أو عن طريق مبرمجة خارجية (ICSP)
الطول، العرض	52 mm , 53.3 mm

11-1 – لوحة Arduino Pro mini

يعتبر الأردوينو Pro mini لوحة مصغرة للوحة Pro، ولها نفس المزايا والخواص المذكورة في الجدول (1-10) باستثناء أبعاد اللوحة ووزنها. تم تصميمها أيضاً ليتم تركيبها بشكل شبه دائم في المشاريع. يوضح الشكل (1-23) لوحة الأردوينو Pro mini، والجدول (1-10) مزايا اللوحة.



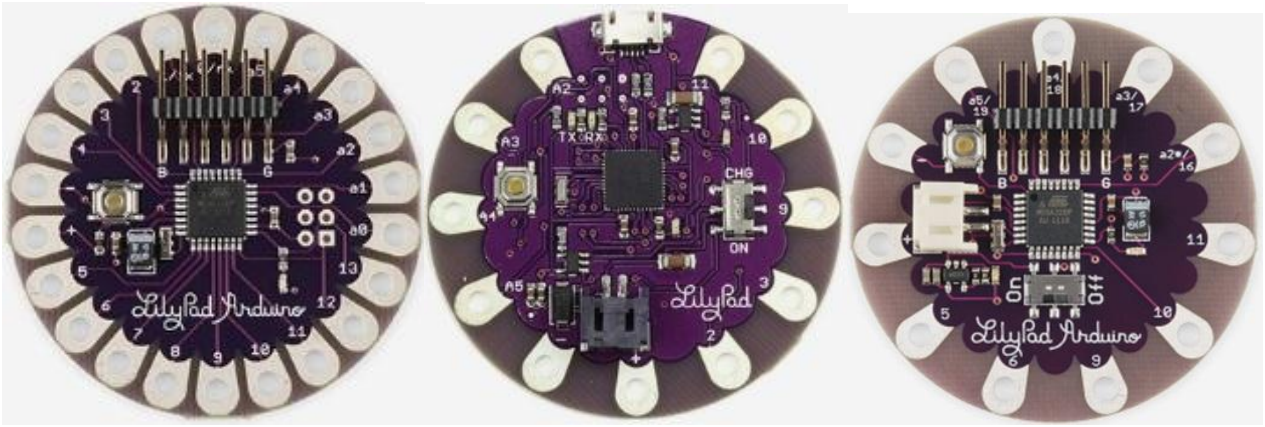
الشكل (1-23): لوحة Arduino Pro min.

الجدول (1-10): مزايا وخواص لوحة Arduino Pro mini.

المتحكم	ATmega328
سعة ذاكرة البرنامج	32KB (2 KB لمحمل الإقلاع bootloader)
سعة ذاكرة SRAM	2KB
سعة ذاكرة EEPROM	1KB
سرعة المعالج	8MHz(3.3V version) or 16MHz(5V version)
جهد عمل اللوحة	3.3V(3.3V version) or 5V(5V version)
جهد الدخل عبر مدخل Vin الموصى به	3.3-12V(3.3V version) or 5-12V(5V version)
عدد منافذ الدخل أو الخرج الرقمية	14
التيار الأعظمي للمنفذ الرقمي الموصى به	20 mA
عدد أرجل PWM	6
عدد أرجل الدخل التشابهية	6
عدد أرجل الخرج التشابهية	لا يوجد
دقة تمييز المبدل ADC	10 bit
الثنائي الضوئي L المدمج	مع المنفذ 13
بروتوكولات الاتصال التسلسلية	SPI، I2C،UART
طرق البرمجة	يمكن عن طريق منفذ USB ولكن تحتاج لمحول USB-Serial أو عن طريق مبرمجة خارجية (ICSP)
الطول، العرض	17.7 mm ، 33 mm

12-1-لوحة أردوينو ليلي باد Arduino Lilypad

تم تصميم لوحات ليلي باد لتعمل مع المنسوجات الالكترونية E-textiles. تقدم نفس وظائف لوحات الأردوينو الأخرى مع مزايا أخرى مثل خفة الوزن، ودائرية الشكل، ولها منافذ خرج كبيرة بهدف تسهيل عملية الاتصال والربط مع الملابس. يوجد عدة إصدارات للوحات ليلي باد كما هو موضح في الشكل (1-24): Lilypad main board - Lilypad USB - Lilypad simple. يقارن الجدول (1-11) مزايا وخواص اللوحات الثلاثة.



(c) (b) (a)

الشكل (1-24): لوحات ليلي باد أردوينو: (a) Lilypad main، (b) Lilypad USB، (c) Lilypad simple.

الجدول (1-11): مقارنة ما بين لوحات Lilypad

	Lilypad main board	Lilypad USB	Lilypad simple
المتحكم	ATmega168 أو ATmega328	ATmega32u4	ATmega328
سعة ذاكرة البرنامج	16KB (يستخدم 2 KB لمحمل الإقلاع)	32KB (يستخدم 4 KB لمحمل الإقلاع)	32KB (يستخدم 2 KB لمحمل الإقلاع)
سعة ذاكرة SRAM	1KB	2.5 KB	2 KB
سعة ذاكرة EEPROM	512 B	1 KB	1 KB
سرعة المعالج	8MHz	8MHz	8MHz
جهد عمل اللوحة	2.7-5.5V	3.3 V	2.7-5.5V
جهد الدخل عبر وصلة USB أو طاقة خارجية	2.7-5.5V	3.8-5 V	2.7-5.5V
عدد منافذ الدخل أو الخرج الرقمية	14	9	9
التيار الأعظمي للمنفذ الرقمي الموصى به	20 mA	mA 20	20 mA
عدد أرجل PWM	6	4	5

عدد أرجل الدخل التشابهية	6	4	4
عدد أرجل الخرج التشابهية	لا يوجد	لا يوجد	لا يوجد
دقة تمييز المبدل ADC	10 bit	10 bit	10 bit
طرق البرمجة	يمكن عن طريق منفذ USB ولكن تحتاج لمحول USB-Serial أو عن طريق مبرمجة خارجية (ICSP)	يمكن عن طريق منفذ USB لأن المتحكم يتضمن وحدة USB مدمجة أو عن طريق مبرمجة (ICSP)	يمكن عن طريق منفذ USB ولكن تحتاج لمحول USB-Serial أو عن طريق مبرمجة خارجية (ICSP)
القطر			50 mm

الفصل الثاني

بيئة التطوير المتكاملة للأردوينو Arduino IDE

2-1-مقدمة

بيئة التطوير المتكاملة للأردوينو Arduino integrated development environment (IDE) هو البرنامج الذي من خلاله يتم كتابة الكود البرمجي للوحة الأردوينو بلغة C، وتحميله إليها. تتسم بيئة التطوير بالبساطة وسهولة التعامل، وتحتوي على كل ما يحتاجه المبرمج لكتابة الشيفرة، وتعمل على أنظمة التشغيل المختلفة: Windows، Mac OS X، Linux .

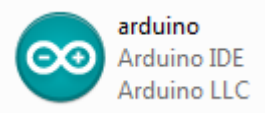
تعرف البرامج المكتوبة باستخدام بيئة التطوير Arduino IDE بالسكتش sketch، ويتم تخزينها في الحاسب بملف يأخذ التوسع ino.

2-2-تحميل بيئة التطوير المتكاملة للأردوينو Arduino IDE

يمكن تحميل Arduino IDE من الموقع الرسمي لمطوري الأردوينو من الرابط التالي:

https://www.arduino.cc/en/Main/Software

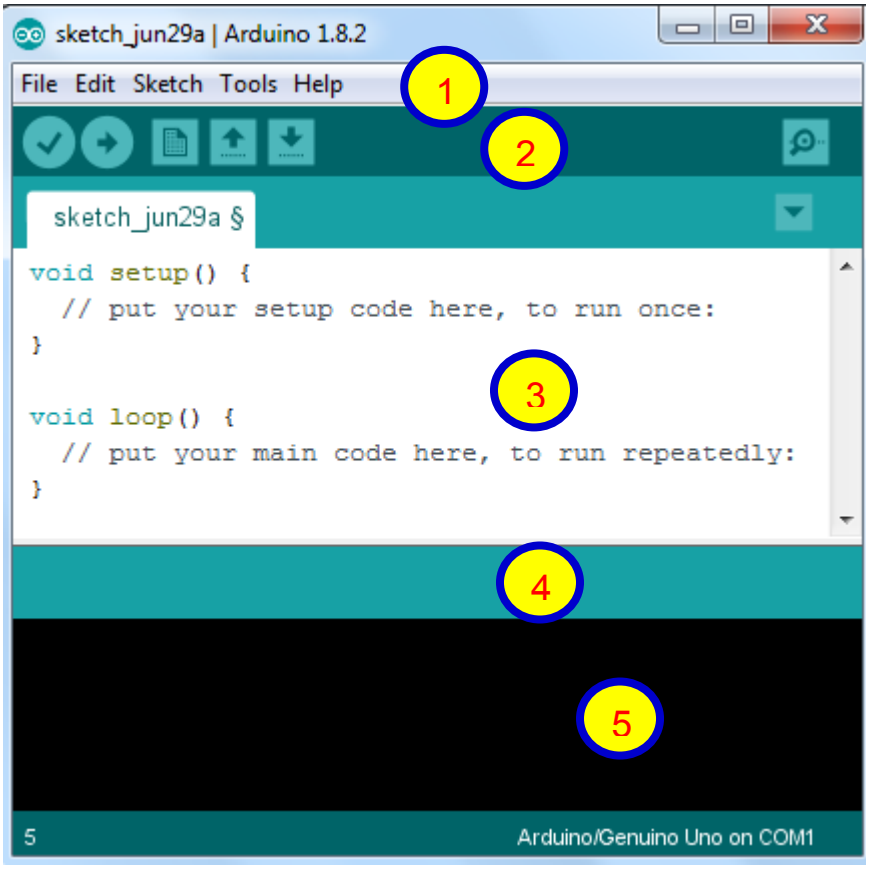
بعد فتح الرابط نختار نظام التشغيل ومن ثم just download ليبدأ التحميل مباشرة ومجاناً. قم بفك الضغط للملف المحمل الذي يحتوي على برنامج بيئة التطوير وتعاريف ومكتبات وأمثلة وغيرها. قم بتشغيل برنامج Arduino.exe لتعمل بيئة التطوير مباشرة من دون الحاجة إلى تنصيب setup.



2-3- بيئة التطوير المتكاملة للأردوينو Arduino IDE

يوضح الشكل (2-1) واجهة بيئة التطوير Arduino IDE التي تتكون من الأجزاء التالية:

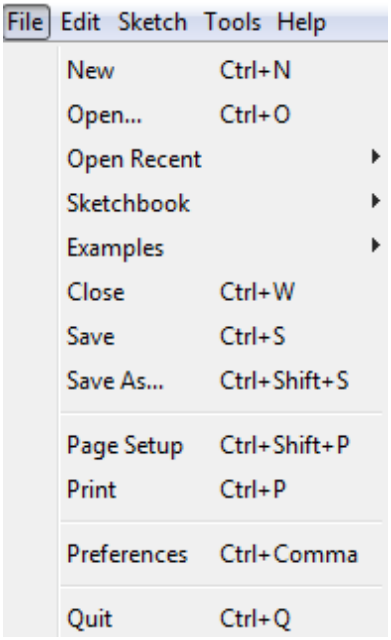
1. شريط أدوات القوائم menu toolbar.
2. شريط أدوات الوظائف العامة common functions toolbar.
3. محرر النص text editor.
4. منطقة الرسالة message area.
5. لوحة مراقبة النص text console.



الشكل (2-1): بيئة التطوير المتكاملة للأردوينو Arduino IDE.

2-1-3-شريط أدوات القوائم

يحتوي هذا الشريط على خمس قوائم هي: File – Edit – Sketch – Tools – Help. في كل قائمة عدد من الوظائف.



تتضمن القائمة File عدداً من الوظائف:

New: إنشاء نموذج جديد للمحرر.

Open: لفتح ملف مخزن.

Open Recent: تقدم قائمة قصيرة لأحداث الملفات البرمجية (sketch) لفتحها.

Examples: تحتوي أمثلة مقدمة من بيئة التطوير.

Close: إغلاق المحرر الذي يتم العمل عليه.

Save: حفظ الملف البرمجي بنفس الاسم.

Save as: حفظ الملف البرمجي باسم مختلف.

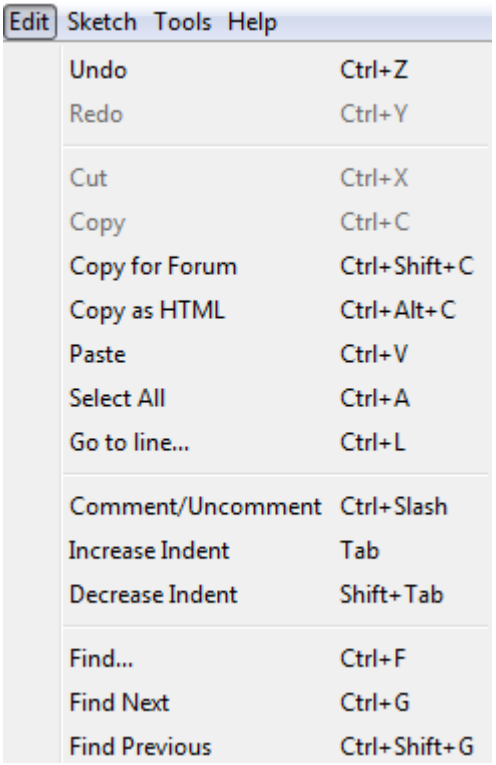
Page setup: إعداد الصفحة للطباعة.

Print: طباعة الملف البرمجي.

Preferences: يتم فتح نافذة التفضيلات لضبط بعض إعدادات بيئة التطوير مثل لغة الواجهة.

Quit: يتم إغلاق كافة نوافذ بيئة التطوير.

تتضمن القائمة Edit عدداً من الوظائف الخاصة بالنسخ، والقص، واللصق، والتراجع، والبحث.

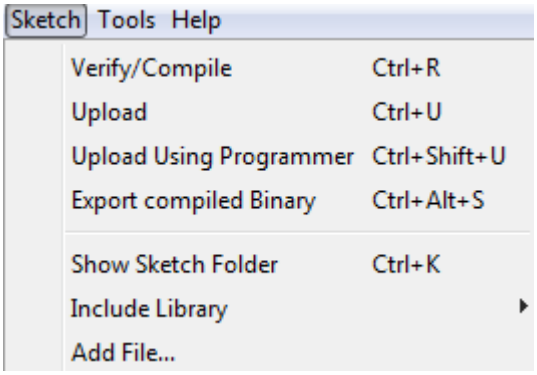


تتضمن القائمة Sketch الوظائف التالية:

Verify/Compile: لفحص النص البرمجي من الأخطاء، مع تقديم معلومات عن مقدار استخدام ذاكرة البرنامج وذاكرة RAM في منطقة مراقبة النص text area.

Upload: يتم تحويل الملف البرمجي إلى ترميز الآلة ومن ثم يتم تحميل الملف الثنائي إلى اللوحة عبر المنفذ الذي تم إعداده. (قبل تحميل الملف للوحة لابد من اختيار اللوحة بشكل صحيح من القائمة Tools>Board، واختيار منفذ COM الصحيح أثناء وصل اللوحة مع الحاسب وذلك من القائمة Tools>Port).

Upload Using Programmer: يستخدم هذا الأمر لنقل الملف البرمجي إلى اللوحة عن طريق مبرمجة خارجية. يتم الكتابة على محمل الإقلاع bootlaoder، وتستخدم كامل سعة ذاكرة البرنامج.

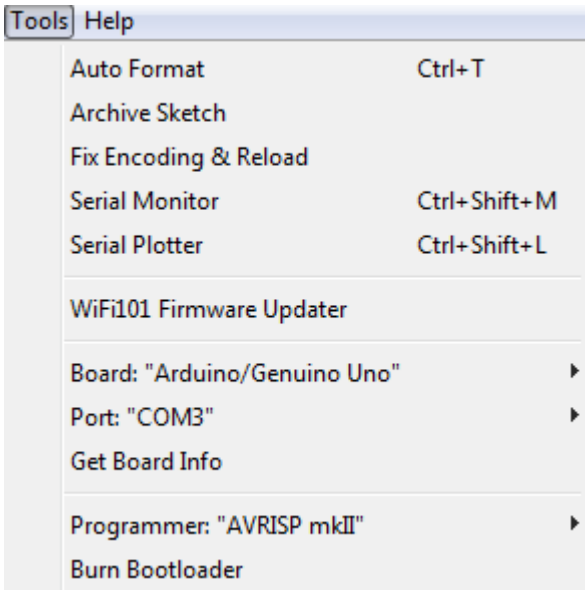


Export Compiled Binary: يتم تخزين ملف hex. (ملف ترميز لغة الآلة مكتوب بصيغة ست عشري hexadecimal) لاستخدامه في برنامج محاكاة، أو تحميله للوحة عن طريق أدوات أخرى.

Show Sketch folder: فتح مجلد الشيفرة البرمجية sketch الحالي.

Include library: إضافة مكتبات إلى الشيفرة البرمجية sketch الحالي بإضافة التعبير #include في بداية الشيفرة.

Add File: إضافة ملف برمجي، ويتم نسخه إلى مكان الشيفرة البرمجية الحالية.



تتضمن القائمة Tools الوظائف التالية:

Auto Format: يجعل الشيفرة البرمجية تظهر بشكل أنيق.

Archive Sketch: يتم أرشفة نسخة للشيفرة البرمجية الحالية بتنسيق zip.

Serial Monitor: فتح نافذة المراقبة التسلسلية.

Board: يتم تحديد اللوحة التي يتم العمل بها.

Port: تحتوي هذه القائمة على كل المنافذ التسلسلية المعرفة على الحاسب (فعلية أو ظاهرية). من خلالها يتم اختيار المنفذ الذي تتصل مع لوحة الأردوينو.

Programmer: تستخدم لاختيار المبرمجة عندما يتم برمجة اللوحة من دون استخدام لوصلة USB-serial. في العادة لن تحتاج إلى ذلك، إلا إذا أردت تحميل محمل الإقلاع bootloader إلى المتحكم الصغري.

Burn Bootloader: تستخدم لنقل محمل الإقلاع bootloader إلى المتحكم الصغري للوحة.

2-3-2-شريط أدوات وظائف عامة common functions toolbar

Verify: لفحص النص البرمجي من الأخطاء.

Upload: يتم تحويل الملف البرمجي إلى ترميز الآلة ومن ثم يتم تحميل الملف الثنائي إلى اللوحة عبر المنفذ الذي تم إعداده.

New: إنشاء نموذج جديد للمحرر.

Open: لفتح ملف مخزن.

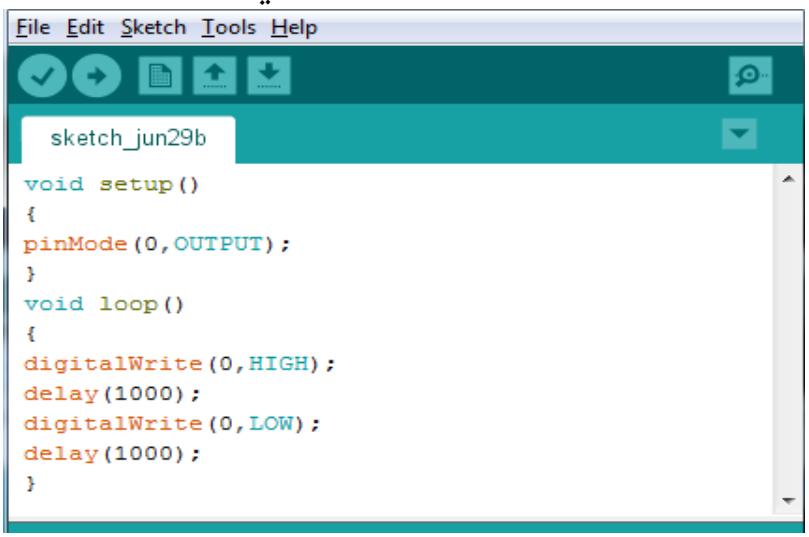
Save: حفظ الملف البرمجي.

Serial Monitor: فتح نافذة المراقبة التسلسلية.

الشكل(2-2):شريط أدوات وظائف عامة:Verify–Upload–New–Open –save – Serial Monitor

2-3-3-محرر النص text editor

يتم في محرر النص كتابة الشيفرة البرمجية الخاصة بعمل لوحة الأردوينو كما هو موضح في الشكل (2-3). سنتعلم لاحقاً كيفية كتابة الشيفرة البرمجية في بيئة التطوير.



الشكل (2-3): محرر النص لكتابة الشيفرة البرمجية.

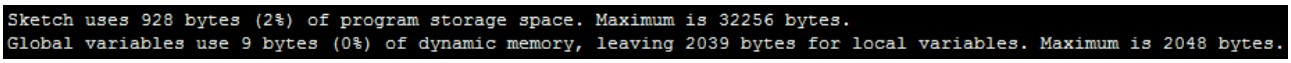
2-3-4-منطقة الرسالة message area

تقدم معلومات تفاعلية أثناء الحفظ، والتصدير، وتعرض الأخطاء كما هو موضح في الشكل (2-4).

الشكل (2-4): منطقة الرسالة لتقديم معلومات تفاعلية.

2-3-5- لوحة مراقبة النص text console

تعرض معلومات عن نتائج الشيفرة البرمجية مثل نسبة استخدام الذواكر، وأماكن التخزين، ورسائل الأخطاء كاملة، وغير ذلك كما هو موضح في الشكل (2-5).



الشكل (2-5): لوحة مراقبة النص.

تعرض الزاوية اليمنى السفلى للواجهة اللوحة والمنفذ التسلسلي اللذين تم إعدادهما كما هو موضح في الشكل (2-6).

الشكل (2-6): إظهار اسم اللوحة والمنفذ التسلسلي اللذين تم إعدادهما.

2-4- خطوات تحميل الشيفرة البرمجية إلى لوحة الأردوينو

يتم نقل الشيفرة البرمجية عن طريق بيئة التطوير إلى لوحة الأردوينو على المراحل التالية:

1) وصل لوحة الأردوينو إلى الحاسب عن طريق منفذ USB.
2) كتابة الشيفرة البرمجية.
3) تحميل ونقل الشيفرة إلى اللوحة.
4) تشغيل اللوحة.

في الفقرات التالية تم شرح هذه المراحل الأربعة.

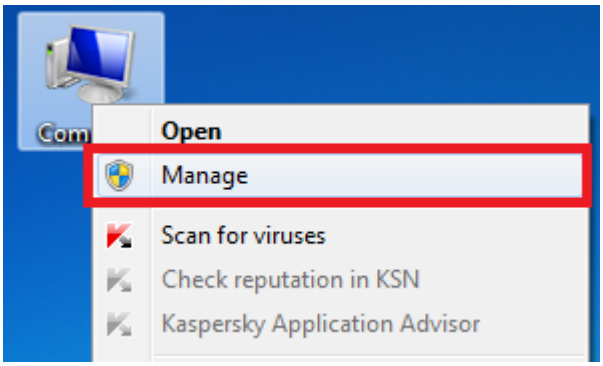
2-4-1-وصل لوحة الأردوينو إلى الحاسب عن طريق منفذ USB.

ذكرنا في الفصل الأول أن معظم لوحات الأردوينو تحتوي على منفذ USB يستخدم من أجل البرمجة، بالإضافة إلى تغذية اللوحة بالطاقة وتبادل البيانات. عندما نقوم بوصل اللوحة مع الحاسب عبر ذلك المنفذ سيتعرف عليها الجهاز على أنها منفذ COM افتراضي وله رقم. إذا لم يتعرف الجهاز على اللوحة كما هو موضح في الشكل(2-7) يجب عندئذ تنصيب التعريفات اللازمة والتي تأتي مع حزمة Arduino IDE التي قمت بتحميلها من الموقع المذكور سابقاً.

الشكل (2-7): رسالة تفيد بأنه لم يتم تعريف لوحة الأردوينو.

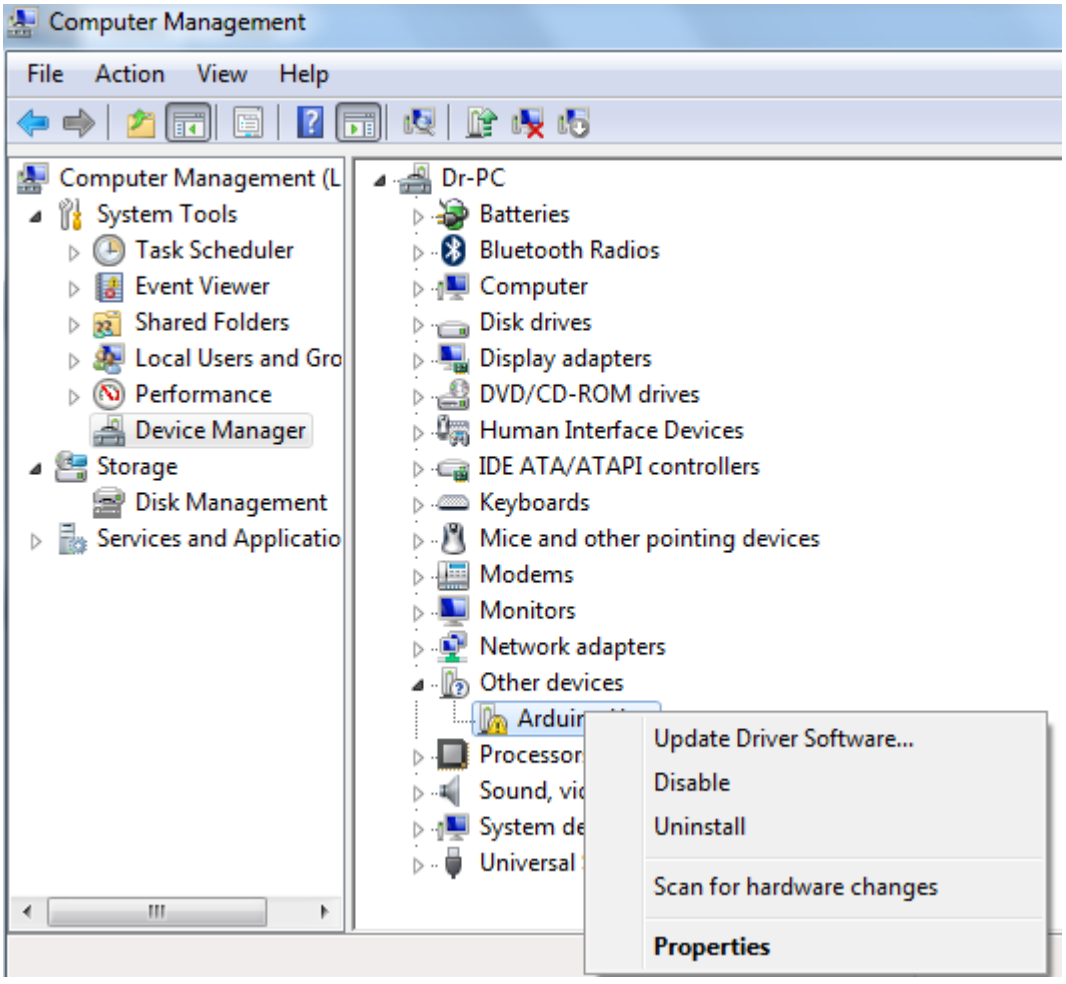
لتنصيب التعريف نتبع الخطوات التالية:

1- اضغط على زر الفارة الأيمن على أيقونة الحاسوب وافتح المدير manager كما في الشكل (2-8).



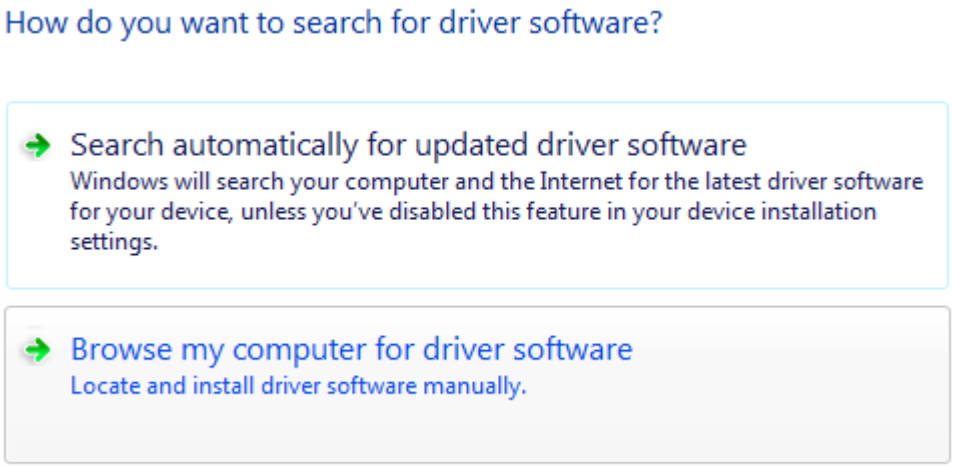
الشكل (2-8): فتح نافذة المدير manager.

2-اختر مدير الأجهزة Device Manager، ومنها اختر other devices ثم بزر الفارة الأيمن على Arduino (أو قد تكون Unknown devices) واختر Update driver software كما هو موضح في الشكل (2-9).



الشكل (2-9): نافذة مدير الأجهزة لتنصيب التعريف.

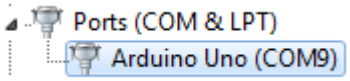
3-اختر تصفح الحاسب لاختيار التعريف كما هو موضح في الشكل (2-10)، واختر المجلد الذي به التعريف وهو drivers (المجلد الذي يتم تحميله مع البيئة التطويرية IDE من الموقع).



الشكل (2-10): اختيار التصفح عن طريق الحاسب لتحديد مكان التعريف.

4-تظهر رسالة إذا كنت ترغب تنصيب التعريف أو لا. نضغط OK وننتظر حتى تنتهي عملية التعريف.

5-بعد الانتهاء من تنصيب التعريف يمكن معرفة رقم المنفذ من مدير الأجهزة أيضاً كما هو موضح في الشكل (2-11).



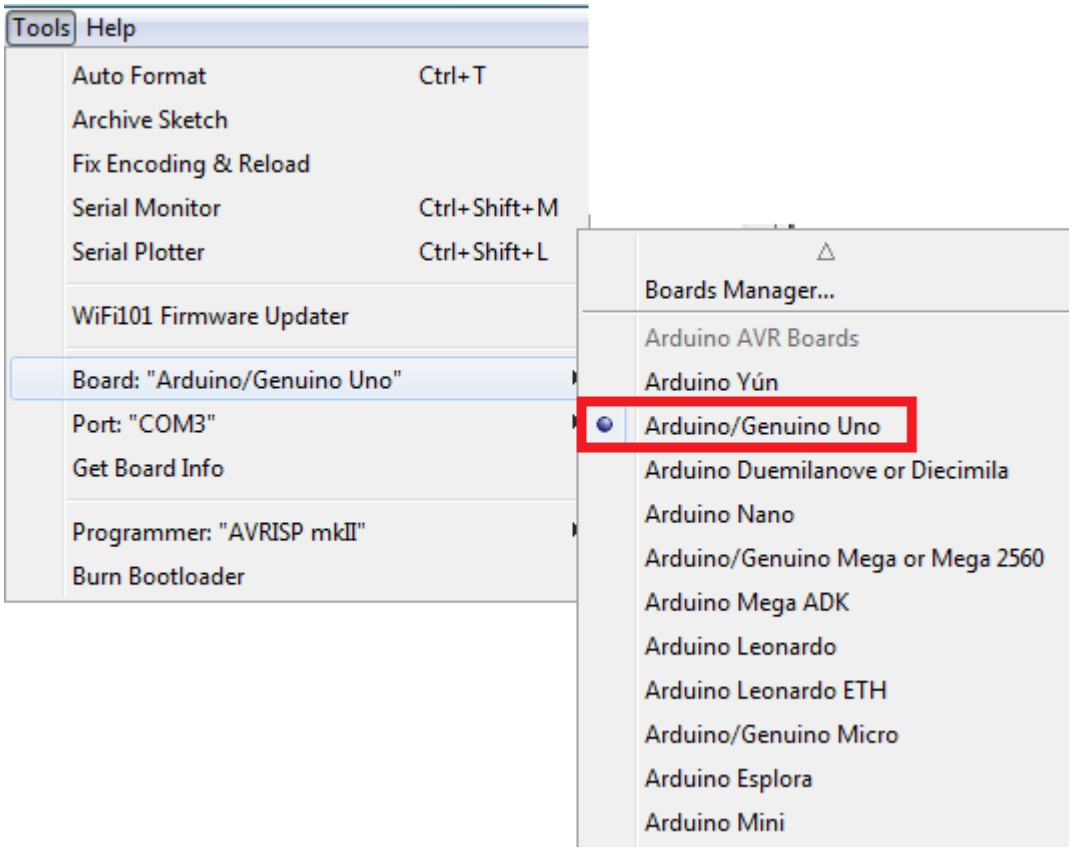
الشكل(2-11): تعريف لوحة الأردوينو كمنفذ COM، ورقم المنفذ 9 (يختلف من جهاز لآخر).

2-4-2-كتابة الشيفرة البرمجية

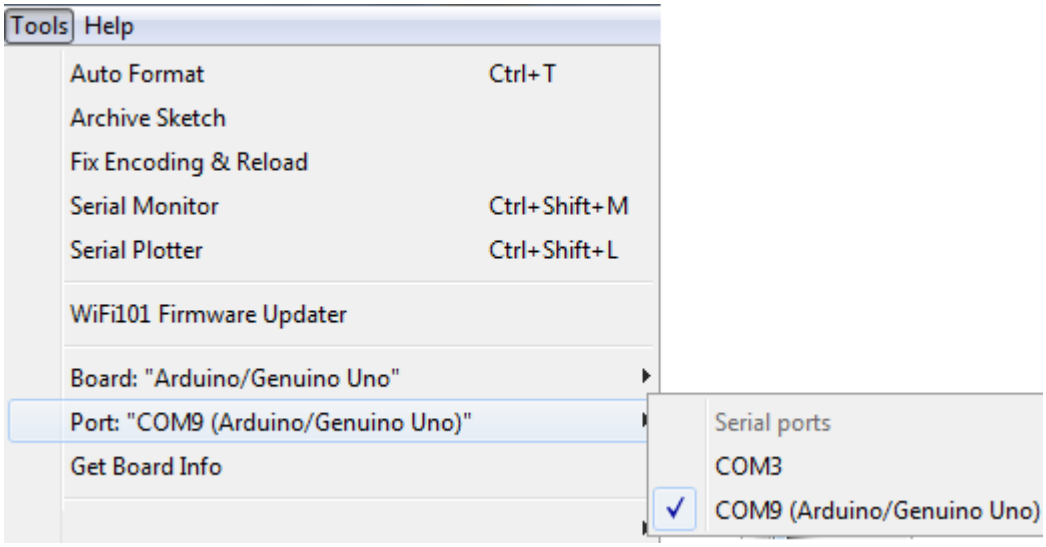
من أهم الخطوات هو تعلم واتقان واحتراف كتابة الشيفرات البرمجية للأردوينو في بيئة التطوير Arduino IDE، وهذا ما سنتعلمه في الأجزاء المتبقية من الكتاب.

2-4-3-تحميل الشيفرة البرمجية إلى لوحة الأردوينو

قبل نقل الشيفرة البرمجية إلى لوحة الأردوينو لابد من اختيار لوحة الأردوينو المناسبة من القائمة Tools ثم board، واختيار رقم المنفذ COM الصحيح من القائمة Tools ومن ثم serial Port كما هو موضح في الشكلين (2-12) (2-13).



الشكل (2-12): اختيار اسم اللوحة الصحيح، على سبيل المثال Arduino Uno.



الشكل (2-13): اختيار رقم منفذ COM الصحيح، على سبيل المثال COM9.

بعد ذلك نضغط Upload من شريط أدوات وظائف العامة المذكور في الفقرة (2-3-2). يضيء الثنائيان RX،TX في اللوحة للدلالة على نقل الشيفرة إلى المتحكم الصغري الرئيسي، ويظهر البرنامج رسالة بأن نقل الشيفرة تم بنجاح كما هو موضح في الشكل (2-14).

الشكل (2-14): رسالة تحميل الشيفرة بنجاح.

2-4-4-تشغيل الدارة

بعد الانتهاء من نقل الشيفرة البرمجية، ستعمل لوحة الأردوينو مباشرة ما دامت متصلة مع الحاسب عبر منفذ USB الذي يزودها بالطاقة اللازمة، ويمكن إزالة الاتصال بينهما، وتشغيل اللوحة عبر مصدر طاقة آخر كما تم ذكره سابقاً في الفصل الأول.

2-5-مبادئ كتابة الشيفرة البرمجية في بيئة التطوير Arduino IDE

2-5-1-هيكلة كتابة الشيفرة البرمجية

يتم كتابة الشيفرة البرمجية وفق الأسلوب والترتيب التالي:

```
#include <اسم مكتبة ما مطلوبة.h>
التصريح عن متغيرات وثوابت عامة

void setup( )
{
التصريح عن متغيرات وثوابت محلية
يتم هنا كتابة تعليمات إعداد وضبط المتحكم، ويتم تنفيذها مرة واحدة بعد تشغيل أو إعادة إقلاع اللوحة
}
void loop( )
{
التصريح عن متغيرات وثوابت محلية
يتم هنا كتابة البرنامج الرئيسي، ويتم تنفيذه بشكل متكرر
}

void .....
{
التصريح عن متغيرات وثوابت محلية
 يتم هنا كتابة تعليمات البرنامج الفرعي
}
```

2-5-2- التصريح عن المتغيرات والمصفوفات والثوابت

2-5-2-1- التصريح عن المتغيرات

يستخدم التصريح أو الإعلان عن المتغيرات لإعطاء اسم لقيمة متغيرة. وهذا ما يتم من خلال حجز موقع أو عدة مواقع ضمن الذاكرة RAM للمتحكم، يتم فيها تخزين القيم المتغيرة.

الفرق بين المتغيرات العامة والمحلية: المتغيرات العامة يتم التعامل معها ومشاهدتها في كل البرنامج (برنامج الإعداد setup( )، البرنامج الرئيسي loop( ) والبرامج الفرعي). المتغيرات المحلية يتم التعامل معها ومشاهدتها فقط في البرنامج الذي تم تعريفها فيه.

يتم التصريح عن المتغيرات في لغة C على الشكل التالي:

Variable_type Variable_name;

Variable_type : نوع أو نمط المتغير. Variable_name: اسم المتغير.

يوضح الجدول (2-1) أنماط المتغيرات Variable_type من أجل اللوحات التي تتضمن متحكم ATMega. من أجل البقية اللوحات يمكن زيارة الصفحة

www.arduino.cc/en/Reference/HomePage.

الجدول (2-1): أنماط تعريف المتغيرات.

نمط المتغير	المجال الذي يأخذه المتغير	السعة	مثال
boolean	False،True	1 Byte	boolean x;
byte	عدد صحيح من 0 وحتى 255	1 Byte	byte x;
unsigned int	عدد صحيح من 0 وحتى 65535	2 Byte	unsigned int x;
int short	عدد صحيح من -32768 إلى 32767	2 Byte	int x; short x;
unsigned long	عدد صحيح من 0 وحتى 2^{32}-1	4 Byte	unsigned long x;
long	عدد صحيح من -2^{31} وحتى 2^{31}-1	4 Byte	long x;
float double	عدد حقيقي من -3.4028235E+38 وحتى 3.4028235E+38	4 Byte	float x; double x;
char	حرف	1 Byte	char x;

يتم إسناد القيم للمتغيرات باستخدام علامة = . فيما يلي مثال على تعريف متغيرين y،x وإسناد قيمة لكل منهما.

```
byte x;
char y;
//…….
x=10;
y='A';
```

2-5-2-2-التصريح عن المصفوفات

المصفوفة هي مجموعة من المتغيرات التي يتم الوصول إليها برقم مفهرس. تأخذ المصفوفة في البرنامج بعد واحد، أو بعدين، ويمكن عدم تحديد حجمها. يبين الجدول (2-2) كيفية التصريح عن هذه المصفوفات.

الجدول (2-2): التصريح عن المصفوفات

نوع المصفوفة	كيفية التصريح عن المصفوفة	مثال
مصفوفة ببعد واحد	نمط عناصر المصفوفة اسم المصفوفة[عدد العناصر] ;	byte x[5];
مصفوفة ببعدين	نمط عناصر المصفوفة اسم المصفوفة[عدد الأسطر] [عدد الأعمدة] ;	byte x[5][4];

يبين الجدول (2-3) كيفية إسناد قيم للمصفوفات

الجدول (2-3): إسناد المصفوفات

نوع المصفوفة	كيفية إسناد قيم للمصفوفة
مصفوفة ببعد واحد	اسم المصفوفة[رقم العنصر] ;
مصفوفة ببعدين	اسم المصفوفة[رقم السطر] [رقم العمود] ;

فيما يلي مثال على كيفية تصريح عن مصفوفتين x, y وإسناد قيم لهما.

```
float  x[5];
byte y[5][4];
//….
x[0]=10.5;
y[0][0]=10;
```

لابد من الإشارة إلى أن البرنامج يعطي تعريفاً مختلفاً للحروف الكبيرة capital عن الحروف الصغيرة small، فمثلاً X تختلف عن x في البرنامج.

2-5-2-3-التصريح عن الثوابت

تستخدم الثوابت لإعطاء اسم لقيمة ثابتة. تختلف الثوابت عن المتغيرات في أن قيمتها تبقى ثابتة ولا تتغير (أي أنها للقراءة فقط). يتم التصريح عنها كما يلي:

const const_type const_name=value;

Const: كلمة مفتاحية للإشارة إلى التصريح عن ثابت.

const_type : نمط الثابت، ويأخذ الأنماط الموضحة في الجدول (2-1).

const_name : اسم الثابت.

Value : قيمة الثابت.

مثال:

```
const float pi = 3.14;   // التصريح عن ثابت
float x;
// ....
x = pi * 2;   // استخدام صحيح للثابت
pi = 7;       // غير صحيح، لأنه لا يمكن تعديل الثابت
```

ملاحظة: هناك تعليمة #define أيضاً تستخدم لإعطاء اسم لثابت، ولكن التصريح على شكل const أفضل من #define.

2-5-3-كتابة الأعداد الصحيحة

يتم ترميز الأعداد بشكل ثنائي أو ثماني أو عشري أو ست عشري كما هو معلوم. يبين الجدول (2-4) كيفية كتابة الأعداد في البرنامج.

الجدول (2-4): كتابة الأعداد

الترميز	كيفية كتابة العدد	مثال
الثنائي	أمام العدد نضع B	B1111011
الثماني	أمام العدد نضع 0	0173
العشري	يكتب العدد بشكل مباشر	123
الست عشري	أمام العدد نضع 0x	0x7B

2-5-4-بنى التحكم

2-5-4-1-الحلقات:

تهدف إلى الحلقة إلى تكرار تنفيذ مجموعة من الأوامر، ولها نوعان:

- غير شرطية: حلقة for.
- شرطية: حلقة while.

2-5-4-1-1- حلقة for

في هذه الحلقة يحدد المبرمج عدد مرات التكرار من خلال اختيار القيمة الابتدائية للعداد والقيمة النهائية ومقدار تزايد أو تناقص العداد. تكتب حلقة for من أجل تزايد المتغير x كما يلي:

```
for ( x = قيمة ابتدائية ; x<=  قيمة نهائية ; x = x + قفزة )
{  أوامر }
```

وتكتب حلقة for من أجل تناقص المتغير x كما يلي:

```
for ( x = قيمة ابتدائية ; x>=  قيمة نهائية ; x = x -قفزة )
{  أوامر }
```

مثال : تكرار أوامر 10 مرات.

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

عندما يكون مقدار القفزة 1، يمكن عندئذ كتابة الحلقة للمثال السابق كما يلي:

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

2-5-4-1-2- حلقة while :

تتبع هذه الحلقة لشرط، ما دام هذا الشرط محققاً تستمر أوامر الحلقة بالتنفيذ.

```
while ( شرط منطقي )
{  أوامر  }
```

ما هو الشرط المنطقي ؟

هو عملية مقارنة تكون نتيجتها True أو False، فعندما تكون النتيجة True (1) يتم تنفيذ أوامرwhile، أما False (0) فلا يتم تنفيذها . وأهم عمليات المقارنة comparison operators يبينها الجدول (2-5).

الجدول (2-5): عمليات المقارنة

عملية المقارنة	المعنى
x==y	x تساوي y
x!=y	x لا تساوي y
x<y	x أصغر من y
x>y	x أكبر من y
x<=y	x أصغر أو يساوي y
x>=y	x أكبر أو يساوي y

من الممكن أيضاً دمج عمليتي مقارنة باستخدام AND من خلال &&. مثال : A==3 && B!=8

من الممكن أيضاً دمج عمليتي مقارنة باستخدام OR من خلال ||. مثال : A==3 || B!=8

2-5-4-2-تعليمة IF الشرطية

تهدف إلى تقييد تنفيذ أوامر بتحقق شرط معين، بمعنى أنه في حال تحقق الشرط المنطقي يتم تنفيذ الأوامر.

```
if ( شرط منطقي )
{ أوامر }
```

يمكن استخدام أيضاً تركيبة if / else . في حال تحقق الشرط يتم تنفيذ الأوامر بعد if. في حال كون الشرط غير محقق يتم تنفيذ الأوامر بعد else

```
if ( شرط منطقي )
{  أوامر A }
else
{  أوامر B }
```

يمكن أيضاً فحص عدة شروط منطقية. في حال تحقق الشرط الأول يتم تنفيذ الأوامر بعد if. في حال تحقق الشرط الثاني يتم تنفيذ الأوامر بعد else if، ..... في حال عدم تحقق أي شرط يتم تنفيذ الأوامر بعد else.

```
if ( شرط منطقي )
{  أوامر A }
else if ( شرط منطقي )
{  أوامر B }
else
{  أوامر C }
```

2-5-4-3- تعليمة القفز goto

الهدف من هذه التعليمة الانتقال من الموقع الحالي للبرنامج إلى موقع آخر

```
label:

goto label;        // label القفز إلى الموقع
```

2-5-5- التعليقات Comments

تستخدم التعليقات لكي تعلم المبرمج أو الآخرين بكيفية عمل البرنامج. يتم تجاهلها من قبل المترجم، ولا يتم تنفيذها من قبل المعالج، لذلك فهي لا تأخذ أي مساحة من ذاكرة برنامج المتحكم الصغري. يوجد طريقتان لإضافة التعليقات :

تعليق على سطر واحد: يتم استخدام العلامة // كما هو موضح في المثال التالي:

```
x = 5;                // This is a single line comment.
```

تعليق متعدد الأسطر: تستخدم العلامة */ في البداية، والعلامة /* في النهاية كما في المثال التالي:

```
/* this is multiline comment
Don't forget closing the comment
*/
```

2-5-6-تعليمة التأخير الزمني

تهدف تعليمة التأخير الزمني إلى إيقاف تنفيذ البرنامج بشكل مؤقت ولفترة زمنية محددة. يوجد تعليمتان: الأولى تقدم تأخيراً زمنياً بالميلي ثانية

```
delay(value);
```

حيث value تأخذ مجال Unsigned long .

التعليمة الثانية تقدم تأخيراً بالميكرو ثانية.

```
delaymicroseconds(value);
```

حيث value تأخذ مجال Unsigned int .

مثال من أجل إيقاف تنفيذ البرنامج لمدة ثانية فإن تعليمة التأخير الزمني هي:

```
delay(1000);
```

2-5-7- البرنامج الفرعي

عندما تحتاج إلى تنفيذ أوامر عدة مرات، يمكن كتابتها في برنامج فرعي منفصل خارج البرنامج الرئيسي loop()، ومن ثم يتم استدعائها في كل مرة. تساعد البرامج الفرعية في تقليل نسبة استخدام ذاكرة البرنامج، وتقلل الوقت المطلوب لكتابة الشيفرة، وتجعل الشيفرة البرمجية المكتوبة مرتبة وأكثر وضوحاً. يوجد نمطان للبرامج الفرعية المستخدمة: إجرائية **Procedure** أو تابع **function**.

2-5-7-1- البرامج الفرعية على شكل إجرائية Procedure

تبدأ بكلمة void ولا تعيد أية قيمة. ويتم كتابتها واستدعاؤها على الشكل التالي:

```
Void  اسم البرنامج الفرعي ( نمط المتغير  اسم المتغير .....)      // البرنامج الفرعي
{
التصريح عن متغيرات محلية
  جسم البرنامج الفرعي
}
اسم البرنامج الفرعي  ( (قيمة يراد تمريرها للبرنامج الفرعي .....; // استدعاء البرنامج الفرعي
```

مثال : برنامج فرعي لضرب عددين بدون إعادة الناتج.

```
void loop() {
 Multiply(2,3);
}
void  Multiply(int x, int y){
 result = x * y;
}
```

2-5-7-2- البرامج الفرعية على شكل تابع function

تبدأ بنمط القيمة المستعادة (الجدول (2-1))، وتنتهي بـ Return، لتعيد قيمة ما، ويتم كتابتها واستدعاؤها على الشكل التالي:

```
(..... اسم المتغير  نوع المتغير ) اسم البرنامج الفرعي  نمط القيمة المستعادة
{
التصريح عن متغيرات تظهر في البرنامج الفرعي  فقط
جسم البرنامج الفرعي
Return ( -قد يكون متغير القيمة المستعادة)
}

Value = اسم البرنامج الفرعي ( اسم البرنامج الفرعي ) ;... (قيمة يراد تمريرها للبرنامج الفرعي // استدعاء البرنامج الفرعي
```

مثال : برنامج فرعي لضرب عددين ويعيد الناتج.

```
void loop() {
  int k;
  k = Multiply(2,3);       // k now contains 6
}

int Multiply(int x, int y){
  int result;
  result = x * y;
  return result;
}
```

الفصل الثالث: التطبيقات العملية

الثنائيات الضوئية LEDs

3-1-1-مقدمة

سنبدأ في هذا الجزء تنفيذ أول دارة عملية بسيطة للتحكم بتشغيل وإطفاء ثنائيات ضوئية LEDs من خلال لوحة الأردوينو. ذكرنا سابقاً أن للوحة عدة مخارج رقمية يمكن من خلالها إخراج 1 منطقي (يكافئ 5V+ للوحة التي تعمل بنفس الجهد)، و0 منطقي (يكافئ 0V). بالتالي عند وصل ثنائيات ضوئية إلى هذه المخارج سنتمكن من التحكم بها. سنتعلم في الجزء التالي تعليمات الإدخال والإخراج الرقمية في بيئة التطوير Arduino IDE، وبعد ذلك ننتقل إلى الأمثلة والتطبيق العملي.

3-1-2-تعليمات الدخل والخرج الرقمية Digital I/O

pinMode(pin,mode)

تستخدم هذه التعليمة لتحديد رجل معينة كدخل أو خرج، أو دخل مع تفعيل لمقاومة السحب الداخليةinternal pull-up resistor.

Pin: رقم الرجل المطلوب ضبط نمطها.

mode: تأخذ إحدى القيم التالية INPUT،OUTPUT، INPUT_PULLUP لتحديد نمط الرجل دخل، خرج، دخل مع تفعيل مقاومة السحب.

digitalWrite(pin,value)

يتم من خلال هذه التعليمة إخراج 1 منطقي، أو 0 منطقي على رجل محددة. جهد 1 منطقي 5V+ من أجل لوحات 5V+، وجهد 0 منطقي 0V.

Pin: رقم الرجل المطلوب الإخراج عليها.

value: تأخذ إحدى القيمتين التاليتين: LOW (يمكن كتابة 0 عوضاً عنها)، HIGH (يمكن كتابة 1 عوضاً عنها) لإخراج 0 منطقي أو 1 منطقي.

ملاحظة: من المفروض أن هذه التعليمة تستخدم مع تعليمة pinMode بحيث يكون بارامتر mode يساوي OUTPUT. ولكن بفرض أن mode يساوي INPUT عندئذ فإن تعليمة digitalWrite ستعمل على تفعيل أو عدم تفعيل مقاومة السحب تبعاً لقيمة value تساوي HIGH أو LOW. على كل حال من الأفضل تفعيل مقاومة السحب من خلال تعليمة pinMode كما تم ذكره سابقاً.

digitalRead(pin)

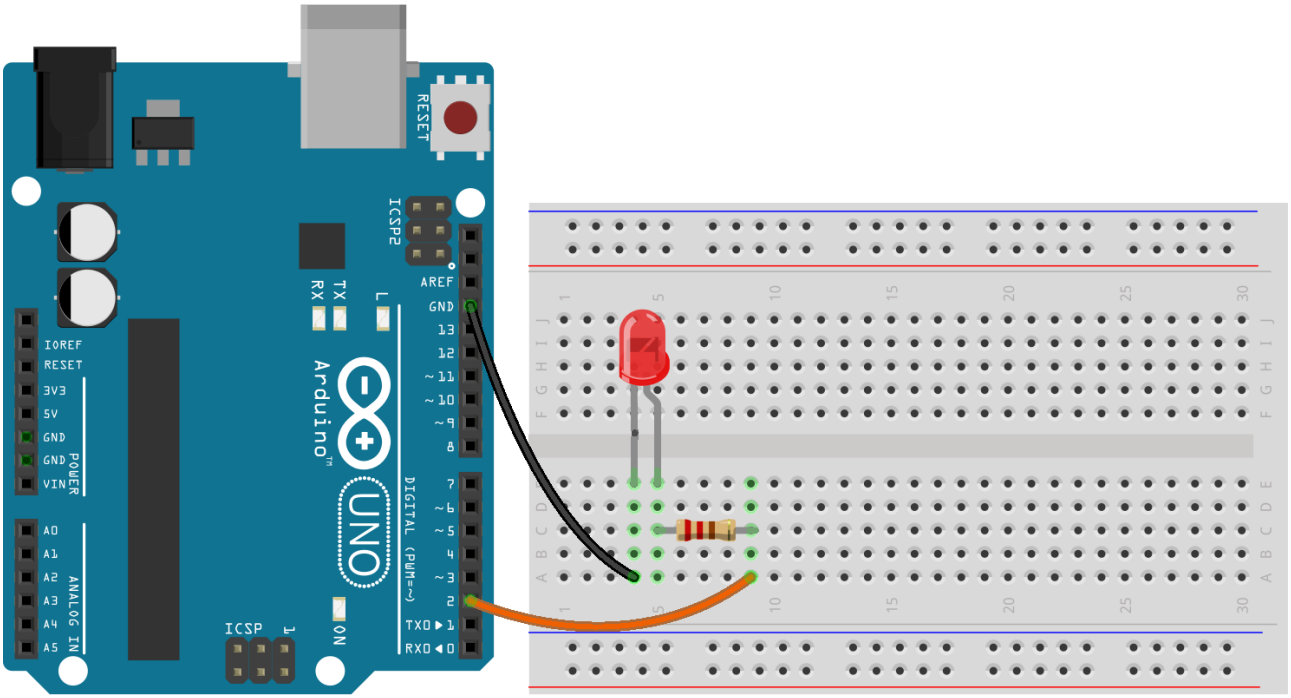
يتم من خلال هذه التعليمة قراءة حالة رجل محددة وتعيد إحدى القيمتين التاليتين HIGH أو LOW. الجهد الأعلى من 3V من أجل لوحات 5V+ يتم اعتباره HIGH.

Pin: رقم رجل الإدخال الرقمية.

3-1-3- التطبيقات العملية

3-1-3-1-تشغيل وإطفاء ثنائي ضوئي بشكل متكرر Blinking LED

يوضح الشكل (3-1) الدارة العملية لتشغيل وإطفاء ثنائي ضوئي عن طريق لوحة أردوينو أونو. تم وصل مصعد الثنائي إلى الرجل ذات الرقم 2 للوحة عن طريق مقاومة قيمتها 220Ω، ومهبط الثنائي تم وصله إلى أرضي اللوحة. تبعاً لذلك عندما نطبق 1 منطقي (+5V) على الرجل 2 للوحة سيضيئ الثنائي، وعندما نطبق 0 منطقي سينطفئ الثنائي. بتكرار تطبيق 1 و0 على الرجل 2 سنلاحظ أن الثنائي سيضيئ وينطفئ بشكل متكرر، مع ترك فاصل زمني بينهما.

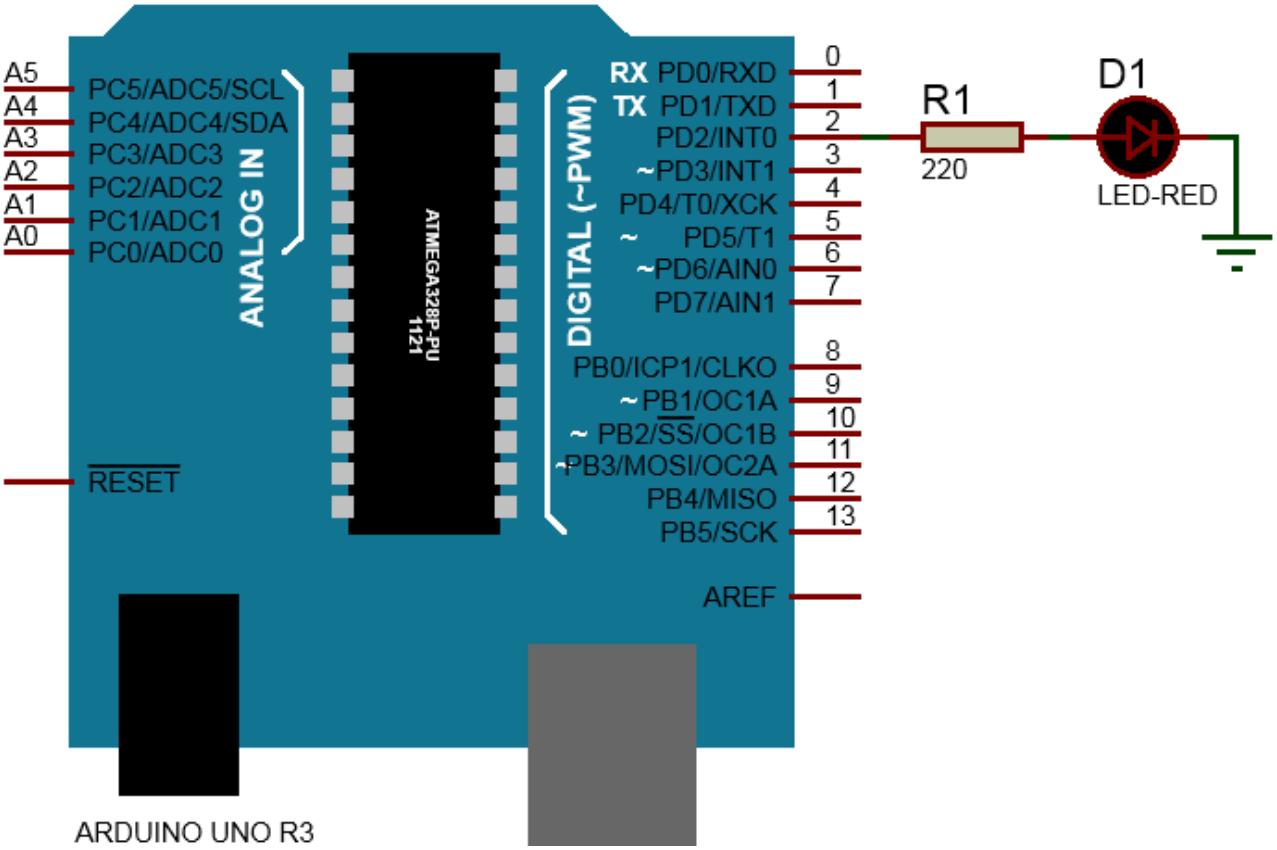


الشكل (3-1): الدارة العملية لتشغيل وإطفاء ثنائي ضوئي بشكل متكرر.

3-1-3-1-1-الكود البرمجي

```
void setup()
{
 pinMode(2,OUTPUT);                    // تفعيل الرجل 2 على أنها خرج
}

void loop()
{
 digitalWrite(2,HIGH);                 // إخراج 1 منطقي على الرجل 2
 delay(1000);                          // تأخير زمني لمدة ثانية
 digitalWrite(2, LOW);                 // إخراج 0 منطقي على الرجل 2
 delay(1000);                          // تأخير زمني لمدة ثانية
}
```

2-1-3-1-3-محاكاة تشغيل وإطفاء ثنائي ضوئي من خلال برنامج Proteus



الشكل (3-2): محاكاة دارة تشغيل وإطفاء ثنائي ضوئي من خلال برنامج Proteus.

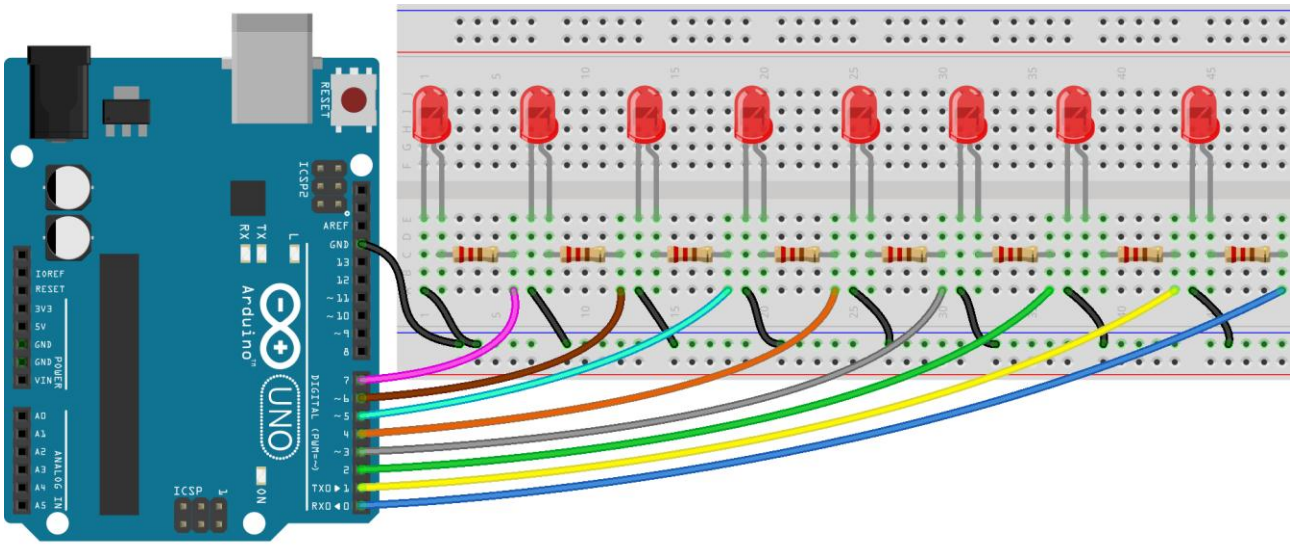
2-3-1-3-تشغيل وإطفاء مجموعة ثنائيات ضوئية LEDs

يوضح الشكل (3-3) الدارة العملية لتشغيل وإطفاء مجموعة ثنائيات ضوئية LEDs عن طريق لوحة أردوينو أونو. لسوء الحظ لا يوجد تعليمة جاهزة في برنامج Arduino IDE تمكننا من التعامل مع بوابة (عدة أرجل) معاً لإخراج قيمة ما. الحل يكون بكتابة برنامج فرعي يؤدي تلك الوظيفة. لنسمي هذا البرنامج PORTA(value)، ويعمل على إخراج القيمة value على الأرجل الرقمية من 0 وحتى 7 للوحة الأردوينو. تكتب القيمة value بـ 8 بتات، تمثل الخانة الأقل أهمية الرجل 0 وهكذا. طريقة كتابة القيمة value تم ذكره في فقرة كتابة الأعداد. كود البرنامج الفرعي PORTA(value) هو

```
void PORTA(byte value)
{
 byte i;
 for(i=0;i<=7;i++)
 {digitalWrite(i,bitRead(value,i));}
}
```

ملاحظات على البرنامج:

- تم تعريف value وفق النمط byte لأن قيمته تأخذ 8 bit .
- التعليمة bitRead(value,i) تعمل على قراءة البت ذي الرقم i من العدد value، والنتيجة بالتأكيد ستكون إما 0 أو 1.
- يعمل البرنامج كما يلي: مع أول قيمة i=0، يكون لدينا digitalWrite(0,bitRead(value,0)) ، وهذا ما يسمح بإخراج قيمة البت ذي الرقم 0 (الخانة الأقل أهمية) على الرجل الرقمية 0 للوحة الأردوينو. يكرر نفس الأسلوب لبقية الأرجل الرقمية.



الشكل (3-3): الدارة العملية لتشغيل وإطفاء مجموعة ثنائيات ضوئية بشكل متكرر.

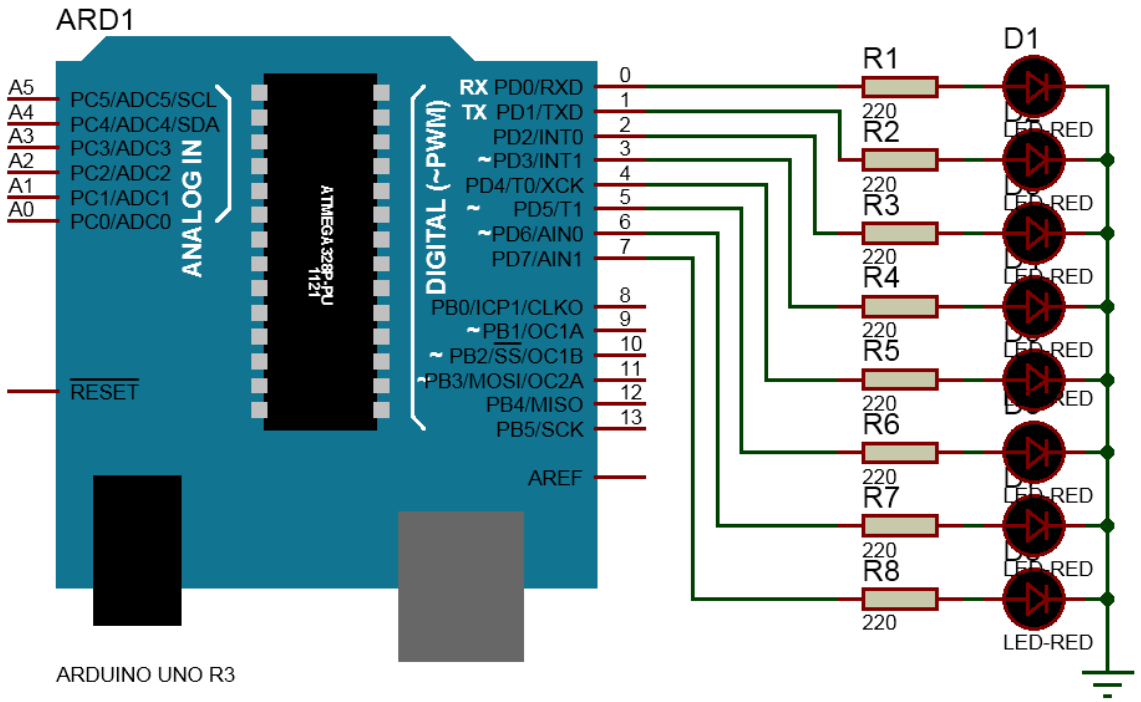
3-1-2-3-1-الكود البرمجي

```
void setup()
{
byte i;
 for(i=0;i<=7;i++)          // تفعيل الأرجل من 0 وحتى 7 كمخارج
{pinMode(i,OUTPUT);}
}

void loop ()
{
PORTA(B11111111);      // استدعاء البرنامج الفرعي لإخراج 1 منطقي على كل الأرجل

delay(1000);               // تأخير زمني لمدة ثانية
PORTA(B00000000);      //استدعاء البرنامج الفرعي لإخراج 0 منطقي على كل الأرجل
delay(1000);               // تأخير زمني لمدة ثانية
}
```

برنامج فرعي لإخراج قيمة على الأرجل من 0 وحتى 7 // void PORTA(byte value)

```
{
 byte i;
 for(i=0;i<=7;i++)
 {digitalWrite(i,bitRead(value,i));}
}
```

3-1-2-2-3-محاكاة تشغيل وإطفاء مجموعة ثنائيات ضوئية من خلال برنامج Proteus

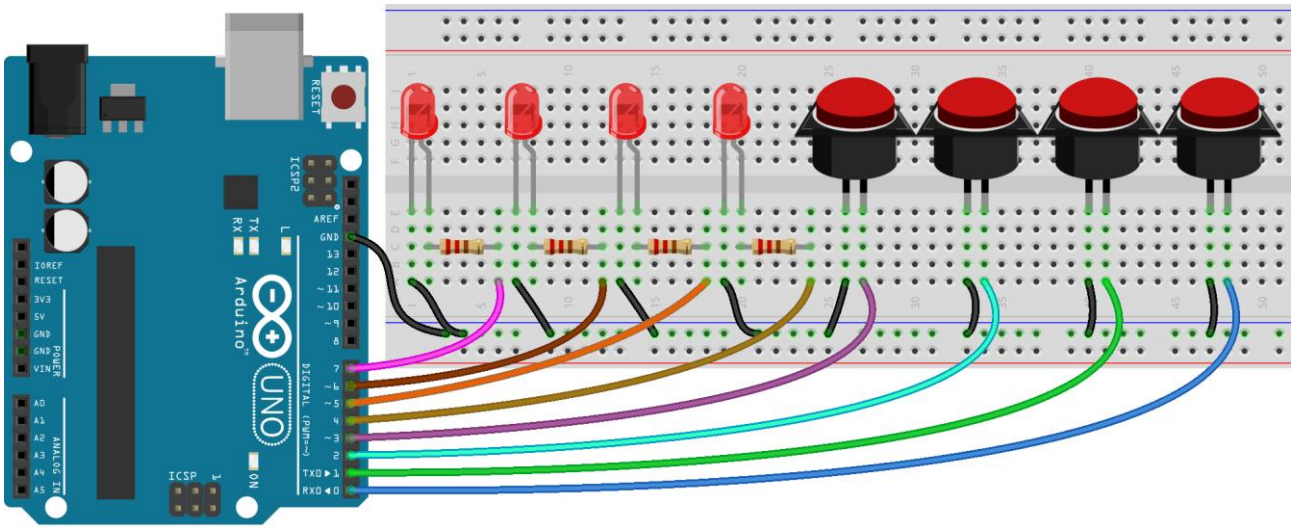


الشكل (3-4): محاكاة دارة تشغيل وإطفاء مجموعة ثنائيات ضوئية من خلال برنامج Proteus.

ملاحظة هامة: تتصل الرجل 0 للمتحكم الرئيسي ATmega328 مع رجل الإرسال التسلسلية (TX) للمتحكم الثانوي ATmega16U2، حيث يرسل من خلال هذا الخط الكود البرمجي. بالتالي عند تنفيذ الوصل في الشكل (3-3) ومن ثم تحميل الكود البرمجي عبر بيئة التطوير Arduino IDE سيحدث خطأ في عملية نقل الكود وستظهر رسالة خطأ. لتفادي ذلك يمكن أن يتم تحميل الكود البرمجي في البداية من دون وصل السلك ما بين الرجل 0 والمقاومة، وبعد الانتهاء من التحميل يتم إعادة السلك. أو يمكن استخدام رجل رقمية أخرى غير الرجل 0 وعندئذ يمكن تحميل الكود من دون أية مشاكل.

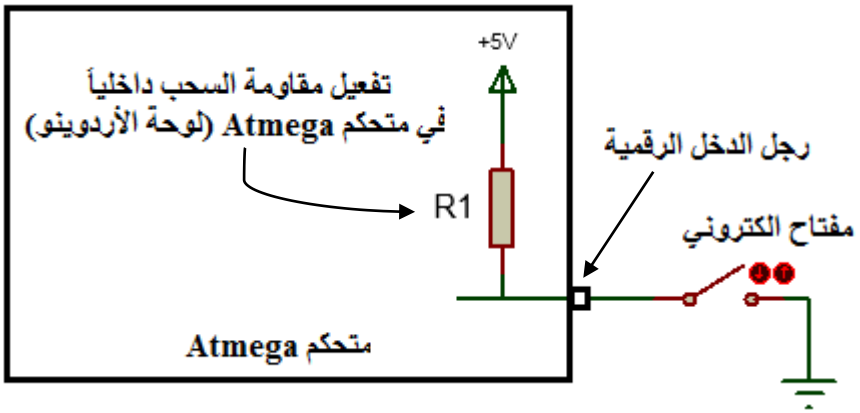
3-1-3-3-التحكم بثنائيات ضوئية من خلال مفاتيح الكترونية

في هذه الفقرة تم ربط أربعة مفاتيح الكترونية مع الأرجل من 0 وحتى 3 للوحة، والتي ستعمل تبعاً لذلك كمداخل رقمية، وتم ربط أربعة ثنائيات ضوئية مع الأرجل من 4 وحتى 7 كما هو موضح في الشكل (3-5). كل مفتاح من المفاتيح الأربعة يتحكم بتشغيل وإطفاء ثنائي ضوئي.



الشكل (3-5): الدارة العملية لتشغيل وإطفاء ثنائيات ضوئية من خلال مفاتيح الكترونية

تم تفعيل مقاومة السحب Pull-up الداخلية لكل رجل من أرجل الدخل من 0 وحتى 3 بحيث عندما يكون المفتاح بحالة OFF تكون رجل الدخل مطبق عليها +5V (1 منطقي) بسبب مقاومة السحب، وعندما يكون المفتاح بحالة ON تكون الرجل متصلة مع الأرضي (0 منطقي) كما هو موضح في الشكل (3-6).



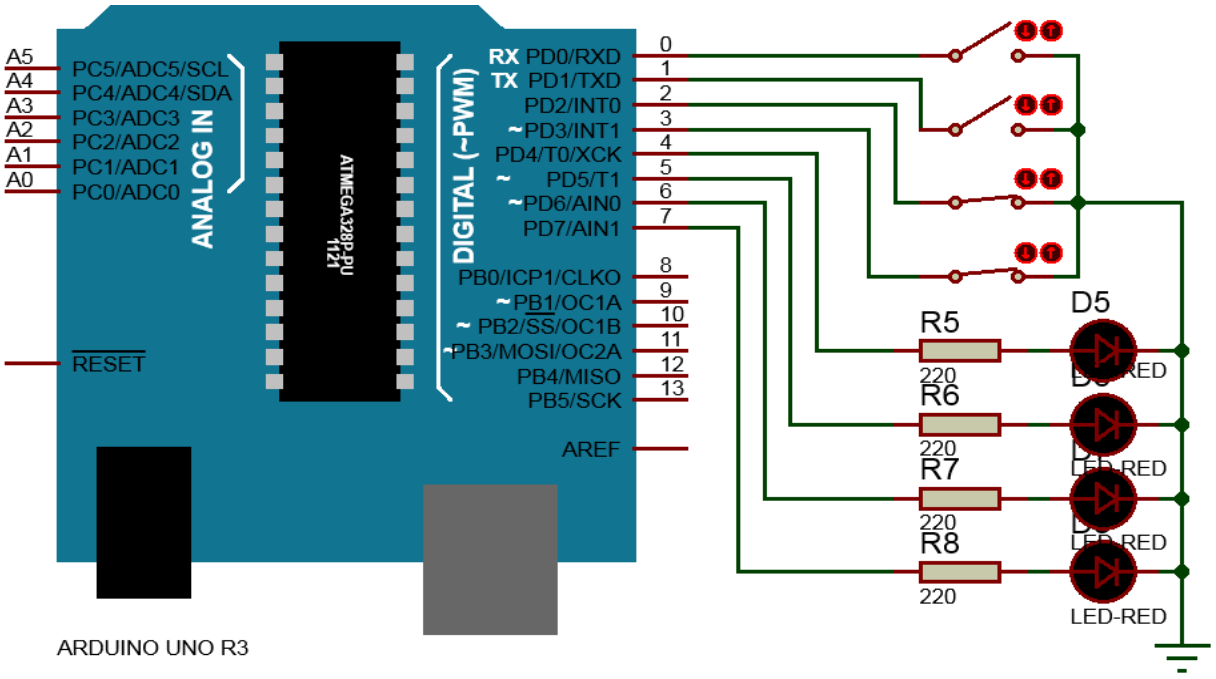
الشكل (3-6): تفعيل مقاومة السحب الداخلية وربط مفتاح الكتروني إلى رجل الدخل الرقمية.

3-1-3-3-1-الكود البرمجي

```
void setup() {
  byte i;
  for (i = 0; i <= 3; i++)
  { pinMode(i, INPUT_PULLUP);}
  for (i = 4; i <= 7; i++)
  { pinMode(i, OUTPUT); }
}

void loop() {
  boolean x1, x2, x3, x4;
  x1 = digitalRead(0);
  x2 = digitalRead(1);
```

```
 x3 = digitalRead(2);
 x4 = digitalRead(3);
 if (x1 == HIGH) {
   digitalWrite(4, HIGH);
 } else {
   digitalWrite(4, LOW);
 }
 if (x2 == HIGH) {
   digitalWrite(5, HIGH);
 } else {
   digitalWrite(5, LOW);
 }
 if (x3 == HIGH) {
   digitalWrite(6, HIGH);
 } else {
   digitalWrite(6, LOW);
 }
 if (x4 == HIGH) {
   digitalWrite(7, HIGH);
 } else {
   digitalWrite(7, LOW);
 }
}
```

3-1-3-3-2-محاكاة التحكم بثنائيات ضوئية من خلال مفاتيح الكترونية من خلال برنامج Proteus



الشكل (3-7): محاكاة دارة التحكم بثنائيات ضوئية من خلال برنامج Proteus.

السبع قطع الضوئية 7-Segment

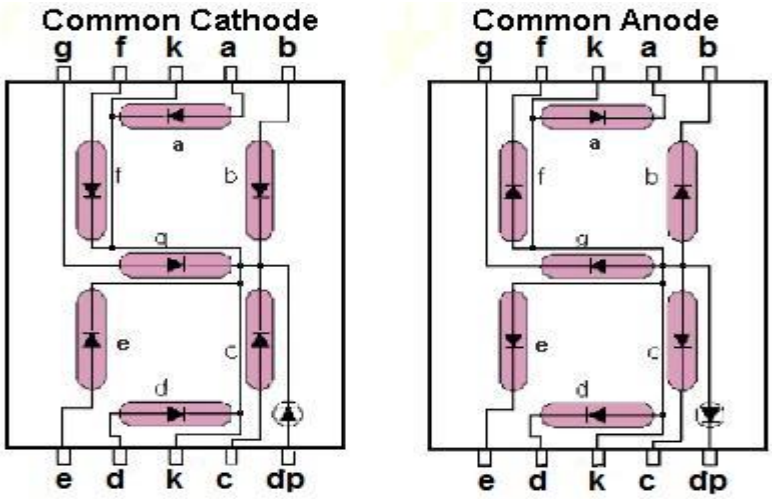
3-2-1-مقدمة

السبع قطع ضوئية هي عبارة عن سبع ثنائيات ضوئية تم ترتيبها بطريقة يمكن من خلالها إظهار الأرقام. للسبع قطع ضوئية نمطان كما هو موضح في الشكل (3-8).

مهبط مشترك common cathode: تتصل فيها مهابط الثنائيات معاً. تتمثل نقطة الوصل المشتركة بالرجل k في الأعلى والأسفل في الشكل (3-8-a). تعمل السبع قطع في هذه الحالة من خلال وصل إحدى الأرجل K إلى 0V (أي تطبيق 0V على مهابط الثنائيات)، ويتم إضاءة كل ثنائي ضوئي على حدى من خلال تطبيق 5V+ على الرجل الخاصة به a،b،c،d،e،f،g .

مصعد مشترك common anode: تتصل فيها مصاعد الثنائيات معاً. تتمثل نقطة الوصل المشتركة بالرجل k في الأعلى والأسفل في الشكل (3-8-b). تعمل السبع قطع في هذه الحالة من خلال وصل إحدى الأرجل K إلى 5V+ (أي تطبيق 5V+ على مصاعد الثنائيات)، ويتم إضاءة كل ثنائي ضوئي على حدى من خلال تطبيق 0V على الرجل الخاصة به a،b،c،d،e،f،g .

يوجد ثنائي ضوئي إضافي dp يضيء كنقطة.



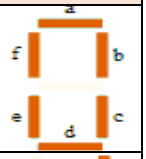
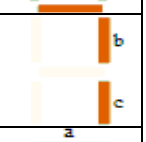
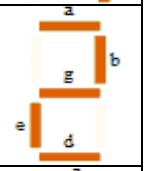
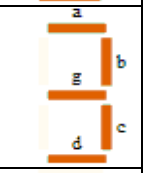
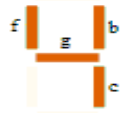
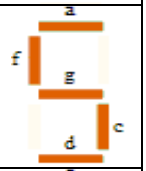
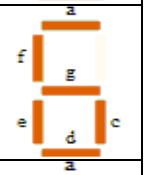
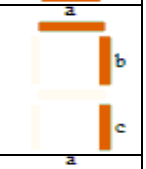
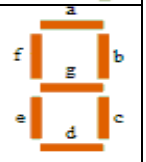
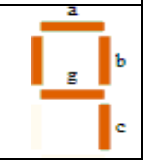
(a) (b)

الشكل (3-8): السبع قطع ضوئية، (a) مهبط مشترك، (b) مصعد مشترك.

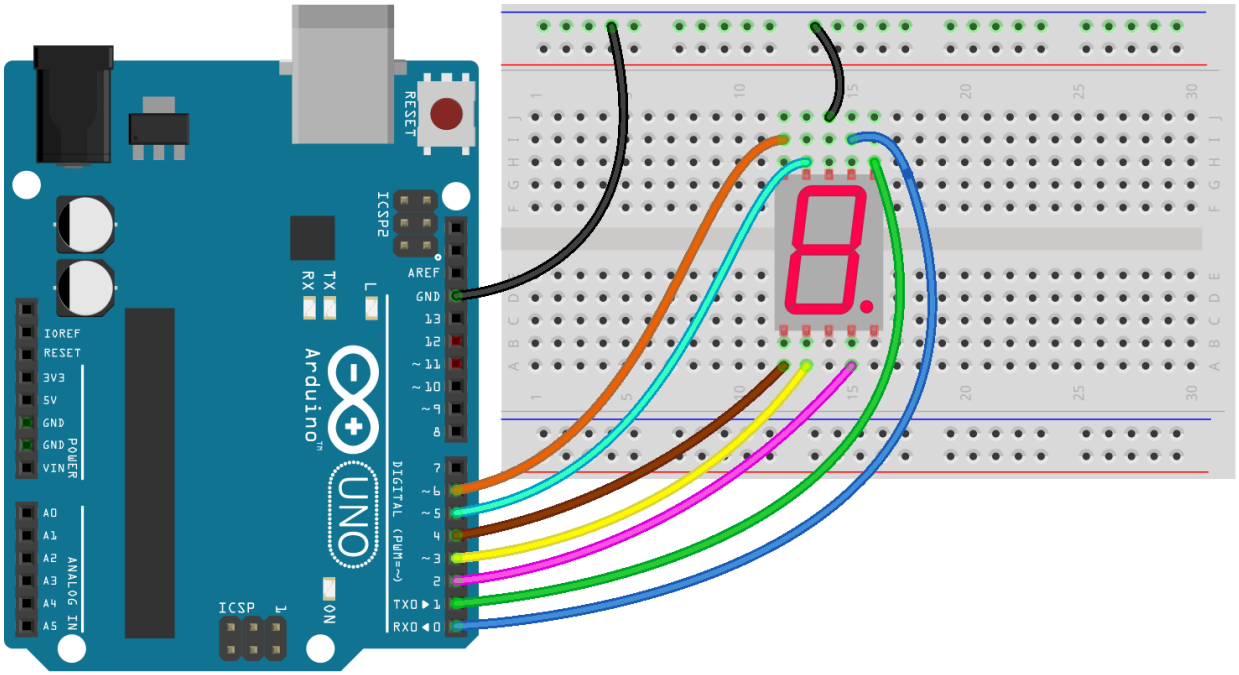
3-2-2-إظهار الأرقام على السبع قطع ضوئية ذات نمط مهبط مشترك

استناداً لما تم ذكره في الفقرة (3-2-1) ، لإظهار الرقم 0 مثلاً على السبع قطع ضوئية ذات نمط مهبط مشترك لابد من تطبيق 1 منطقي على الأرجل a،b،c،d،e،f للسبع قطع ضوئية، و0 منطقي على الرجل g. يطبق نفس الأسلوب لبقية الأرقام. يوضح الجدول (3-1) القيم المطلوب تطبيقها على الأرجل a،b،c،d،e،f،g لإظهار الأرقام من 0 وحتى 9.

الجدول (3-1): شيفرات الأرقام من 0 وحتى 9 للسبع قطع ضوئية

g	f	e	d	c	b	a	الرقم
0	1	1	1	1	1	1	
0	0	0	0	1	1	0	
1	0	1	1	0	1	1	
1	0	0	1	1	1	1	
1	1	0	0	1	1	0	
1	1	0	1	1	0	1	
1	1	1	1	1	0	1	
0	0	0	0	1	1	1	
1	1	1	1	1	1	1	
1	1	0	1	1	1	1	

لنقم الآن بوصل لوحة الأردوينو مع السبع قطع ضوئية كما هو موضح في الشكل (3-9) حيث تتصل الرجل 0 للوحة مع الرجل a للسبع قطع، والرجل 1 مع الرجل b....وهكذا. رجل التفعيل K (الرجل المشتركة) يتم وصلها للأرضي. إذا قمنا بإخراج شيفرات الأرقام الموضحة في الجدول (3-1) على الأرجل من 0 وحتى 6 للوحة الأردوينو، ستعد السبع قطع الأرقام من 0 وحتى 9.



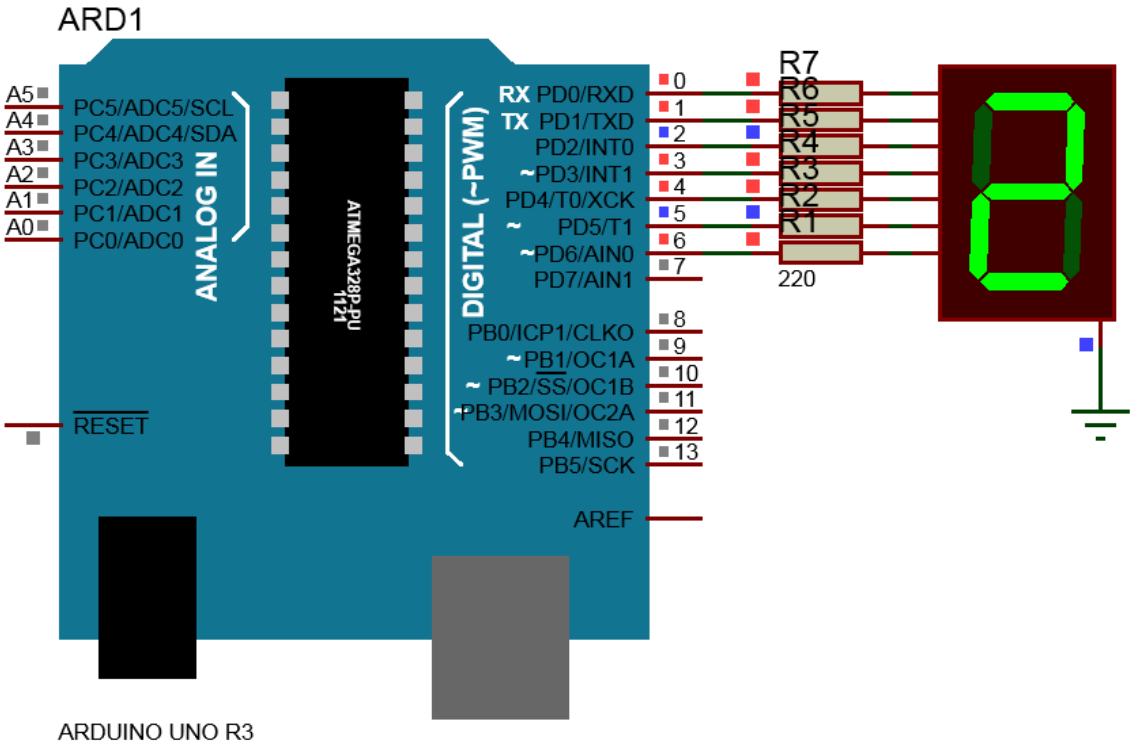
الشكل (3-9): الدارة العملية لربط لوحة الأردوينو مع السبع قطع ضوئية.

3-2-2-1-الكود البرمجي

```
void setup() {
  byte i;
  for (i = 0; i <= 6; i++)                  // تفعيل الأرجل من 0 وحتى 6 كمخارج
  {
    pinMode(i,OUTPUT);
  }
}

void loop() {
  PORTA(B0111111);                          // إخراج شيفرة الرقم 0
  delay(1000);
  PORTA(B0000110);                          // إخراج شيفرة الرقم 1
  delay(1000);
  PORTA(B1011011);                          // إخراج شيفرة الرقم 2
  delay(1000);
  PORTA(B1001111);                          // إخراج شيفرة الرقم 3
  delay(1000);
  PORTA(B1100110);                          // إخراج شيفرة الرقم 4
  delay(1000);
  PORTA(B1101101);                          // إخراج شيفرة الرقم 5
  delay(1000);
```

```
 PORTA(B1111101);                    // إخراج شيفرة الرقم 6
 delay(1000);
 PORTA(B0000111);                    // إخراج شيفرة الرقم 7
 delay(1000);
 PORTA(B1111111);                    // إخراج شيفرة الرقم 8
 delay(1000);
 PORTA(B1101111);                    // إخراج شيفرة الرقم 9
 delay(1000);
}

void PORTA(byte value)   // برنامج فرعي لإخراج قيمة على الأرجل من 0 وحتى 6
{
 byte i;
 for (i = 0; i <= 6; i++)
 {
  digitalWrite(i,bitRead(value,i));
 }
}
```

3-2-2-2-محاكاة إظهار أرقام على سبع قطع ضوئية باستخدام برنامج Proteus.



الشكل (3-10): محاكاة دارة السبع قطع ضوئية من خلال برنامج Proteus.

3-2-3-إظهار أرقام من خلال عدة أجزاء سبع قطع ضوئية

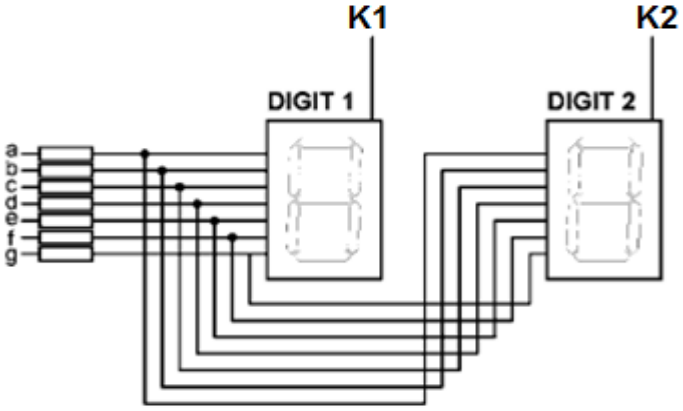
يتم وصل قطعتي سبع ثنائيات ضوئية لإظهار رقمين كما هو موضح بالشكل (3-11). بفرض أن القطعتين من نمط مهبط مشترك، وتم وصل كل من K1، وK2 إلى الأرضي، عندئذ عند تطبيق شيفرة رقم ما عن طريق g،f،e،d،c،b،a فإن الرقم سيظهر على القطعتين بنفس الوقت، ولا يمكن إظهار رقمين مختلفين وفق هذا الأسلوب. حل تلك المشكلة بأن يتم وصل K1 و K2 إلى منفذي خرج رقميين للوحة الأردوينو، ونطبق الخطوات التالية:

1-نضع 0V على الرجل K1 (تفعيل السبع قطع الأولى)، و نضع 5V على الرجل K2 (إلغاء تفعيل السبع قطع الثانية)، بعد ذلك نرسل شيفرة الرقم المطلوب إظهاره على السبع قطع الأولى على الأرجل g،f،e،d،c،b،a.

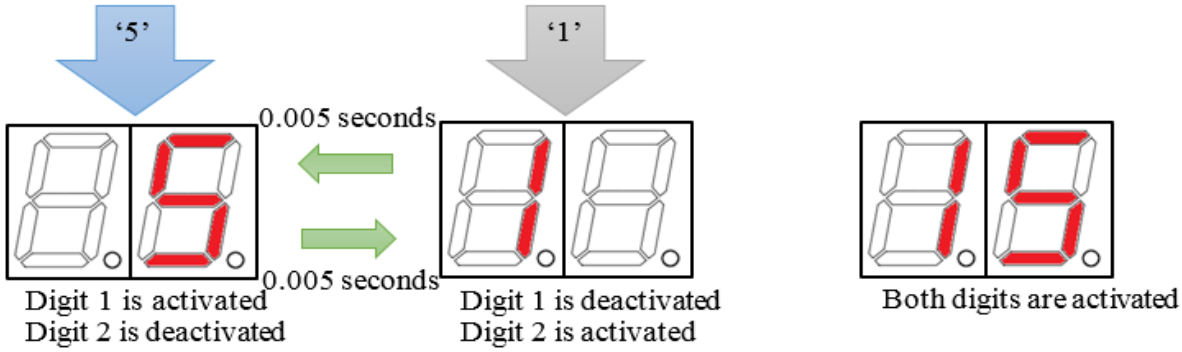
2-نضع 5V على الرجل K1 (إلغاء تفعيل السبع قطع الأولى)، و نضع 0V على الرجل K2 (تفعيل السبع قطع الثانية)، بعد ذلك نرسل شيفرة الرقم المطلوب إظهاره على السبع قطع الثانية على الأرجل g،f،e،d،c،b،a.

3-نكرر الخطوتين السابقتين بشكل سريع، ليظهر العدد المطلوب إظهاره على قطعتي السبع ثنائيات ضوئية بدون أن تلاحظ العين تبديل الإظهار من واحدة لأخرى وهذا يعود لعملية المسح السريعة.

يبين الشكل (3-12) كيفية تنفيذ الخطوات السابقة بحيث نشاهد الرقم على كلتا الخانتين.



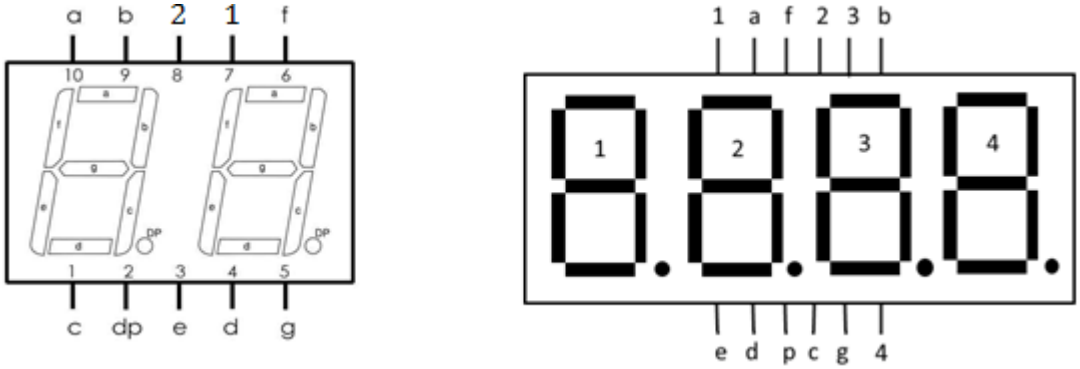
الشكل (3-11): وصل قطعتي سبع قطع ضوئية.



الشكل (3-12): إظهار عدد على خانتين سبع قطع. يتم تفعيل الخانة الأولى وإظهار رقم عليها، ثم يتم تفعيل الخانة الثانية وإظهار الرقم الآخر، بتكرار العملية سريعاً يظهر العدد من دون أن تلاحظ العين عملية التبديل.

إذا كانت القطعتان نمط مصعد مشترك يتم تفعيل الواحدة منهما عن طريق تطبيق 5V، وإلغاء التفعيل عن طريق 0V.

يبين الشكل (3-13) كيفية توزع الأرجل A,B,C,D,E,F,G,dp وأرجل التفعيل K1, K2.... لمجموعة سبع قطع ضوئية بخانتين وأربع خانات متوفرة بالأسواق. (من الممكن وجود توزيع آخر، ويتم الكشف وتحديد الأرجل باستخدام مقياس الآفو باختيار قياس الديود).



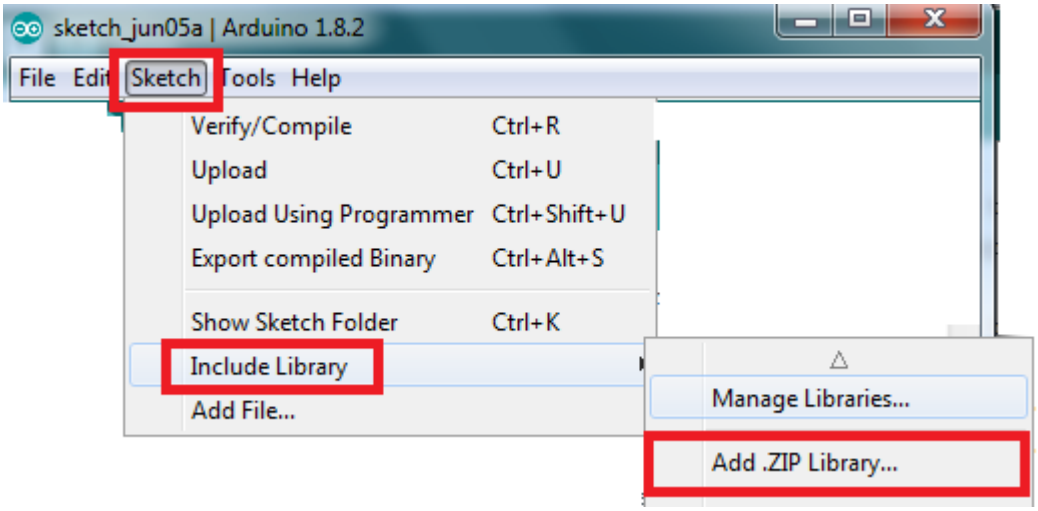
الشكل (3-13): توزع الأرجل A,B,C,D,E,F,G,dp وأرجل التفعيل 1, 2.... لمجموعة سبع قطع ضوئية بخانتين وأربع خانات متوفرة بالأسواق.

3-2-3-1-الكود البرمجي

لكتابة الكود البرمجي الخاص بإظهار عدة قطع سبع ضوئية، سنستخدم إحدى المكتبات المتوفرة على شبكة الانترنت، وهي المكتبة SevSeg الموجودة أيضاً ضمن الملفات المرفقة مع هذا الكتاب التي يمكن تحميلها من الموقع

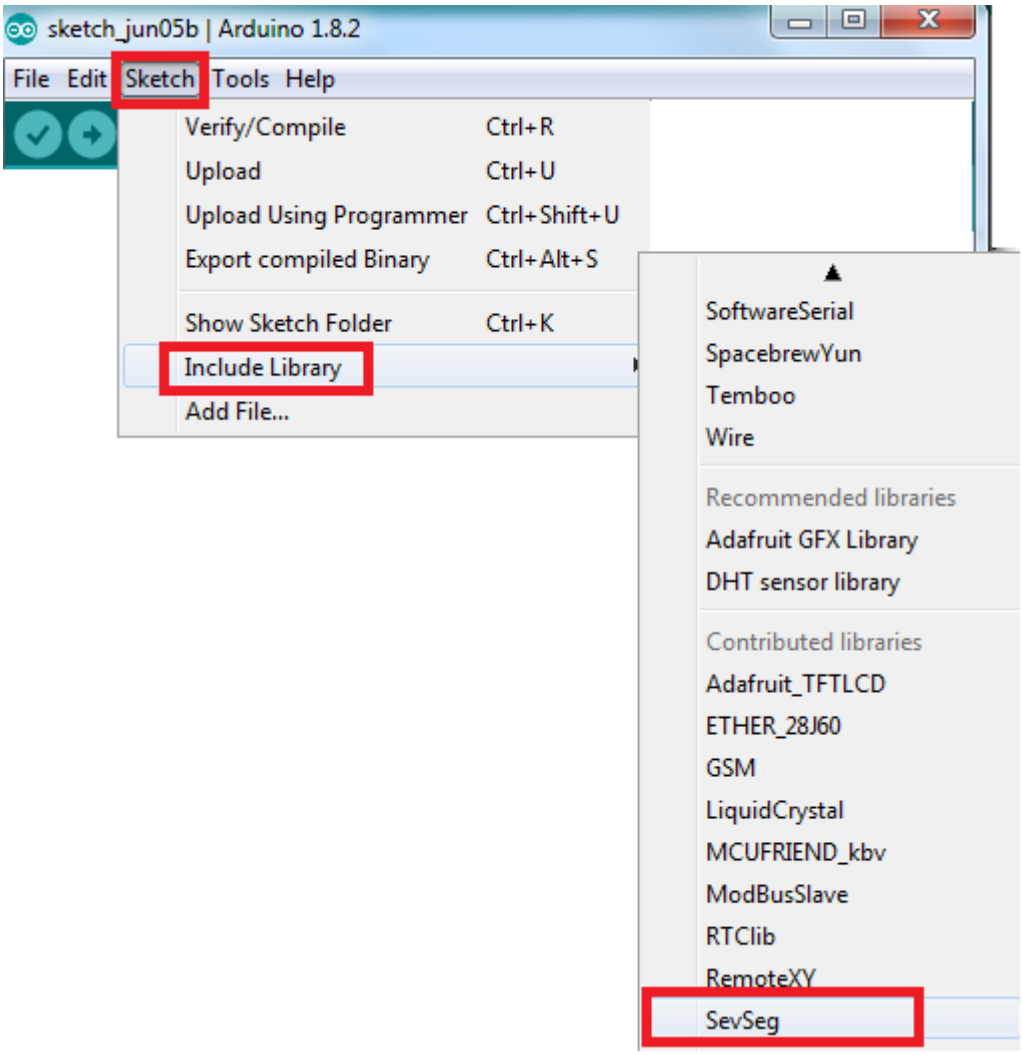
http://www.mediafire.com/file/4zo1dcc678js1ae/examples+libraries.rar/file

بعد ذلك نختار من الأعلى Sketch ثم Include Library ثم Add .ZIP Library كما هو موضح في الشكل (3-14). تظهر نافذة إضافة مكتبة، نختار الملف SevSeg.zip الذي قمنا بتحميله في الخطوة السابقة.



الشكل (3-14): إضافة مكتبة لبرنامج Arduino IDE.

بعد الانتهاء من هذه الخطوة نلاحظ أنه قد تم إضافة مكتبة SevSeg لبرنامج Arduino IDE كما هو موضح في الشكل (3-15).



الشكل (3-15): إضافة مكتبة SevSeg.

لكتابة كود برمجي باستخدام المكتبة السابقة، نقوم باتباع الخطوات التالية:

1-نقوم في البداية باستدعاء المكتبة في بداية البرنامج قبل ( ) void setup كما يلي:

```
#include <SevSeg.h>
```

2-نعرف كائن لنسميه مثلاً sevseg بعد استدعاء المكتبة مباشرةً كما يلي:

```
SevSeg sevseg;
```

3-بعد ذلك نقوم بإعداد الاتصال ما بين لوحة الأردوينو والسبع قطع المتعددة داخل () void setup كما يلي:

أ-ندخل عدد خانات مجموعة السبع قطع ضوئية. على سبيل المثال إذا كانت المجموعة أربع خانات نكتب ما يلي:

```
byte numDigits = 4;
```

ب-يتم تحديد أرجل الأردوينو التي سترتبط مع الأرجل المشتركة K1,K2,K3,K4 لمجوعة السبع قطع. على سبيل المثال إذا تم ربط الأرجل المشتركة مع الأرجل الرقمية 2, 3, 4, 5 نكتب ما يلي:

```
byte digitPins[]= {2, 3, 4, 5};
```

ج-يتم تحديد أرجل الأردوينو التي سترتبط مع الأرجل a,b,c,d,e,f,g,dp . على سبيل المثال إذا تم ربط الأرجل a,b,c,d,e,f,g,dp مع الأرجل الرقمية 6,7,8,9,10,11,12,13 للوحة الأردوينو على الترتيب، نكتب ما يلي:

```
byte segmentPins[]={6, 7, 8, 9, 10, 11, 12, 13};
```

د -يتم تحديد نوع المجموعة مهبط مشترك أم مصعد مشترك. على سبيل المثال إذا كانت المجموعة مهبط مشترك نكتب ما يلي:

```
byte hardwareConfig = COMMON_CATHODE;
```

يمكن اختيار المصعد المشترك بكتابة COMMON_ANODE

هـ- أخيراً نكتب تعليمة الإعداد لكل ما سبق كما يلي:

```
sevseg.begin(hardwareConfig, numDigits, digitPins, segmentPins);
```

4-نستخدم تعليمة ضبط السطوع (التأخير الزمني بعد إضاءة كل قطعة).

```
sevseg.setBrightness(X);
```

حيث X قيمة ما بين 0 و 100. من أجل 0 يكون التأخير الزمني يساوي 1µs، من أجل 100 يكون التأخير الزمني يساوي 2000µs.

5-لتحديد العدد المطلوب إظهاره نستخدم التعلمية:

```
sevseg.setNumber(number,decPlace);
```

حيث number هو العدد المطلوب إظهاره، و decPlace مكان الفاصلة.

```
sevseg.setNumber(3141,3);     // Displays '3.141'
sevseg.setNumber(3141,0);     // Displays '3141'
```

تدعم التعليمة إدخال عدد حقيقي floats، وعندئذ decPlace تحدد عدد الأماكن العشرية للدقة المطلوب عرضها.

```
sevseg.setNumber(3.14159f,3);  //Displays '3.141'
```

6-لتحديد مجموعة أحرف مطلوب إظهارها نستخدم التعليمة:

```
Sevseg.setChars(X);
```

حيث X مصفوفة المحارف المطلوب إظهارها بأكبر قدر ممكن من الدقة.

```
Sevseg.setChars(“abcd”);
```

7-تعليمة العرض:

```
sevseg.refreshDisplay();
```

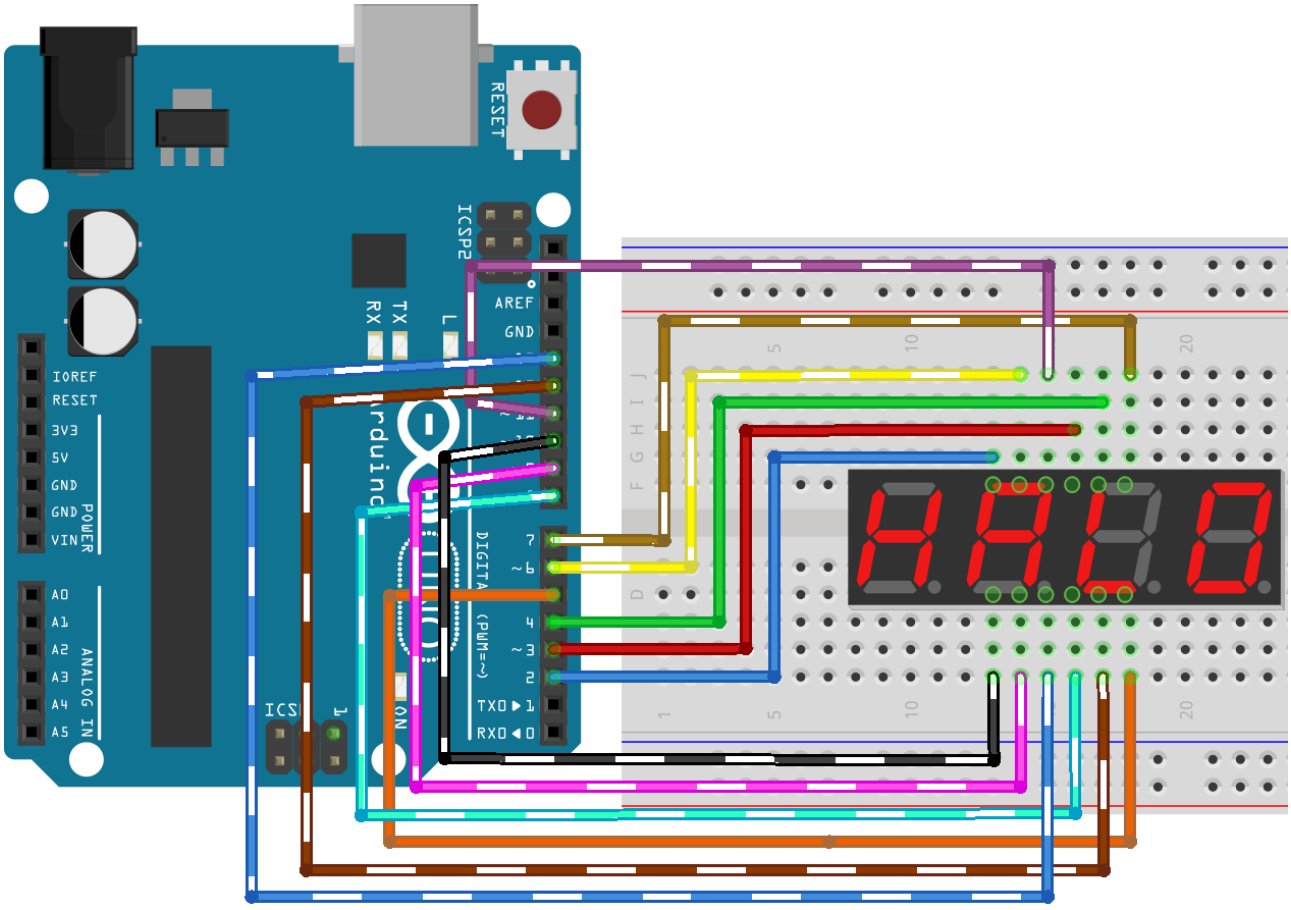
لابد من استدعاء وتنفيذ هذه التعليمة بشكل متكرر لإظهار الرقم أو الحرف الذي قمت بتحديده في التعليمتين السابقتين. أي تأخير زمني خارج هذه التعليمية سيؤدي إلى تأثيرات إظهار غير مرغوبة.

8-لمسح الإظهار نستخدم التعليمة:

```
sevseg.blank();
```

ملاحظة: إذا كان المطلوب استخدام مجموعة سبع قطع عدد خاناتها أكثر من 8، لابد من زيادة قيمة MAXNUMDIGITS في ملف SevSeg.h.

سنكتب الآن كوداً لإظهار رقم على مجموعة سبع قطع مكونة من أربع خانات. في البداية سنقوم بوصل لوحة الأردوينو مع المجموعة كما هو موضح في الشكل (3-16). استناداً لتوزيع الأرجل الموضحة في الشكل (3-13)، تم وصل الأرجل a,b,c,d,e,f,g,dp مع الأرجل الرقمية 6,7,8,9,10,11,12,13 للوحة الأردوينو على الترتيب، وتم وصل الأرجل المشتركة K1, K2, K3, K4 مع الأرجل الرقمية 2, 3, 4, 5. من الأفضل إضافة مقاومات قيمتها بحدود 200Ω إلى 500Ω ما بين الأرجل الرقمية 6,7,8,9,10,11,12,13 والأرجل a,b,c,d,e,f,g,dp.

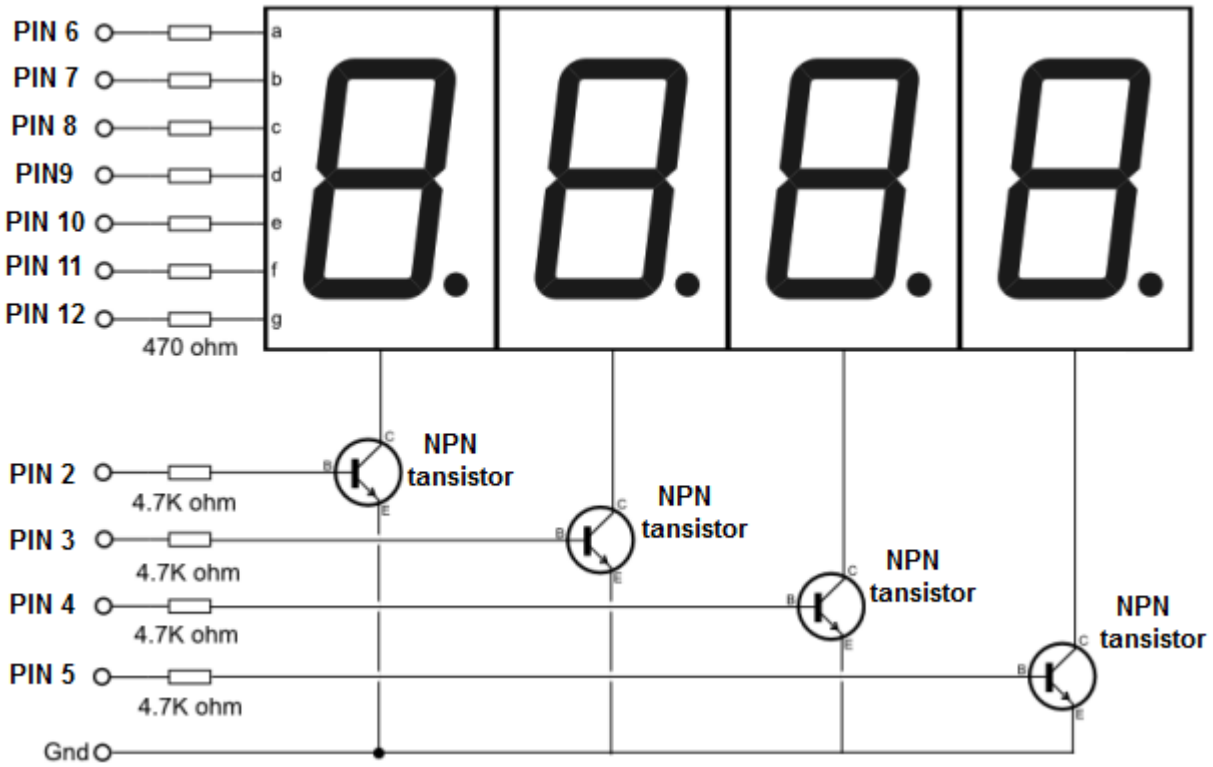


الشكل (3-16): وصل لوحة الأردوينو مع مجموعة السبع قطع ضوئية بأربع خانات.

فيما يلي الكود الخاص بإظهار العدد 3.141. يمكن تعميم هذا الكود لاحقاً لإظهار درجة الحرارة أو المسافة أو التاريخ والوقت، وغير ذلك على مجموعة السبع قطع ضوئية.

```
#include <SevSeg.h>
SevSeg sevseg;


void setup()
{
 byte numDigits = 4;
 byte digitPins[] = {2, 3, 4, 5};                    //Digits: 1,2,3,4
 byte segmentPins[] = {6, 7, 8, 9, 10, 11, 12, 13};  //Segments: A,B,C,D,E,F,G,dP
 byte hardwareConfig = COMMON_CATHODE;               // common cathode
 sevseg.begin(hardwareConfig, numDigit s, digitPins, segmentPins);
 sevseg.setBrightness(10);
 sevseg.setNumber(3141,3);
}


void loop()
{
 sevseg.refreshDisplay();
}
```

ملاحظة: عملياً من الأفضل أن لا يتم وصل أرجل لوحة السبع قطع ضوئية بشكل مباشر مع أرجل لوحة الأردوينو خصوصاً أرجل التفعيل K. وذلك لأن أرجل الأردوينو الرقمية يجب أن لا يزيد التيار المار في كل منها عن 20 mA كما تم ذكره في الفصل الأول، في حين رجل التفعيل قد يمر من خلالها تيار يزيد عن هذه القيمة خصوصاً عندما تعمل كل ثنائيات السبع قطع، لذلك يمكن وصل الدارة النموذجية الموضحة في الشكل (3-17). يعمل الترانزستور NPN كمفتاح ON/OFF الكتروني يتم التحكم به من خلال رجل الأردوينو، ويسمح بمرور التيار من خلاله بدلاً من رجل الأردوينو.

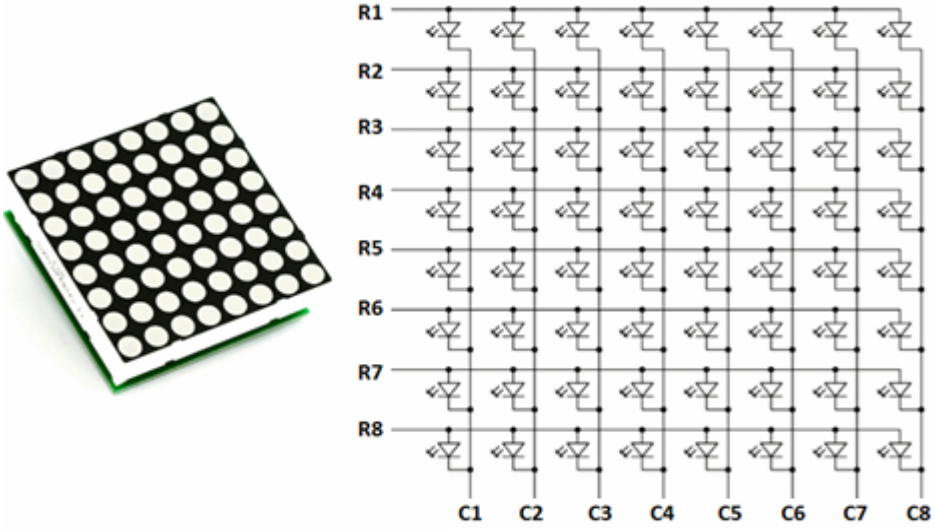


الشكل (3-17): الوصل النموذجي للوحة سبع قطع ضوئية من نمط مهبط مشترك ذات أربع خانات مع أرجل الأردوينو الرقمية.

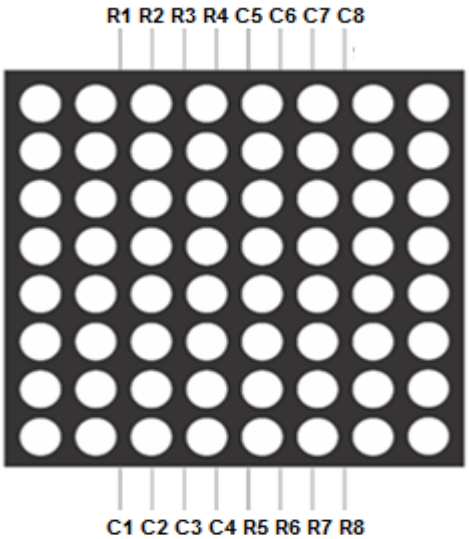
مصفوفة الثنائيات الضوئية Led Matrix

3-1-3-مقدمة

مصفوفة الثنائيات الضوئية هي عبارة عن مجموعة من الثنائيات الضوئية موزعة ضمن أسطر وأعمدة (مصفوفة)، تتصل الثنائيات الموجودة ضمن العمود الواحد بمصاعدها أو مهابطها وكذلك الأسطر. يمكن من خلال مصفوفة الثنائيات الضوئية إظهار الأرقام والأحرف أو حتى الرسم عليها، لذلك كثيراً ما نجدها تستخدم في اللوحات الإعلانية. يوضح الشكل (3-18) مصفوفة ثنائيات ضوئية 8×8 تتصل الثنائيات الموجودة في السطر الواحد بمصاعدها في حين تتصل الثنائيات الموجودة في العمود الواحد بمهابطها. عملياً تتوزع أرجل المصفوفة (16) في خلف لوحة مصفوفة الثنائيات بطرق مختلفة، يوضح الشكل (3-19) إحدى طرق توزيع الأرجل على الأعمدة والأسطر، يمكن استخدام مقياس الآفو لمعرفة التوزيع.



الشكل (3-18): مصفوفة ثنائيات ضوئية 8×8.

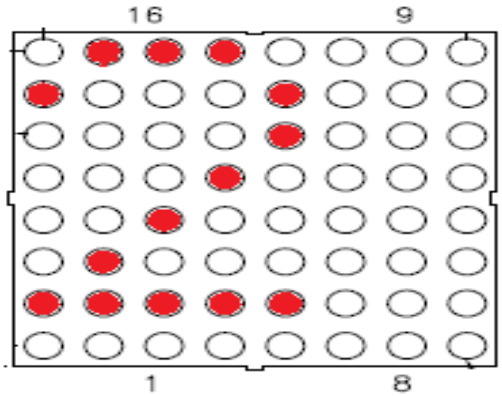


الشكل (3-19): إحدى طرق توزيع أرجل مصفوفة ثنائيات على الأعمدة (يرمز لها بحرف C)، وعلى الأسطر (يرمز لها بحرف R).

3-3-2-ربط مصفوفة الثنائيات الضوئية مع لوحة الأردوينو أونو

قد يتضح لدينا أنه نحتاج لمصفوفة LED 8×8=64 إلى 64 رجل من لوحة الأردوينو لقيادة وتشغيل المصفوفة، ولكن باستخدام طريقة تعرف بالتجميع multiplexing يمكن تقليل هذا العدد بشكل كبير. تشبه عملية قيادة مصفوفة الثنائيات الضوئية بالتجميع مسح مجموعة سبع قطع ضوئية المذكورة في الفقرة (3-2-3). في السبع قطع ضوئية يتم تفعيل إحدى السبع قطع ومن ثم إرسال القيمة المراد إظهارها عليها وتكرر العملية للبقية وهكذا، في مصفوفة الثنائيات الضوئية يتم تفعيل أحد الأعمدة ومن ثم إرسال القيمة المراد إظهارها على ذلك العمود وتكرر العملية لبقية الأعمدة وهكذا. نتيجة لذلك فإن الثنائيات الضوئية في العمود الواحد ستعمل لفترة زمنية، وستتوقف لفترة زمنية أخرى، وبسبب كون عملية المسح سريعة فإن العين ستلاحظ أن ثنائيات ذلك العمود بحالة عمل ON وهذه هي فكرة طريقة التجميع. باعتبار أن الثنائيات الموجودة في السطر الواحد تتصل بمصاعدها في حين تتصل الثنائيات الموجودة في العمود الواحد بمهابطها، يتم تفعيل العمود بوضع صفر منطقي عليه، وبقية الأعمدة 5V.

لتوضيح الفكرة المذكورة لنفرض أننا أردنا إظهار الشكل (3-20). إذا قمنا بتفعيل العمود الأول من اليسار، عندئذٍ لابد من إطفاء ليدات السطر الأول والثالث والرابع والخامس والسادس والثامن، وتشغيل ليدات السطر الثاني والسابع. أي أننا سنرسل على الأسطر من 1 وحتى 8 القيمة 0,1,0,0,0,0,1,0 على الترتيب. تكرر نفس الفكرة لبقية الأعمدة. يلخص الجدول (3-2) القيم المطلوب تطبيقها على الأسطر مع كل مرة يتم فيها تفعيل عمود من الأعمدة.



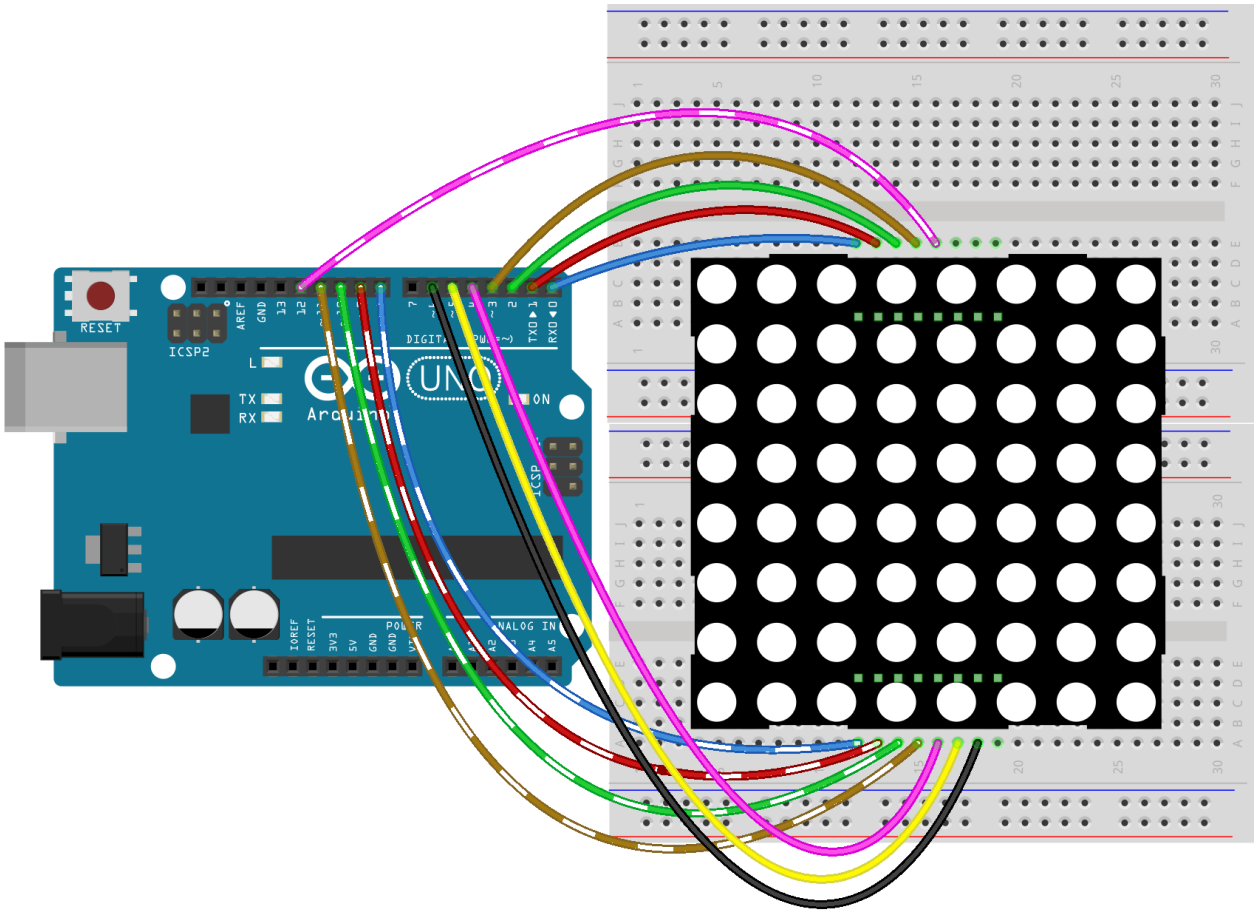
الشكل (3-20): مثال على إظهار شكل على مصفوفة الثنائيات.

الجدول(3-2): القيم المطلوب تطبيقها على الأسطر مع تفعيل كل عمود.

العمود المفعل من اليسار / أرقام الأسطر	1	2	3	4	5	6	7	8
السطر 1	0	1	1	1	0	0	0	0
السطر 2	1	0	0	0	1	0	0	0
السطر 3	0	0	0	0	1	0	0	0
السطر 4	0	0	0	1	0	0	0	0
السطر 5	0	0	1	0	0	0	0	0
السطر 6	0	1	0	0	0	0	0	0
السطر 7	1	1	1	1	1	0	0	0
السطر 8	0	0	0	0	0	0	0	0

يمكن تطبيق القيم السابقة بوصل الأسطر والأعمدة إلى منافذ الخرج للوحة الأردوينو، أي أننا سنحتاج إلى 16 منفذ من اللوحة لقيادة المصفوفة LED 64=8×8. يمكن تقليل عدد الأرجل المطلوبة لقيادة المصفوفة باستخدام بعض الدارات المتكاملة IC مثل مسجلات الإزاحة وبعض مفككات الترميز. وهذا ما سندعه للمستوى الثاني عندما سيتم قيادة مصفوفة ثنائيات كبيرة.

سنستخدم جزءاً من المصفوفة 8×8 يساوي 7 أسطر و5 أعمدة (أي مصفوفة 5×7 بدلاً من 8×8). تم وصل أسطر المصفوفة السبعة مع الأرجل من 0 وحتى 6، والأعمدة الخمسة تم وصلها مع الأرجل من 8 وحتى 12 كما هو موضح في الشكل (3-21).



الشكل (3-21): وصل لوحة الأردوينو مع مصفوفة 5×7 بدلاً من 8×8.

لنفرض أننا أردنا إظهار الرقم 2، خطوات كتابة البرنامج ستكون على الشكل التالي

1- نفعل العمود الأول من خلال وضع صفر منطقي على الرجل 8.

2- يتم إرسال القيمة المراد إظهارها على العمود الأول وهي تشغيل الثنائي في السطر الثاني و الثنائي في السطر السابع و بالتالي إخراج 5V+ على الأرجل 1 و 6 وبقية الأرجل 0V . أي أن القيمة التي لابد من إخراجها على الأرجل من 0 وحتى 6 هي 0x42 = B1000010. تستمر العملية لفترة بسيطة لتكن 1ms.

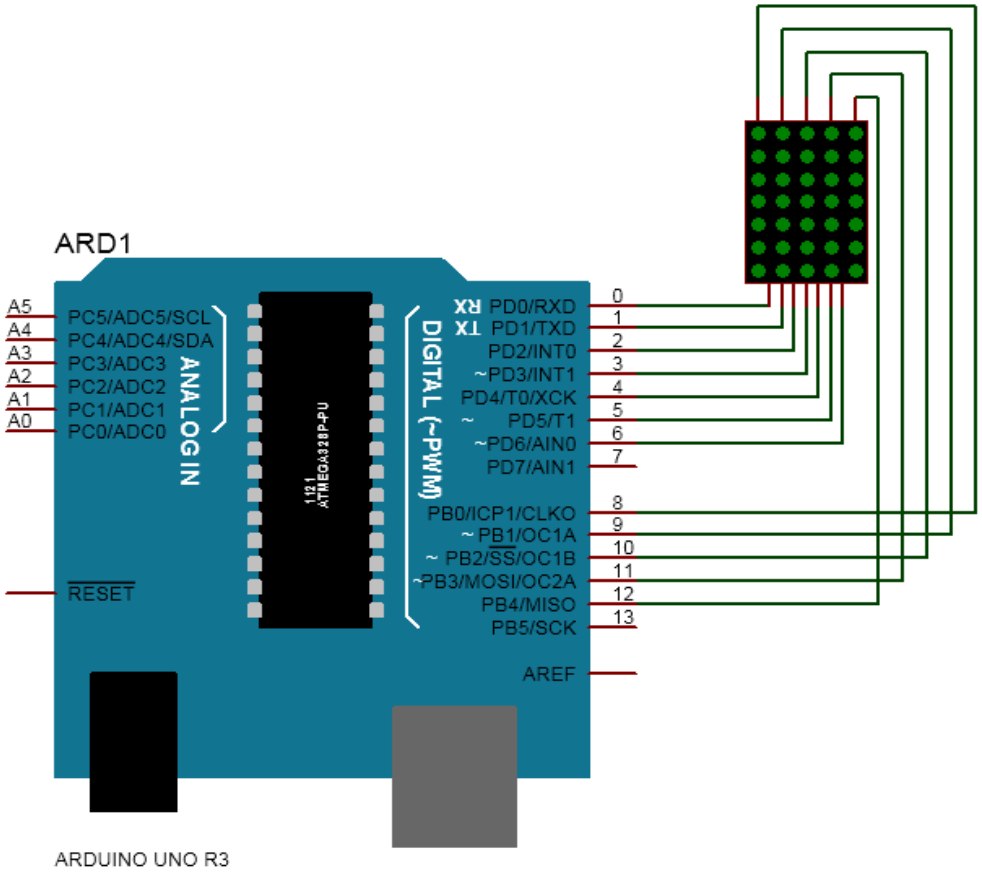
3- يتم إلغاء تفعيل العمود الأول من خلال وضع واحد منطقي على الرجل 8.

4- نكرر العمليات السابقة لكل عمود و سطر ليظهر الشكل السابق.

3-3-3-الكود البرمجي

```
void setup() {
   byte i;
   for (i = 0; i <= 13; i++)    // تفعيل الأرجل من 0 وحتى 13 كمخارج
   {
    pinMode(i,OUTPUT);
   }
   for (i = 8; i <= 12; i++)  // إخراج 1 منطقي على الأرجل من 8 وحتى 12 أي على أعمدة المصفوفة
   {
    digitalWrite(i,HIGH);
   }
 }


 void loop()
 {
digitalWrite(8,LOW); PORTA(0x42); delay(1); digitalWrite(8,HIGH); PORTA(0); //العمود الأول
digitalWrite(9,LOW); PORTA(0x61); delay(1); digitalWrite(9, HIGH); PORTA(0); // العمود الثاني
digitalWrite(10, LOW); PORTA(0x51); delay(1); digitalWrite(10, HIGH); PORTA(0); //العمود الثالث
digitalWrite(11, LOW); PORTA(0x49); delay(1); digitalWrite(11, HIGH); PORTA(0); //العمود الرابع
digitalWrite(12, LOW); PORTA(0x46); delay(1); digitalWrite(12, HIGH);PORTA(0); // العمود الخامس
 }


 void PORTA(byte value)          // برنامج فرعي لإخراج قيمة على الأرجل من 0 وحتى 6
 {
   byte i;
   for (i = 0; i <= 7; i++)
   {
    digitalWrite(i, bitRead(value, i));
   }
}
```

3-3-4-محاكاة مصفوفة الثنائيات الضوئية من خلال برنامج Proteus



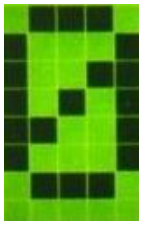
الشكل (3-22): محاكاة دارة مصفوفة الثنائيات الضوئية من خلال برنامج Proteus.

شاشة الكريستال السائلة LCD

3-4-1-مقدمة

تستخدم شاشة الكريستال السائلة Liquid Crystal Display(LCD) كأداة لإظهار الحروف والأرقام المختلفة تبعاً للأوامر التي يرسلها المتحكم أو لوحة الأردوينو للشاشة. سميت الشاشة بذلك لاحتوائها على قطع كريستال معلقة ضمن سائل لزج. تتواجد شاشة LCD بأشكال متنوعة فقد تكون مؤلفة من سطر أو عدة أسطر. يحتوي كل سطر على عدد من الخانات، والخانة عبارة عن مصفوفة ذات بعدين من البكسيلات pixels يتم من خلالها إظهار الحرف المطلوب كما هو موضح في الشكل (3-23). أكثر الشاشات شيوعاً:1×16, 1×20, 1×24, 2×16, 2×20, 4×16, 4×20 .

تمتلك الشاشة معالج إظهار خاص بها وذاكرة داخلية خاصة تقسم إلى قسمين: ذاكرة المعطيات DD_RAM تسمح بالاحتفاظ بالحروف المرسلة دون الحاجة إلى إرسالها كل مرة، وذاكرة مولد الرموز CG_RAM التي تحتفظ بأشكال مجموعة من الرموز والمحارف وهذا ما يسمح بإظهار الحرف المطلوب بإرسال ترميزة أسكي لهذا الحرف للشاشة.



الشكل (3-23): شاشة الكريستال السائل 2×16. كل خانة عبارة عن مصفوفة من البكسيلات ببعدين.

تتألف الشاشة من 16 رجل مرقمة كما هو موضح بالجدول (3-3):

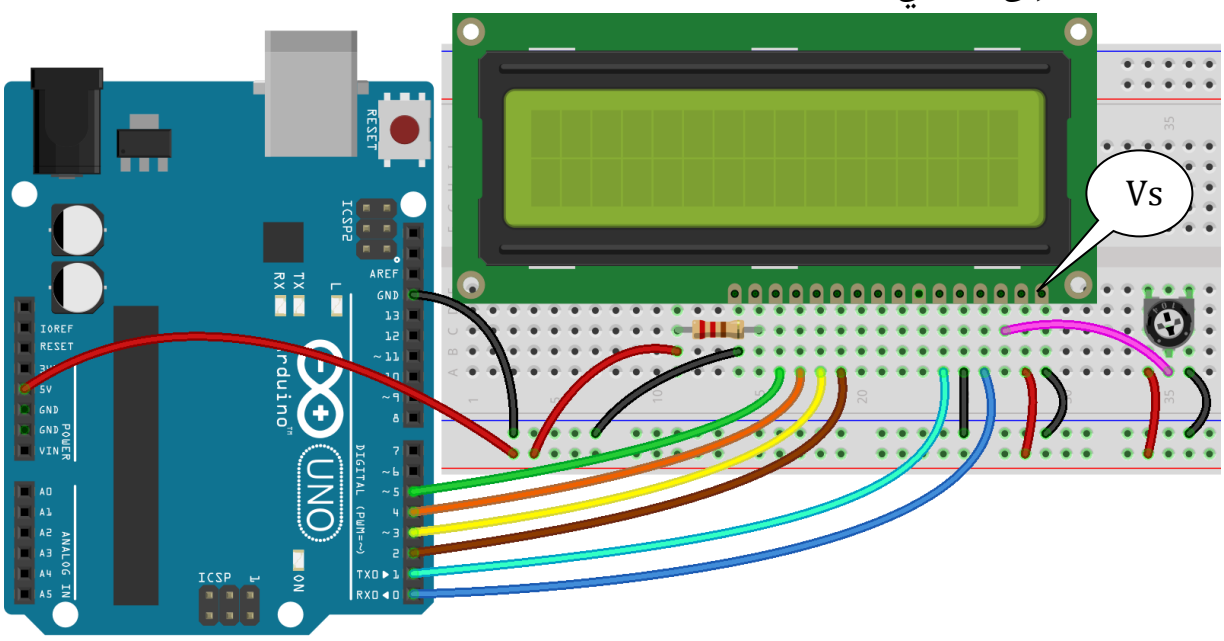
الجدول (3-3): أرجل شاشة الكريستال السائل.

الرقم	الرمز	الوظيفة
1	Vss	أرضي (0 V).
2	Vdd	التغذية (5 V).
3	Vee	التحكم بشدة التباين contrast. يطبق على هذه الرجل جهد يتغير بين 0 وحتى 5V. عند ما يكون الجهد 0V يكون لون الحروف أسود داكن، كلما ازداد الجهد يقل اللون الأسود (يزداد التباين).
4	RS	اختيار السجل أو نمط العمل : 0V : تعليمة أو أمر ، +5V : حرف أو معطيات
5	R/W	اختيار القراءة من الشاشة أو الكتابة إلى الشاشة: 0V : وضع الكتابة ، +5V : وضع القراءة
6	E	تمكين وتنفيذ الأمر المطبق على خطوط المعطيات: عند الكتابة: ينفذ الأمر عند الانتقال من المنطق المرتفع إلى المنطق المنخفض. عند القراءة: ينفذ الأمر عند الانتقال من المنطق المنخفض إلى المنطق المرتفع.

الرقم	الرمز	الوظيفة
7	D0	خطوط معطيات، يتم من خلالها إرسال شيفرات الحروف والأوامر
8	D1	
9	D2	
10	D3	
11	D4	
12	D5	
13	D6	
14	D7	
15	A	قطب موجب للإضاءة (5V).
16	K	قطب سالب للإضاءة (0V).

3-4-2-ربط شاشة الكريستال السائل مع لوحة الأردوينو أونو

يوضح الشكل (3-24) كيفية وصل شاشة LCD مع لوحة الأردوينو أونو. تم وصل الرجل RS مع المنفذ 0 للوحة الأردوينو، والرجل E مع المنفذ 1، وأرجل المعطيات D4,D5,D6,D7 مع المنافذ 2,3,4,5. لا يتم وصل أرجل المعطيات D0,D1,D2,D3 لأن الشاشة سيتم التعامل معها وفق نمط 4 بت. يمكن استخدام أرجل رقمية أخرى للأردوينو بنفس الطريقة. تم وصل الرجل Vee مع مقاومة متغيرة لتطبيق جهد متغير وبالتالي التحكم بالتباين كما هو موضح في الشكل. تم وصل الرجل WR مع الأرضي. تم وصل رجل الإضاءة الموجب A مع التغذية 5V عن طريق مقاومة 220Ω، ورجل الإضاءة السالب إلى الأرضي.



الشكل (3-24): وصل لوحة الأردوينو مع شاشة LCD وفق نمط 4 بت.

3-4-3- الكود البرمجي

يتم برمجة الشاشة من خلال استدعاء المكتبة LiquidCrystal.h، وتطبيق التعليمات في الجدول (3-4).

الجدول (3-4): تعليمات المستخدمة لبرمجة شاشة LCD.

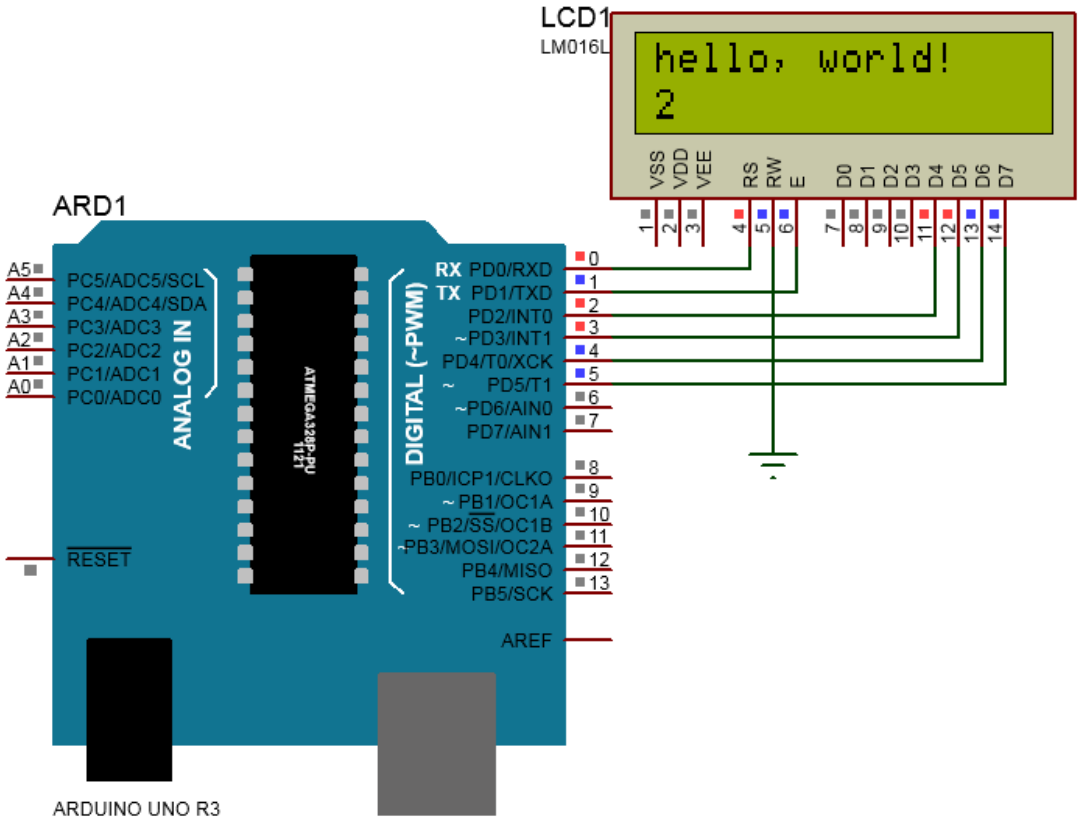
التعليمة	شرح التعليمة
LiquidCrystal lcd(rs, enable, d4, d5, d6, d7)	يتم وضع هذه التعليمة بعد استدعاء مكتبة <LiquidCrystal.h> مباشرةً. يتم من خلالها تعريف كائن اسمه lcd، مع تحديد أرجل لوحة الأردوينو التي ستتصل مع أرجل الشاشة RS، E، d4، d5، d6، d7 . التعليمة وفق هذه الشكل ستجعل الشاشة تعمل وفق نمط 4 أرجل فقط (d7,d6,d5,4). يمكن كتابتها بشكل آخر للعمل بنمط 8 أرجل وعندئذ سيتم حجز أرجل أكثر للوحة الأردوينو. رجل RW يتم وصلها مع الأرضي ولا تدخل في هذه التعليمة. مثال، بفرض أن دارة ربط الشاشة مع الأردوينو كما في الشكل (3-23) عندئذ يتم كتابة التعليمة على الشكل التالي: LiquidCrystal lcd(0,1,2,3,4,5)
lcd.begin(cols, rows)	تكتب هذه التعليمة داخل ( ) void setup. تعمل على تهيئة الاتصال مع الشاشة وتحديد الأبعاد: عدد الأعمدة cols، وعدد الأسطر rows. مثال: بفرض أن الشاشة عبارة عن سطرين و16 خانة، عندئذ يتم كتابة التعليمة على الشكل التالي: lcd.begin(16, 2)
lcd.clear()	يتم مسح الشاشة مع إعادة المشيرة إلى الزاوية العلوية اليسرى.
lcd.home()	يتم إعادة المشيرة إلى الزاوية العلوية اليسرى من دون مسح الشاشة.
lcd.setCursor(col، row)	تحريك المشيرة إلى العمود col، والسطر row. حيث الزاوية العلوية اليسرى إحداثياتها (0,0). مثال: بفرض أننا أردنا تحريك المشيرة إلى بداية السطر الثاني، عندئذ يتم كتابة التعليمة على الشكل التالي: lcd.setCursor(0، 1)
lcd.write(data)	كتابة حرف على الشاشة. حيث data الحرف الذي سيظهر على الشاشة.
lcd.print(data)	كتابة نص على الشاشة. حيث data النص الذي سيظهر على الشاشة. البارامتر data قد يكون char، byte، int، long، string . مثال: بفرض أننا أردنا إظهار كلمة Hello، عندئذ يتم كتابة التعليمة على الشكل التالي: lcd.print(“Hello”)

التعليمة	شرح التعليمة
lcd.cursor()	إظهار المشيرة على شكل خط سفلي في المكان الذي سيتم كتابة الحرف التالي.
lcd.noCursor()	إخفاء المشيرة.
lcd.blink()	إظهار المشيرة على شكل وميض.
lcd.noBlink()	إيقاف وميض المشيرة.
lcd.scrollDisplayLeft()	تحريك محتويات الشاشة (النص والمشيرة) مسافة واحدة لليسار.
lcd.scrollDisplayRight()	تحريك محتويات الشاشة (النص والمشيرة) مسافة واحدة لليمين.
lcd.createChar(num، data)	تسمح هذه التعليمة من إنشاء رمز خاص. كل رمز 5x8 pixels، ويتم إدخاله على شكل مصفوفة بـ 8 بايت. البتات الخمسة الأولى من كل بايت تحدد بكسيلات السطر. البارامتر num هو رقم الرمز (من 0 وحتى 7)، والبارامتر data مصفوفة الرمز. تستخدم التعليمة write() لإظهار الرمز. يمكن الاطلاع على reference الخاص بمكتبة الشاشة لمعرفة كيفية تطبيق هذه التعليمة.

فيما يلي الكود البرمجي لإظهار عبارة Hello, world، بالإضافة لعداد يزداد كل ثانية.

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(0, 1, 2, 3, 4, 5);


void setup() {
  lcd.begin(16, 2);
  lcd.print("hello, world!");
}


void loop() {
  unsigned long x;
  lcd.setCursor(0, 1);
  x = millis() / 1000 ;        // the number of seconds since reset:
  lcd.print(x);
}
```

3-4-4-محاكاة شاشة الكريستال السائل من خلال برنامج Proteus

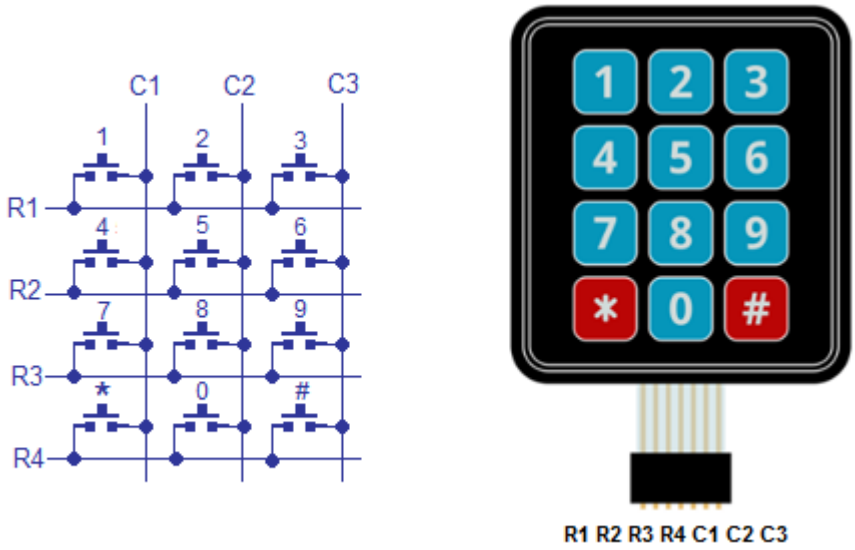


الشكل (3-25): محاكاة ربط شاشة LCD مع لوحة الأردوينو من خلال برنامج Proteus.

لوحة المفاتيح Keypad

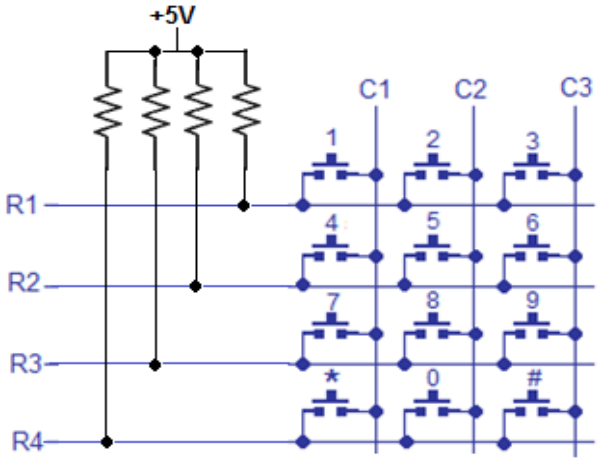
3-5-1-مقدمة

تستخدم لوحة المفاتيح keypad كأداة إدخال للأردوينو. تتكون لوحة المفاتيح من مجموعة من الأزرار push button تكون مرتبة على شكل مصفوفة. يوضح الشكل (3-26) لوحة مفاتيح بـ 4 أسطر، و3 أعمدة. تتصل مفاتيح كل سطر معاً بسلك مشترك، وكذلك كل عمود. لذلك ينتج لدينا لوحة المفاتيح في الشكل (3-26) 4 أسلاك مشتركة لكل سطر R1, R2, R3, R4، و3 أسلاك مشتركة لكل عمود C1, C2, C3. ما يميز لوحة المفاتيح هو أنه عند الضغط على أحد الأزرار سيتقاطع (يتصل) سلك السطر الذي يتواجد فيه الزر مع سلك العمود. مثلاً عند الضغط على الزر 1 يتصل السلك R1 مع السلك C1 وهكذا.



الشكل (3-26): لوحة مفاتيح 3×4.

يتم الكشف عن زر مضغوط للوحة المفاتيح والتعرف عليه من قبل لوحة الأردوينو من خلال طريقة تعرف بمسح الأعمدة. تعتمد هذه الطريقة بأن يتم ربط أسطر لوحة المفاتيح مع 4 مقاومات سحب pull-up إلى +5V كما هو موضح في الشكل (3-27).



الشكل (3-27): كيفية إعداد لوحة المفاتيح لمعرفة الزر المضغوط.

يتم ربط الأعمدة مع 3 مخارج رقمية للوحة الأردوينو، ويتم ربط الأسطر مع 4 مداخل رقمية بعد ذلك يتم تنفيذ الخطوات الثلاثة التالية:

1- نطبق على العمود الأول 0 منطقي (C1=0) وبقية الأعمدة 1 منطقي (C2=1, C3=1) من خلال المخارج الثلاث الرقمية. القيمة الناتجة عن الأسطر ستدل على الزر المضغوط من العمود الأول كما هو موضح في الجدول (3-5).

الجدول (3-5): مسح العمود الأول.

العملية	حالة الأسطر			
	R4	R3	R2	R1
الضغط على الزر 1	1	1	1	0
الضغط على الزر 4	1	1	0	1
الضغط على الزر 7	1	0	1	1
الضغط على الزر *	0	1	1	1
عدم الضغط على أي زر أو أحد الأزرار الأخرى	1	1	1	1

2- نطبق على العمود الثاني 0 منطقي (C2=0) وبقية الأعمدة 1 منطقي (C1=1, C3=1). القيمة الناتجة عن الأسطر ستدل على الزر المضغوط من العمود الثاني كما هو موضح في الجدول (3-6).

الجدول (3-6): مسح العمود الثاني.

العملية	حالة الأسطر			
	R4	R3	R2	R1
الضغط على الزر 2	1	1	1	0
الضغط على الزر 5	1	1	0	1
الضغط على الزر 8	1	0	1	1
الضغط على الزر 0	0	1	1	1
عدم الضغط على أي زر أو أحد الأزرار الأخرى	1	1	1	1

3- نطبق على العمود الثالث 0 منطقي (C3=0) وبقية الأعمدة 1 منطقي (C1=1, C2=1). القيمة الناتجة عن الأسطر ستدل على الزر المضغوط من العمود الثالث كما هو موضح في الجدول (3-7).

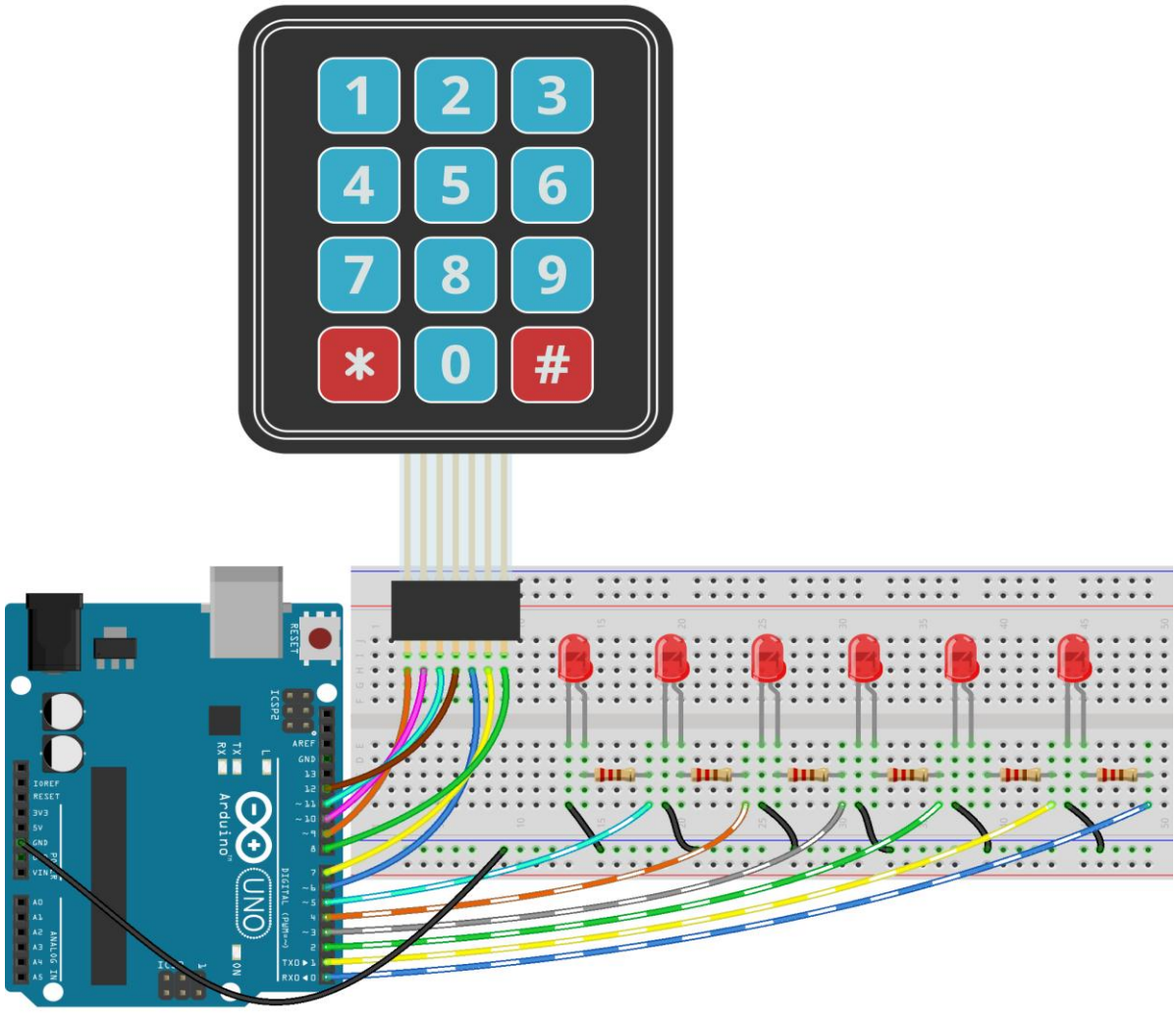
الجدول (3-7): مسح العمود الثالث.

العملية	حالة الأسطر			
	R4	R3	R2	R1
الضغط على الزر 3	1	1	1	0
الضغط على الزر 6	1	1	0	1
الضغط على الزر 9	1	0	1	1
الضغط على الزر #	0	1	1	1
عدم الضغط على أي زر أو أحد الأزرار الأخرى	1	1	1	1

3-5-2-ربط لوحة المفاتيح مع لوحة الأردوينو

يتم ربط أرجل لوحة المفاتيح مع الأرجل الرقمية للوحة الأردوينو. يوضح الشكل (3-28) مثالاً على كيفية ربط لوحة المفاتيح مع الأردوينو أونو. تم ربط السطر الأول مع الرجل 9، والسطر الثاني مع الرجل 10، والسطر الثالث مع الرجل 11، والسطر الرابع مع الرجل 12. تم ربط العمود الأول مع الرجل 6، والعمود الثاني مع الرجل 7، والعمود الثالث مع الرجل 8. تتميز أرجل لوحة الأردوينو بإمكانية تفعيل مقاومات سحب داخلية، بالتالي عدم الحاجة لإضافتها خارجياً، ليتم وصل أسطر المصفوفة مباشرةً مع أرجل الأردوينو (9,10,11,12). يمكن استخدام أي أرجل رقمية أخرى.

لاختبار وتجربة عمل لوحة المفاتيح تم ربط ثنائيات ضوئية على الأرجل من 0 وحتى 5 لاختبار عمل لوحة المفاتيح. سنكتب برنامجاً فيما بعد يعمل على مسح لوحة المفاتيح، ويعيد الزر المكبوس. إذا كان الزر المكبوس 1 يتم تشغيل الثنائي الضوئي الأول، وإذا كان 2 يتم إطفاء الثنائي الأول، وإذا كان الزر المكبوس 3 يتم تشغيل الثنائي الضوئي الثاني، وإذا كان 4 يتم إطفاء الثنائي الثاني وهكذا لبقية الأزرار.



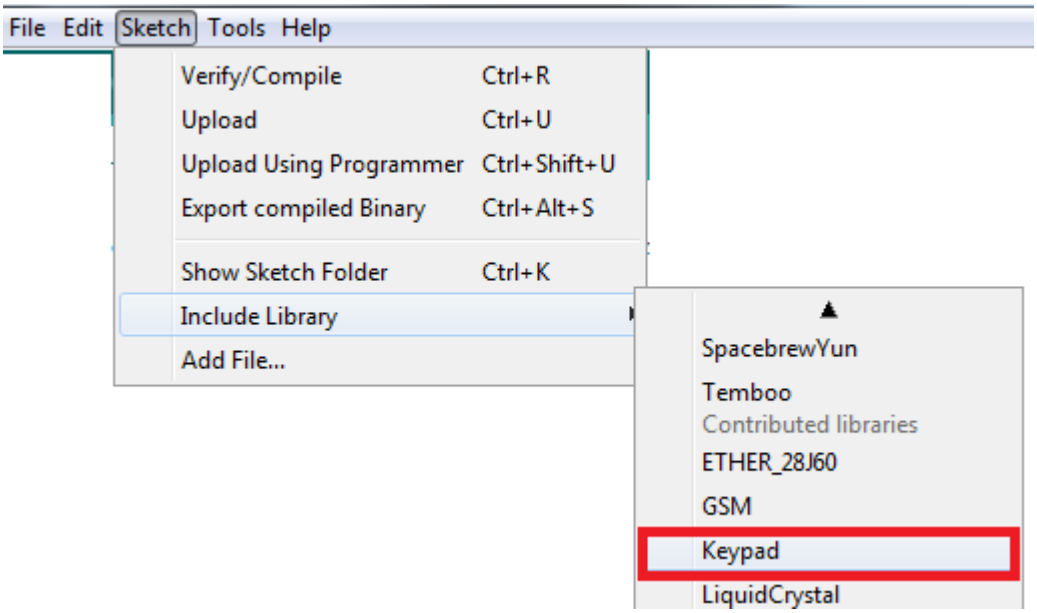
الشكل (3-28): ربط لوحة المفاتيح مع الأردوينو.

3-5-3-الكود البرمجي

لكتابة الكود البرمجي الخاص بلوحة المفاتيح، سنستخدم إحدى المكتبات المتوفرة على شبكة الانترنت، وهي المكتبة Keypad الموجودة ضمن الملفات المرفقة مع هذا الكتاب التي يمكن تحميلها من الموقع

http://www.mediafire.com/file/4zo1dcc678js1ae/examples+libraries.rar/file

2-نختار من الأعلى Sketch ثم Include Library ثم Add .ZIP Library كما هو موضح في الشكل (3-14) سابقاً. تظهر نافذة إضافة مكتبة، نختار الملف keypad.zip الذي قمنا بتحميله في الخطوة السابقة. بعد الانتهاء من هذه الخطوة نلاحظ أنه قد تم إضافة مكتبة keypad لبرنامج Arduino IDE كما هو موضح في الشكل (3-29).



الشكل (3-29): إضافة مكتبة keypad.

3-نقوم باستدعاء المكتبة كما يلي:

```
#include <keypad.h>
```

4-نقوم بتعريف كائن، مع إعداد للاتصال ما بين لوحة المفاتيح ولوحة الأردوينو كما يلي:

أ-ندخل عدد الأسطر، وعدد الأعمدة كما يلي:

```
const byte ROWS = 4;    // أربع أسطر
const byte COLS = 3;    // ثلاث أعمدة
```

ب-ندخل خريطة أزرار لوحة المفاتيح كما يلي من أجل لوحة مفاتيح أربعة أسطر، وثلاثة أعمدة

```
char keys[ROWS][COLS] = {
{'1','2','3'},
{'4','5','6'},
{'7','8','9'},
{'*','0','#'}
};
```

من أجل لوحة مفاتيح أربعة أسطر، وأربعة أعمدة

```
char keys [ROWS][COLS] = {
{'0','1','2','3'},
{'4','5','6','7'},
{'8','9','A','B'},
{'C','D','E','F'}
};
```

ج-ندخل أرجل الأردوينو التي سيتم ربطها مع أسطر لوحة المفاتيح، ابتداءً من السطر الأول. على سبيل المثال إذا تم تطبيق الشكل (3-27) تكون التعليمة على الشكل التالي:

```
byte rowPins[ROWS] = {9, 10, 11, 12};
```

أيضاً ندخل أرجل الأردوينو التي سيتم ربطها مع أعمدة لوحة المفاتيح، ابتداءً من العمود الأول. على سبيل المثال إذا تم تطبيق الشكل (3-27) تكون التعليمة على الشكل التالي:

```
byte colPins[COLS] = {6, 7, 8};
```

د-أخيراً ننشئ الكائن (لنسميه مثلاً mykeypad) كما يلي:

```
Keypad mykeypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS);
```

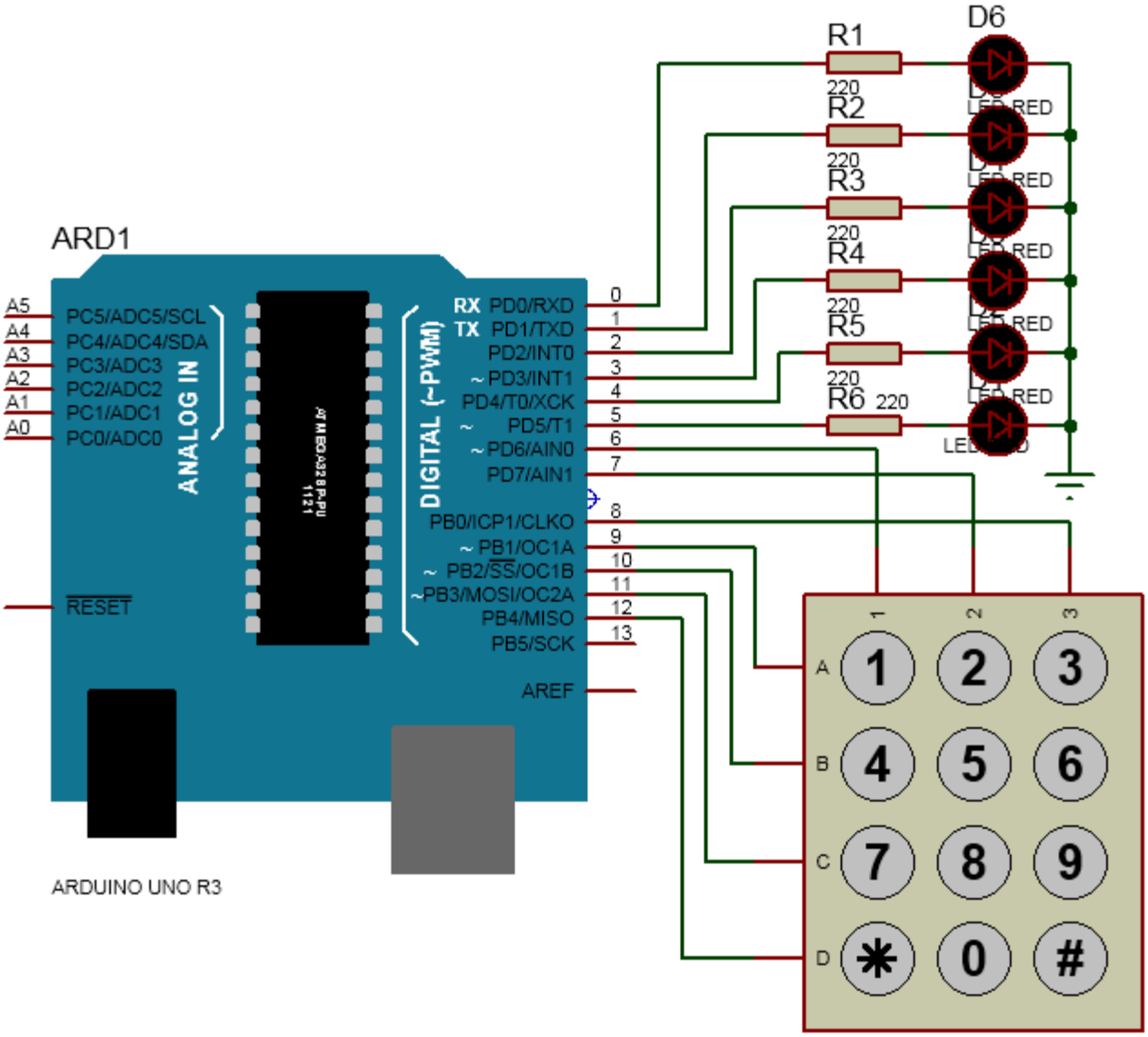
5- بعد إنشاء الكائي keypad يمكننا استخدام التعليمات التالية:

التعليمة	شرح التعلمية
mykeypad.waitForKey( )	تنتظر هذه التعليمية أن يقوم المستخدم بالضغط على أحد أزرار لوحة المفاتيح، لتعيد عندئذ الزر المضغوط تبعاً للخريطة المذكورة في الخطوة 3. لابد من الانتباه إلى أن هذه التعليمة تسبب توقف تنفيذ التعليمات الأخرى حتى يتم الضغط على أحد المفاتيح. مثال: char key; key= keypad.waitForKey( );
mykeypad.getKey( )	تعيد هذه التعليمة الزر المضغوط في حالة الضغط على أحد الأزرار، ولا تسبب توقف البرنامج كما في التعليمة السابقة. مثال: char key; key= keypad.getKey( );
mykeypad.getState( )	تعيد هذه التعليمة حالة لوحة المفاتيح الحالية. أربع حالات هي: IDLE, PRESSED, RELEASED and HOLD
mykeypad.setHoldTime (unsigned int time)	تحدد القيمة الزمنية بالميلي ثانية التي سيحتاجها المستخدم للضغط على الزر حتى يتم تفعيل حالة المسك HOLD. مثال: mykeypad.setHoldTime(500);
mykeypad.setDebounceTime (unsigned int time)	تحدد القيمة الزمنية بالميلي ثانية التي سوف تنتظر لوحة المفاتيح حتى تقبل الضغط على مفتاح جديد. مثال: mykeypad.setDebounceTime(20);

فيما يلي الكود البرمجي الخاص بمسح لوحة المفاتيح ومعرفة الزر المكبوس، وتشغيل وإطفاء الثنائيات الضوئية الموضحة في الشكل (3-27).

```
#include <Keypad.h>
const byte ROWS = 4; //four rows
const byte COLS = 3; //three columns
char keys[ROWS][COLS] = {
  {'1','2','3'},
  {'4','5','6'},
  {'7','8','9'},
  {'*','0','#'}
};
byte rowPins[ROWS] = {9, 10, 11, 12};//connect to the row pinouts of the keypad
byte colPins[COLS] = {6, 7, 8}; //connect to the column pinouts of the keypad

Keypad mykeypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );

void setup(){
byte i;
for (i=0;i<=5;i++)
{pinMode(i,OUTPUT);}
}

void loop() {
char x;
x=mykeypad.waitForKey();
if (x=='1') {digitalWrite(0,HIGH);}
if (x=='2') {digitalWrite(0,LOW);}
if (x=='3') {digitalWrite(1,HIGH);}
if (x=='4') {digitalWrite(1,LOW);}
if (x=='5') {digitalWrite(2,HIGH);}
if (x=='6') {digitalWrite(2,LOW);}
if (x=='7') {digitalWrite(3,HIGH);}
if (x=='8') {digitalWrite(3,LOW);}
if (x=='9') {digitalWrite(4,HIGH);}
if (x=='*') {digitalWrite(4,LOW);}
if (x=='0') {digitalWrite(5,HIGH);}
if (x=='#') {digitalWrite(5,LOW);}
}
```

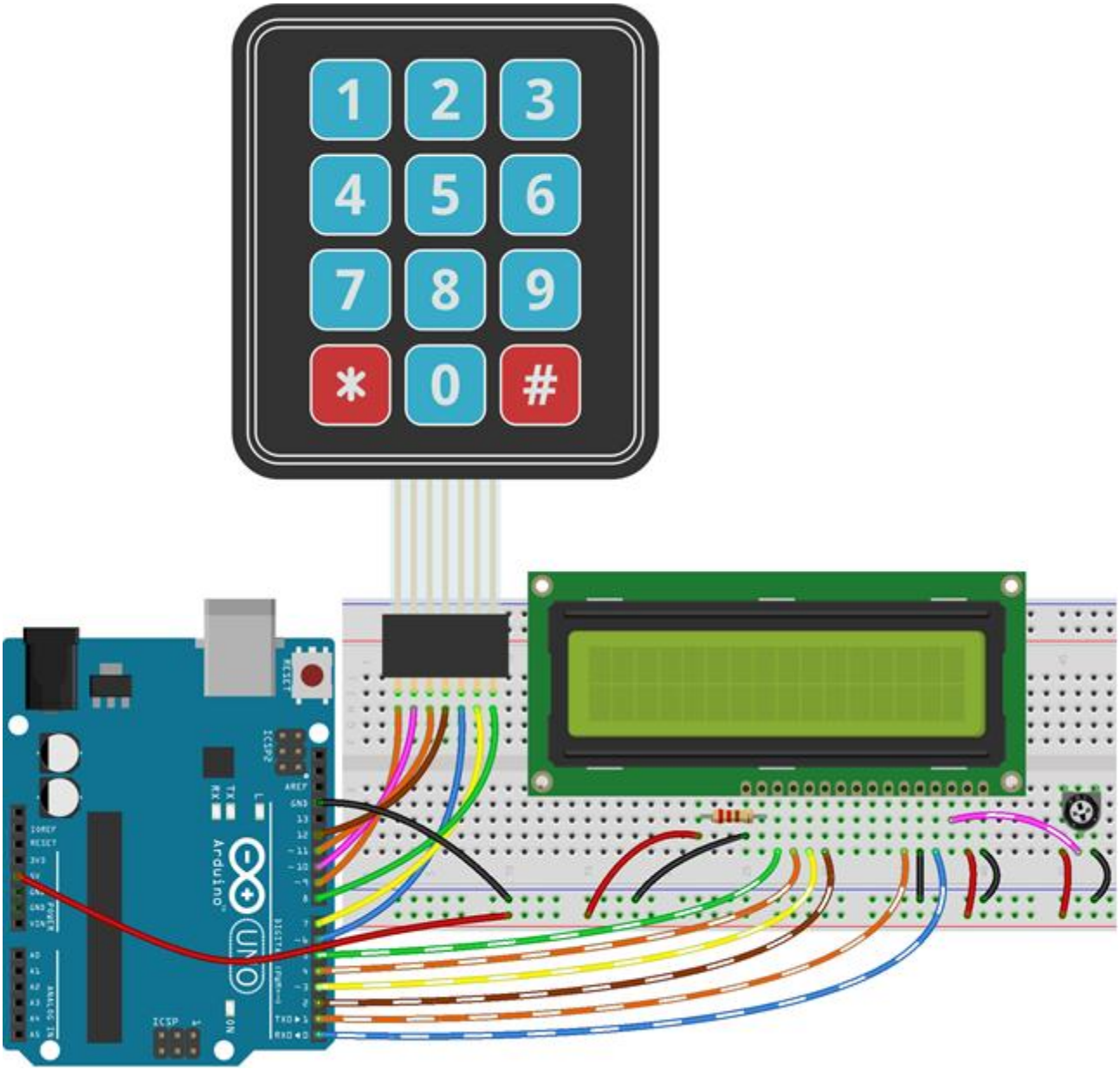
3-5-4-محاكاة لوحة المفاتيح من خلال برنامج Proteus



الشكل (3-30): محاكاة ربط لوحة المفاتيح مع الأردوينو من خلال برنامج Proteus.

3-5-5-تطبيق على شاشة LCD ولوحة المفاتيح

يوضح الشكل (3-31) تطبيقاً على وصل شاشة LCD ولوحة مفاتيح إلى الأردوينو. تم وصل القطب RS للشاشة مع المنفذ 0 للوحة الأردوينو، والقطب E مع المنفذ 1، وأقطاب المعطيات D4,D5,D6,D7 مع المنافذ 2,3,4,5. كذلك تم ربط السطر الأول للوحة المفاتيح مع المنفذ 9، والسطر الثاني مع المنفذ 10، والسطر الثالث مع المنفذ 11، والسطر الرابع مع المنفذ 12. تم ربط العمود الأول مع المنفذ 6، والعمود الثاني مع المنفذ 7، والعمود الثالث مع المنفذ 8. يطلب هذا التطبيق من المستخدم إدخال كلمة سر من 4 حروف. إذا كانت صحيحة يتم إظهار عبارة أن الكلمة صحيحة Password True على الشاشة، وإذا كانت خاطئة يتم إظهار عبارة أن الكلمة خاطئة Password False. بعد ثلاث مرات إدخال كلمات سر خاطئة يتم تشغيل ثنائي ضوئي على المنفذ 13 يعتبر بمثابة إنذار.



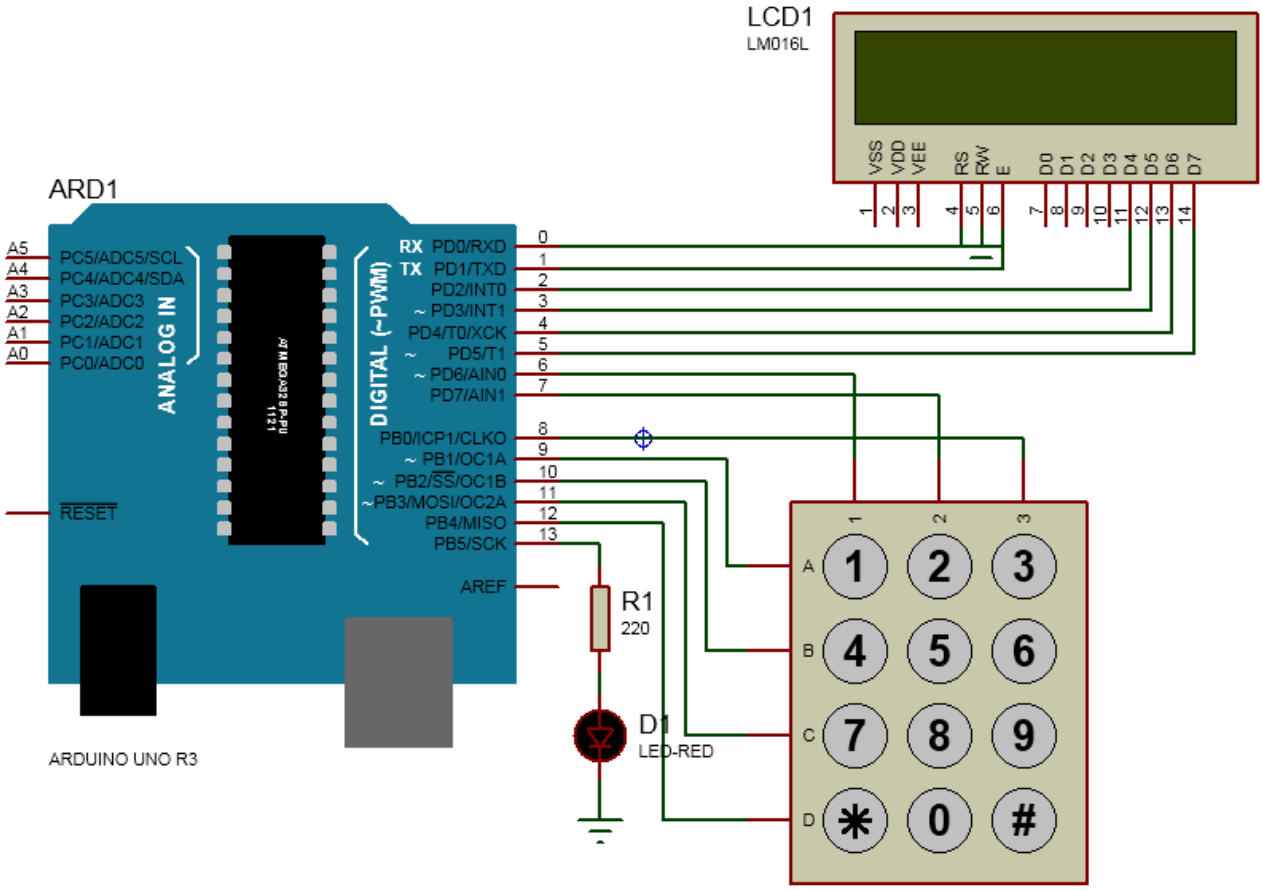
الشكل (3-31): تطبيق إدخال كلمة سر من خلال شاشة LCD ولوحة مفاتيح.

3-5-6-الكود البرمجي

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(0,1,2,3,4,5);


#include <Keypad.h>
const byte ROWS = 4; //four rows
const byte COLS = 3; //three columns
char keys[ROWS][COLS] = {
  {'1','2','3'},
  {'4','5','6'},
  {'7','8','9'},
  {'*','0','#'}
};
```

```
byte rowPins[ROWS] = {9, 10, 11, 12};//connect to the row pinouts of the keypad
byte colPins[COLS] = {6, 7, 8}; //connect to the column pinouts of the keypad
Keypad mykeypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );


byte x;


void setup() {
pinMode(13,OUTPUT);
digitalWrite(13,LOW);
lcd.begin(16, 2);
lcd.setCursor(0, 0);
x=0;
}


void loop() {
char password[4];
byte i;


lcd.print(" Enter Password ");
lcd.setCursor(6, 1);
for(i=0;i<=3;i++){
password[i]=mykeypad.waitForKey();
lcd.print("*");
}
lcd.setCursor(0, 0);
if(password[0]=='1' && password[1]=='2' && password[2]=='3' &&
password[3]=='4'){
lcd.print(" Password True ");
digitalWrite(13,LOW);}
else
{lcd.print(" Password False ");
x=x+1;
if( x==3) {digitalWrite(13,HIGH);}
delay(2000);
lcd.clear();
}
}
```

3-5-7-محاكاة لوحة المفاتيح وشاشة LCD من خلال برنامج Proteus



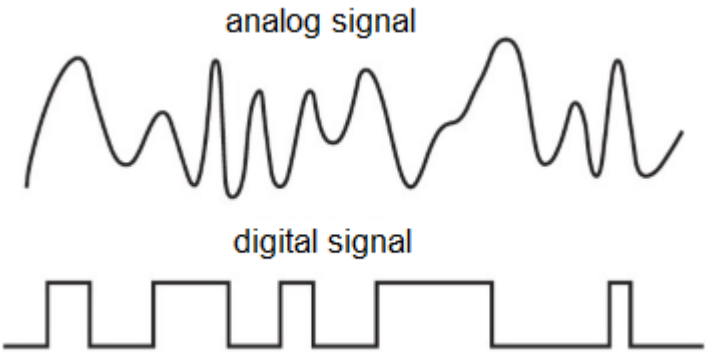
الشكل (3-32): محاكاة ربط شاشة LCD ولوحة المفاتيح مع الأردوينو من خلال برنامج Proteus.

قراءة إشارة الحساسات التشابهية

3-6-1- مقدمة

الحساس sensor هو عبارة عن أداة تعمل على كشف، وتحويل البارامتر الفيزيائي (درجة حرارة، رطوبة، ضوء، ضغط،.....) إلى إشارة كهربائية (جهد أو تيار) تتناسب مع تغيرات البارامتر الفيزيائي المقاس. تصنف الحساسات تبعاً لإشارة الخرج الكهربائية الناتجة إلى:

- حساسات تشابهية: تعطي في خرجها إشارة تشابهية (إشارة مستمرة تأخذ عدة قيم أو مطالات مع الزمن) كما هو موضح في الشكل (3-33).
- حساسات رقمية: تعطي في خرجها إشارة متقطعة ثنائية (0 منطقي، 1 منطقي). يتم تمثيل 0 منطقي بجهد، و1 منطقي بجهد آخر مختلف كما هو موضح في الشكل (3-33).

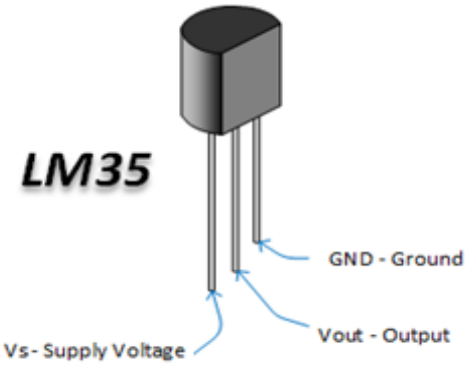


الشكل (3-33): الإشارة التشابهية analog signal، والإشارة الرقمية digital signal.

لابد لأي حساس تشابهي معرفة بارامتر هام عنه وهو حساسيته sensitivity التي تعبر عن مقدار تغير إشارة الخرج الكهربائية بالنسبة للبارامتر الفيزيائي. لنأخذ مثالاً شائعاً على الحساسات التشابهية وهو حساس الحرارة LM35 الموضح في الشكل (3-34). يعمل الحساس ضمن مجال درجة حرارة يمتد من -55 °C إلى 150 °C. لهذا الحساس ثلاث أرجل:

- تغذية Vs : تغذية الحساس من 4V إلى 20V.
- خرج Vout : تعطي هذه الرجل جهداً يتغير مع درجة الحرارة.
- أرضي GND.

حساسية الحساس LM35 هي 10mV/°C، بمعنى أن كلما ازدادت درجة الحرارة درجة مئوية واحدة (1 °C) كلما ازدادت إشارة خرج الحساس 10mV.



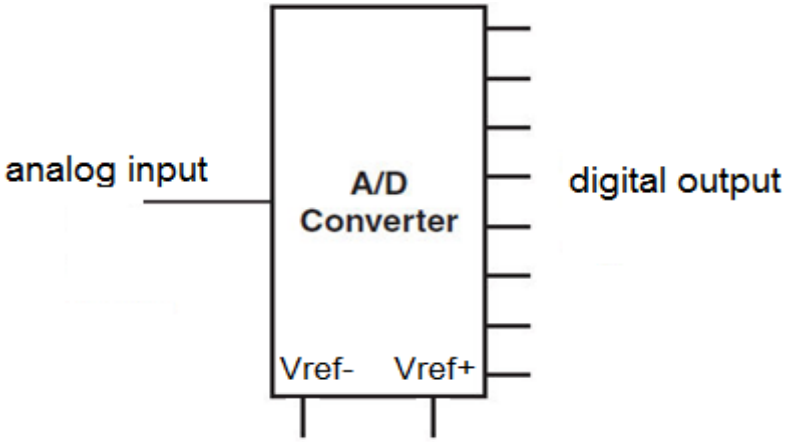
الشكل (3-34): حساس الحرارة LM35.

كما هو معلوم يتم معالجة الإشارات داخل المتحكم الصغري للوحة الأردوينو على شكل إشارات رقمية ثنائية، لذلك لا يمكن التعامل مع إشارة خرج الحساس التشابهي. لحل تلك المشكلة يستخدم ما يعرف بالمبدل التشابهي الرقمي Analog Digital Converter(ADC) الذي يعمل على تحويل الإشارة التشابهية إلى سلسلة من البتات الرقمية والتي لها قيمة مكافئة لقيمة الإشارة التشابهية. لن نهتم بالبنية الداخلية للمبدل ADC وإنما بالأرجل الخاصة به الموضحة في الشكل (3-35) وهي:

مدخل تشابهي analog input: يتم فيه تطبيق الإشارة التشابهية.

مخارج رقمية digital output: يظهر فيها القيمة الثنائية المكافئة. يحدد عدد هذه المخارج دقة التبديل كما سنرى.

مدخلان مرجعيان Vref-, Vref+: يطبق عليهما جهدان يحددان مجال جهد إشارة الدخل التشابهية (القيمة العظمى، والقيمة الصغرى). على سبيل المثال بفرض أنه تم تطبيق 0V على المدخل المرجعي Vref- ، وتم تطبيق 5V+ على المدخلVref+، عندئذ إذا كانت إشارة الدخل التشابهية تساوي Vref- أي 0V فإن إشارة الخرج الرقمية ستكون عند أدنى قيمة لها أي كلها أصفار (00000000)، وإذا كانت إشارة الدخل التشابهية تساوي Vref+ أي 5V+ فإن إشارة الخرج الرقمية ستكون عند أعلى قيمة لها أي كلها واحدات (11111111). الجهود المطبقة على المدخلين Vref-, Vref+ أيضاً لها دور في تحديد دقة التبديل.



الشكل (3-35): أرجل المبدل التمثيلي الرقمي.

هناك بارامتر هام للمبدل التمثيلي الرقمي وهو دقة التمييز resolution الذي يمثل الجهد المطلوب لكل تزداد إشارة الخرج الرقمية واحد منطقي (أي الخانة الأقل أهمية least significant bit (LSB). على سبيل المثال إذا كانت دقة تمييز مبدل ADC تساوي 5mV/LSB فهذا يعني أنه كلما ازدادت إشارة الدخل التشابهية 5mV كلما ازدادت إشارة الخرج الرقمية واحد منطقي. يقدم المبدل ذو دقة التمييز الأقل أداءً أفضل حيث يمكن له عندئذ تحسس وتبديل التغيرات الصغيرة لإشارة الدخل. تعطى دقة التمييز وفق العلاقة التالية:

$$R = \frac{(V_{ref+})-(V_{ref-})}{2^n}$$

n: عدد بتات خرج المبدل.

من الواضح من العلاقة السابقة أنه لإنقاص دقة التمييز نحتاج إلى إنقاص الفرق ما بين المدخلين المرجعيين $(V_{ref+} - V_{ref-})$ ، ولكن هذا سيكون على حساب إنقاص مجال إشارة الدخل التشابهية كما تم ذكره سابقاً. يمكن أيضاً إنقاص دقة التمييز من خلال زيادة عدد المخارج أو البتات الرقمية.

مثال: لدينا مبدل تشابهي رقمي ADC عدد مخارجه الرقمية n=10، تم ضبط Vref-=0V، Vref+=5V. المطلوب:

1-حساب دقة التمييز.

2- القيمة الرقمية المكافئة عندما يكون جهد إشارة الدخل التشابهية 100mV.

دقة التمييز resolution

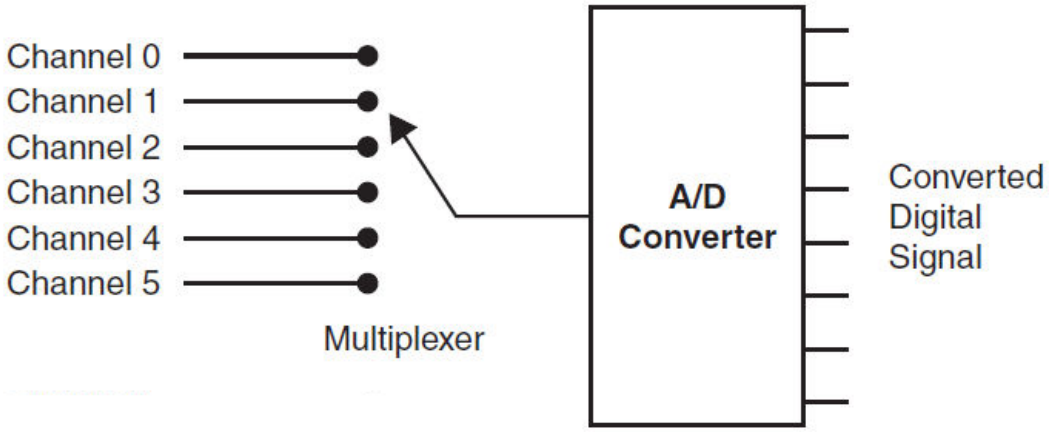
$$R = \frac{5-0}{2^{10}} \approx 4.88mV/LSB$$

القيمة الرقمية المكافئة عندما يكون جهد إشارة الدخل التشابهية 100mV:

$$\frac{V_{in}}{R} = \frac{100}{4.88} \approx 20 = 0000010100$$

3-2-6- ربط الحساسات التشابهية مع لوحة الأردوينو

تم في بعض المتحكمات الصغرية دمج وحدة مبدل ADC ضمن بنيتها الداخلية بحيث يمكن وصل الحساس التشابهي مباشرة مع المتحكم الصغري كما هو الحال في المتحكم ATmega328 الخاص بلوحة الأردوينو أونو. عدد بتات خرج المبدل للمتحكم 10، والجهد المرجعي Vref-=0V، أما الجهد المرجعي Vref+ يتم تطبيقه داخلياً ويأخذ 5V+ أو 1.1V أو قد يتم تطبيقه خارجياً على الرجل AREF في لوحة الأردوينو. عدد قنوات المبدل 6 قنوات، بمعنى أنه يمكن تطبيق 6 إشارات تشابهية كما هو موضح في الشكل (3-36). يتم الانتقال من إشارة إلى إشارة أخرى باستخدام التعليمات البرمجية. زمن قراءة إشارة الدخل التشابهية وتحويلها لإشارة رقمية 100µs.



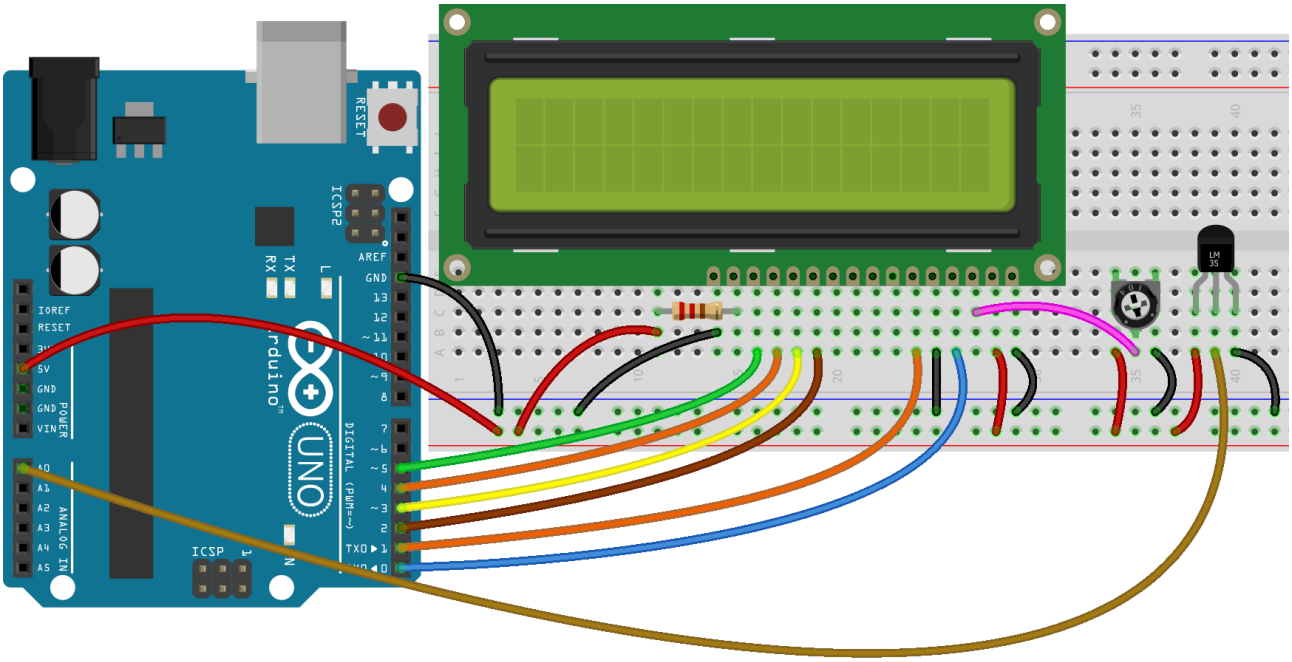
الشكل (3-36): استخدام التجميع multiplexing لإدخال عدة إشارات تشابهية (قنوات) إلى المبدل ADC من خلال التبديل من قناة إلى قناة أخرى.

يبين الجدول (3-8) التعليمات المستخدمة في بيئة التطوير Arduino IDE التي تسمح بأن تتعامل لوحة الأردوينو مع الإشارات التشابهية.

الجدول (3-8): تعليمات التعامل مع الإشارات التشابهية.

التعليمة	الشرح
analogRead(pin)	تسمح هذه التعليمة بقراءة القيمة التشابهية على الرجل المحددة pin. تعيد هذه التعليمة قيمة من 0 وحتى 1023.
analogReference(type)	تسمح هذه التعليمة بتغيير الجهد المرجعي تبعاً لخيارات النمط type: **DEFAULT** : يتم ضبط الجهد المرجعي افتراضياً ليساوي 5V في لوحات 5V Arduino ، و 3.3V في لوحات 3.3V Arduino. **INTERNAL**: يتم ضبط الجهد المرجعي داخلياً ليساوي 1.1 V من أجل لوحات متحكمات ATmega168، ATmega328، وليساوي 2.56 V من أجل لوحات متحكمات ATmega8. **EXTERNAL** : يأخذ الجهد المرجعي عندئذ الجهد الخارجي المطبق على رجل AREF (من 0V وحتى 5V+).

يوضح الشكل (3-37) تطبيقاً عملياً تم فيه ربط حساس الحرارة LM35 وشاشة LCD إلى لوحة الأردوينو أونو، ويراد إظهار درجة الحرارة على الشاشة. تم ربط الحساس مع المدخل التشابهي A0، وتم وصل القطب RS للشاشة مع المنفذ 0 للوحة الأردوينو، والقطب E مع المنفذ 1، وأقطاب المعطيات D4,D5,D6,D7 مع المنافذ 2,3,4,5.



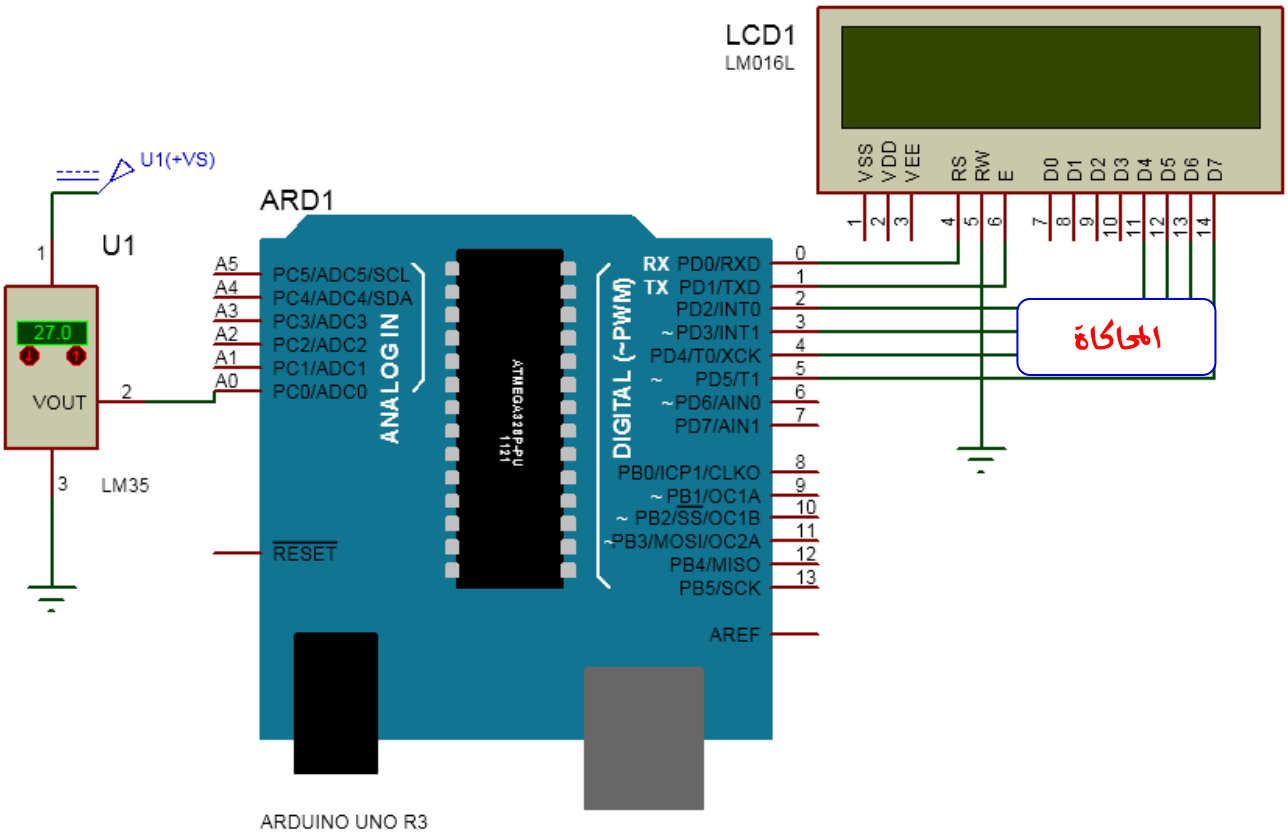
الشكل (3-37): تطبيق إظهار درجة الحرارة على شاشة LCD.

3-6-3-الكود البرمجي

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(0, 1, 2, 3, 4, 5);


void setup() {
 lcd.begin(16, 2);
 lcd.print(" Temperature is ");
}


void loop() {
unsigned int sensorValue;
lcd.setCursor(7, 1);
sensorValue=analogRead(0)/2;
lcd.print(sensorValue);
}
```

ملاحظة : القيمة الرقمية الناتجة عن المبدل ADC لا تعبر عن درجة الحرارة الواقعية، لهذا لابد من معايرتها (أي ضربها أو قسمتها على قيمة معينة قبل إظهارها). ما هي هذه القيمة؟ حساس الحرارة له حساسية 10mv/°C، والمبدل ADC له دقة تمييز تساوي $(5-0)/2^{10}=4.88mv/LSB$ مع افتراض أن Vref+=5V. هذا يعني أنه تقريباً كل درجة مئوية ستزداد القيمة الرقمية بمقدار 2 ، لهذا لابد من أن نقسم القيمة الرقمية الناتجة على 2 لنحصل على القيمة الحقيقية لدرجة الحرارة و هذا ما نجده في البرنامج.

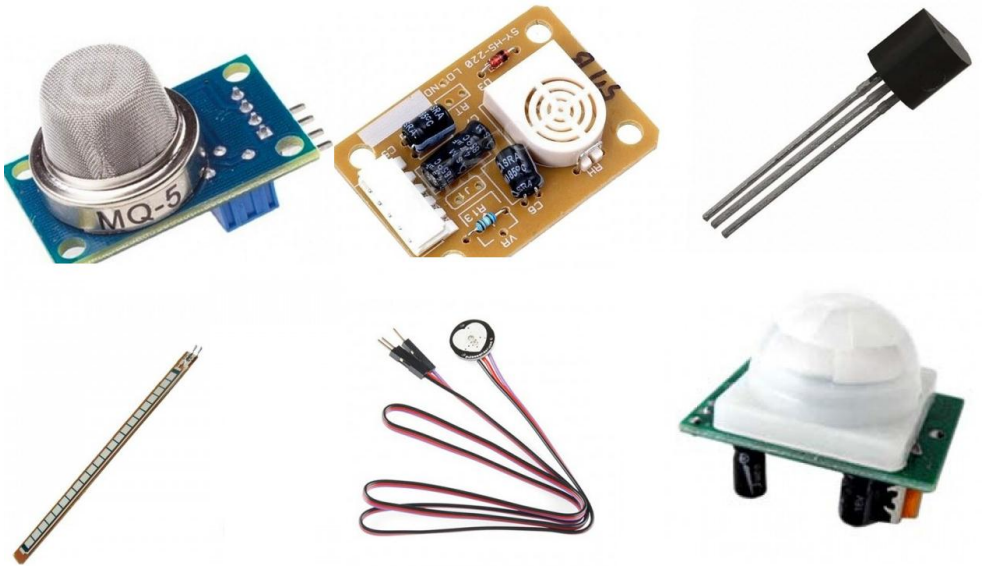
3-6-4-محاكاة ربط حساس درجة الحرارة من من خلال برنامج Proteus



الشكل (3-38): محاكاة إظهار درجة الحرارة على شاشة LCD من خلال برنامج Proteus.

3-6-5- نماذج لحساسات تشابهية ورقمية

يوضح الشكل (3-39) بعض الحساسات التشابهية والرقمية المتوفرة في الأسواق ويمكن من خلالها تصميم العديد من المشاريع والدارات الالكترونية.



الشكل (3-39): حساسات مختلفة: حرارة، رطوبة، غاز، حركة، معدل نبضات القلب، انحناء.

التحكم بمحركات التيار المستمر، والخطوية، والسيرفو

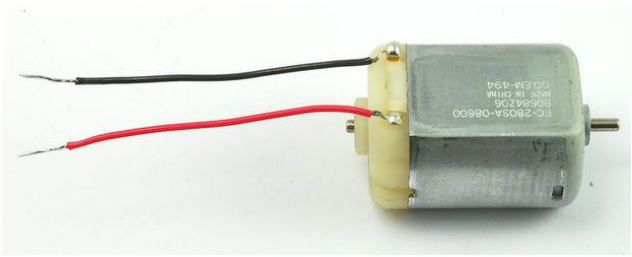
3-7-1-مقدمة

المحرك الكهربائي هو آلة تحول الطاقة الكهربائية إلى طاقة ميكانيكية. يمكن من خلال المحرك المتصل مع لوحة الأردوينو تصميم العديد من الدارات والمشاريع الالكترونية كالتحكم بحركة سيارة وذراع روبوتية وغير ذلك. سنتكلم في الفقرات التالية عن المحركات التالية وكيفية ربطها مع لوحة الأردوينو والتحكم بها:

- محركات التيار المستمر DC motor
- محركات خطوية Stepping motor
- محركات السيرفو Servo motor

3-7-2- محرك التيار المستمر DC motor

تعتبر أبسط أنواع المحركات الكهربائية مثل المحركات الموجودة في المسجلة وألعاب السيارات. تحتاج إلى تغذية بجهد مستمر. لهذه المحركات سلكان فقط كما هو موضح في الشكل (3-40). عندما يتم توصيل فرق جهد مستمر إلى هذين السلكين فإن المحرك سيدور باتجاه معين، عند عكس قطبية التغذية فإن المحرك سيتحرك بالاتجاه الآخر. جهد التغذية يتراوح ما بين 6V إلى 12V. تستجر هذه المحركات تياراً لا بأس به لا يستطيع المتحكم الصغري في لوحة الأردوينو تأمينه لهذا لا بد من استخدام دارة عزل ما بين لوحة الأردوينو والمحرك تؤمن له هذا التيار. هناك طرق عديدة يمكن بها تنفيذ دارة العزل: ترانزستورات – ريليهات – دارات متكاملة مثل L293 أو L298.



الشكل (3-39): محرك التيار المستمر DC motor.

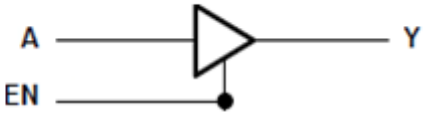
3-7-2-1- الدارة المتكاملة L293

الدارة المتكاملة L293 عبارة عن أربع بوابات تتصف بأنها:

1-ثلاثية الحالة tri-state: أي أن خرجها يأخذ ثلاث حالات تبعاً لكل من الدخل A، ورجل التفعيل EN. يبين الشكل(3-41) البوابة ثلاثية الحالة وجدول الحقيقة. يشير الحرف Z إلى الممانعة العالية high impedance.

INPUTS†		OUTPUT
A	EN	Y
H	H	H
L	H	L
X	L	Z

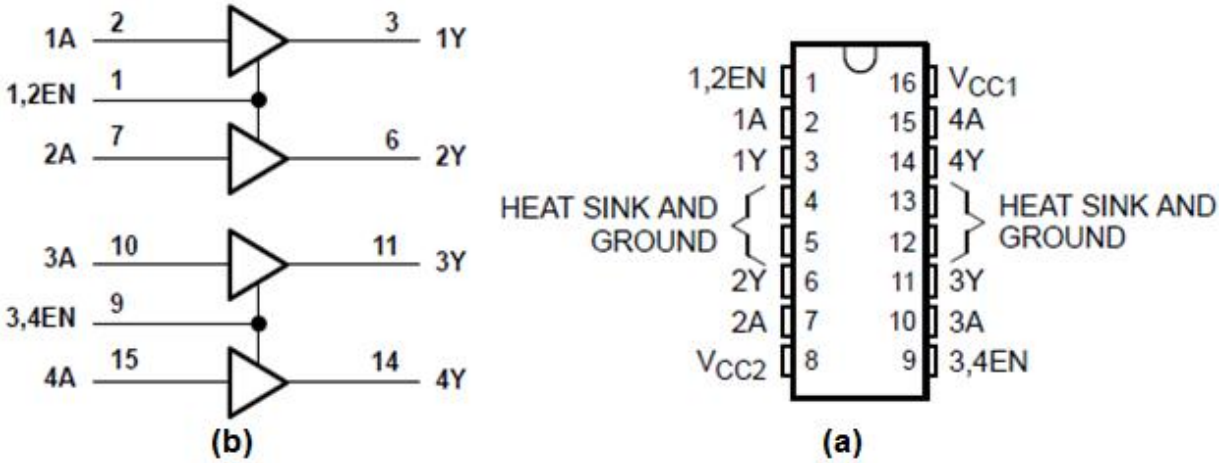
H = high level, L = low level, X = irrelevant,
Z = high impedance (off)



الشكل (3-41): البوابة ثلاثية الحالة وجدول الحقيقة.

2-من نمط دفع – جذب push-pull حيث تعمل كمصدر للتيار وفي نفس الوقت كمصرف له. تسمح L293 بمرور تيار في البوابة يصل إلى 1A. يوجد نمط آخر لهذه الدارة هو L293D الذي يمتاز بأنه يحتوي على ديودات حماية داخلية وبالتالي ليس هناك حاجة لتركيبها أثناء وصل الدارة مع المحركات ولكن مع تيار بوابة أقل يصل إلى 600mA.

يبين الشكل (3-42) أرجل الدارة L293، وأرقام مداخل ومخارج البوابات الداخلية الأربعة الموافقة.



الشكل (3-42): (a) أرجل الدارة المتكاملة L293، (b) البوابات الأربعة للدارة المتكاملة L293.

يبين الجدول (3-9) وظيفية كل رجل لدارة L293.

الجدول (3-9): وظائف أرجل الدارة المتكاملة L293.

رقم الرجل	الرجل	الوظيفة
1	EN 1,2	رجل تفعيل للبوابتين 1,2 لابد من وصلها مع +5V
2	1A	مدخل البوابة 1
3	1Y	مخرج البوابة 1
4	Ground	أرضي
5	Ground	أرضي
6	2Y	مخرج البوابة 2
7	2A	مدخل البوابة 2
8	Vcc2	يطبق جهد يؤمن التغذية لخرج البوابة إذا كانت في حالة المستوى المرتفع high level. مجال الجهد من 4.5V إلى 36V.

رقم الرجل	الرجل	الوظيفة
9	EN 3,4	رجل تفعيل للبوابتين 3,4 لابد من وصلها مع +5V
10	3A	مدخل البوابة 3
11	3Y	مخرج البوابة 3
12	Ground	أرضي
13	Ground	أرضي
14	4Y	مخرج البوابة 4
15	4A	مدخل البوابة 4
16	Vcc1	يتم وصلها إلى +5V لتأمين التغذية لدارة دخل البوابة.

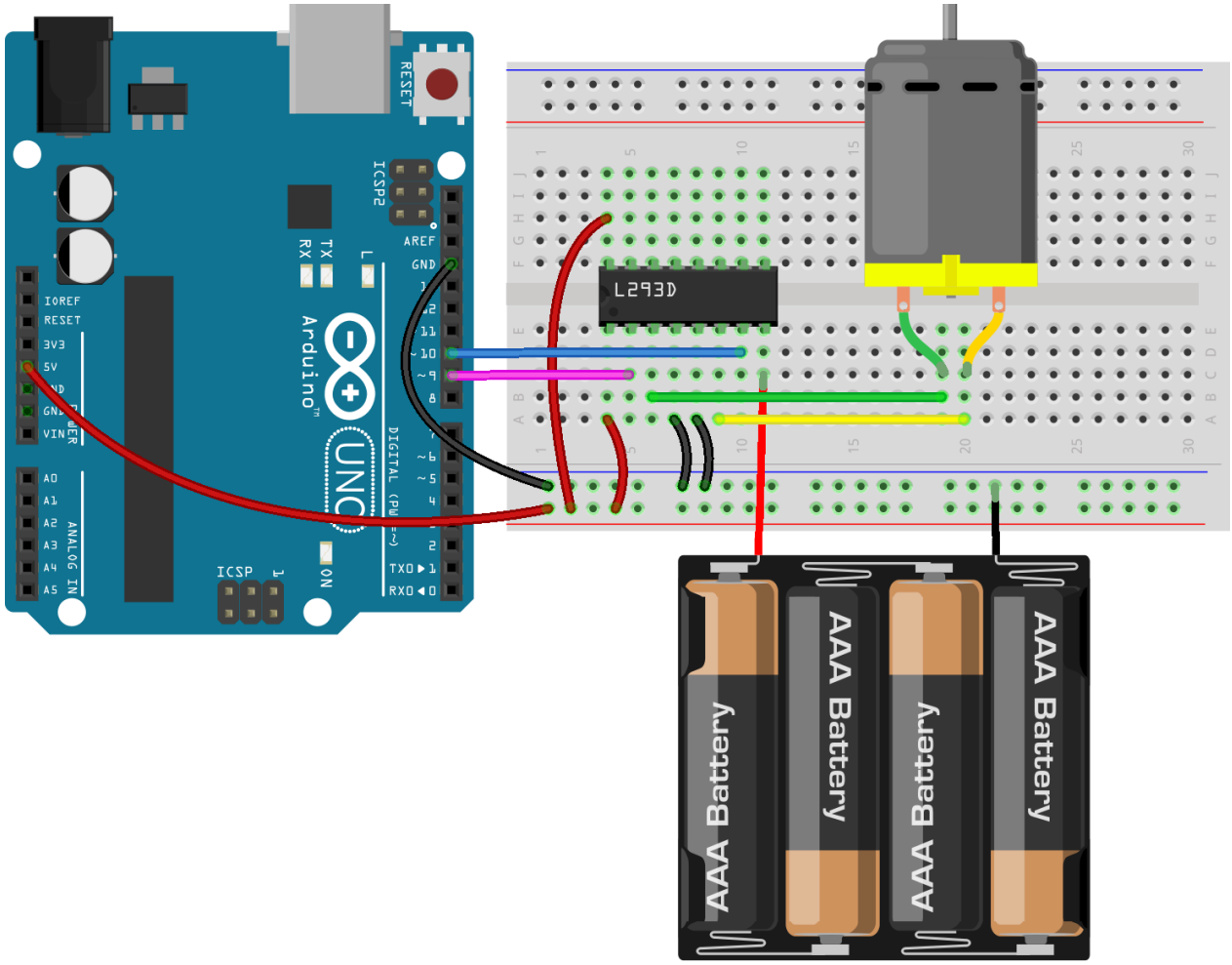
3-7-2-1-1-التحكم بالمحرك المستمر من خلال لوحة الأردوينو مع استخدام لدارة القيادة L293

يبين الشكل (3-43) مخطط التوصيل العملي للوحة الأردوينو مع دارة القيادة L293 والمحرك المستمر. تم ربط المنفذين 9, 10 للوحة الأردوينو مع مدخلي البوابتين 1, 2 للدارة L293. تم وصل الرجل الأولى للدارة المتكاملة (E 1,2) مع جهد +5V لتفعيل البوابتين الأولى والثانية. تم وصل المحرك المستمر مع مخارج البوابتين 1,2 لدارة L293، مع العلم أنه يمكن وصل محرك آخر مع البوابتين 3,4 بنفس الطريقة. تم استخدام بطارية 6V كمنبع تغذية لمخرج بوابات L293 المتصلة مع المحرك، ويمكن استخدام منابع تغذية أخرى. بين الجدول (3-10) حالة المحرك المستمر من حيث الحركة وجهة الدوران تبعاً للقيم المطبقة على الأرجل الرقمية pin9, pin10 (أي مداخل البوابتين 1A,2A).

الجدول (3-10): حالة المحرك المستمر تبعاً
للقيم المطبقة على الأرجل الرقمية pin9, pin10.

pin9	Pin10	حالة المحرك
Low	Low	متوقف
High	Low	يدور عكس عقارب الساعة
Low	High	يدور مع عقارب الساعة
High	High	متوقف

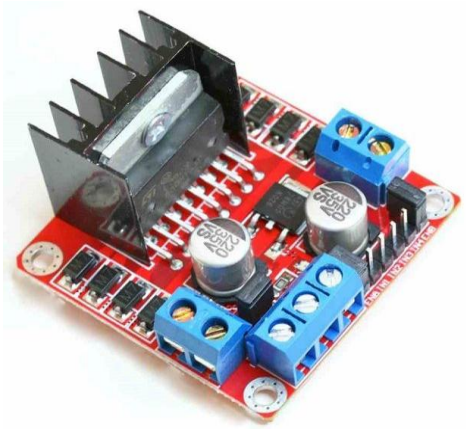
لابد من الإشارة إلى أن دارة l293D لا تحتاج لوصل الديودات فهي تحتوي داخلياً عليها. كذلك فإن جهد خرج بوابات دارة L293 فعلياً أقل من Vcc2 بحوالي 0.9V.



الشكل(3-43): وصل لوحة الأردوينو مع المحرك المستمر باستخدام الدارة المتكاملة L293D.

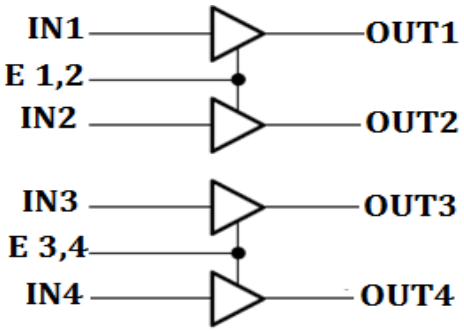
3-7-2-2-وحدة التحكم بالمحركات L298N

في هذه الفقرة سنتحدث أيضاً عن دارة متكاملة أخرى تشبه دارة L293 ولكن تمتاز عنها بأنها تقدم تياراً أكبر للمحركات وهي دارة L298N. يمكن شراء هذه الدارة مثبتة على لوحة مع العناصر الالكترونية الأخرى المطلوبة بحيث يمكن التعامل معها بسهولة. بين الشكل (3-44) وحدة L298N التي تستخدم لقيادة المحرك المستمر، أو المحرك الخطوي كما سنرى لاحقاً. دارة L298N هي دارة الكترونية من نمط H-bridge (أي أنها تسمح بتطبيق جهد على الحمولة في اتجاهات متعاكسة).



الشكل (3-44): وحدة التحكم بالمحركات L298N.

تشبه دارة L298N الدارة L293 والتي تم دراستها في الفقرة السابقة من حيث أنها تحتوي على 4 بوابات من نمط دفع – جذب كما هو موضح في الشكل (3-45). بين الجدول (3-11) العلاقة ما بين المداخل المنطقية IN1, IN2 مع المخارج OU1, OUT2 والجدول (3-12) العلاقة ما بين المداخل المنطقية IN3, IN4 مع المخارج OU3, OUT4. المستوى L يمثل 0 منطقي (0V....1,5V) ، المستوى H يمثل 1 منطقي (2,3V...+5V). يشير X إلى أنه بغض النظر عن قيمة الجهد. ويشير Z إلى حالة الممانعة العالية (دارة مفتوحة). Vin هو الجهد المطبق على الرجل 4، والذي يستخدم لتغذية المحركات، Vdrop تتراوح قيمته ما بين 1V و 3V اعتماداً على جهد الدخل، والتيار المسحوب.



الشكل (3-45): مخطط مبسط لدارة L298.

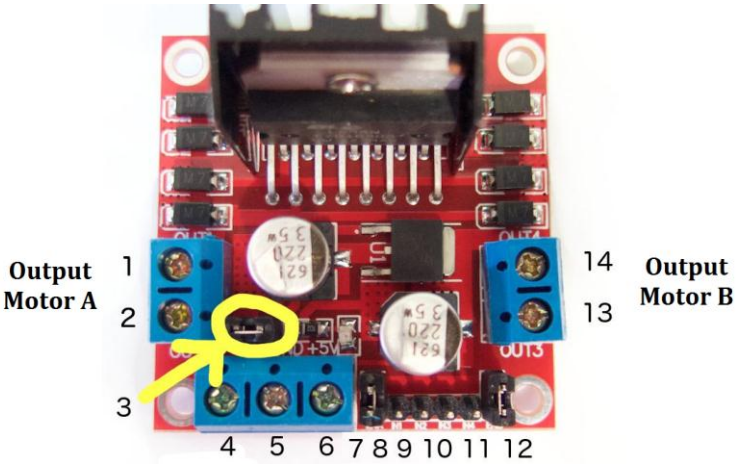
الجدول (3-11) العلاقة ما بين المداخل المنطقية IN1, IN2 مع المخارج OU1, OUT2

EN 1,2	IN1, IN2	OUT1, OUT2
L	X	Z
H	H	Vin-Vdrop
H	L	L

الجدول (3-12) العلاقة ما بين المداخل المنطقية IN3, IN4 مع المخارج OU3, OUT4

EN 3,4	IN3, IN4	OUT3, OUT4
L	X	Z
H	H	Vin-Vdrop
H	L	L

يبين الشكل (3-46) مداخل ومخارج وحدة التحكم L298N:



الشكل (3-46): مداخل ومخارج وحدة التحكم L298N.

يلخص الجدول (3-13) وظائف أرجل مداخل ومخارج وحدة التحكم L298N:

الجدول (3-13) وظائف أرجل مداخل ومخارج وحدة التحكم L298N

الرقم	اسم الرجل أو الوصلة	الوظيفة
1	OUT1	مخرج OUT1 البوابة الأولى. يتم ربطه مع أحد أطراف المحرك الأول DC أو المحرك الخطوي +A.
2	OUT2	مخرج OUT2 البوابة الثانية. يتم ربطه مع الطرف الآخر للمحرك الأول DC أو المحرك الخطوي -A
3	Jumper	عبارة عن وصلة jumper. إذا كانت موجودة يتم تفعيل منظم الجهد المثبت على اللوحة. يتم إزالة هذه الوصلة إذا تم تطبيق جهد تغذية أعلى من 12V على الرجل رقم 4.
4	Vin	يتم هنا تطبيق جهد تغذية المحركات. يتراوح الجهد المطبق ما بين 7V وحتى35V.
5	GND	أرضي.
6	+5V	إذا كانت وصلة jumper موجودة (الرجل 3) فإن هذه الرجل تقدم جهداً مقداره 5V+ ناتج عن منظم الجهد الداخلي. وبالتالي إمكانية تغذية دارات أخرى مثل الأردوينو. إذا لم تكن الوصلة موجودة، فلن يتم تفعيل منظم الجهد الداخلي، وعندئذ لابد من تطبيق جهد 5V+ خارجي عليها. أي أن هذه الرجل من الممكن أن تكون خرج أو دخل 5V+.
7	Jumper (ENA)	عبارة عن وصلة jumper لتفعيل مخرجي البوابتين الأولى والثانية (OUT1, OUT2). إذا كانت موجودة فهذا يعني أنه تم تطبيق 5V+ على رجل تفعيل البوابتين الأولى والثانية EN 1,2.
8	IN1	مدخل منطقي للبوابة الأولى. يتم ربطه مع رجل رقمية للأردوينو.
9	IN2	مدخل منطقي للبوابة الثانية. يتم ربطه مع رجل رقمية للأردوينو.
10	IN3	مدخل منطقي للبوابة الثالثة. يتم ربطه مع رجل رقمية للأردوينو.
11	IN4	مدخل منطقي للبوابة الرابعة. يتم ربطه مع رجل رقمية للأردوينو.
12	Jumper (ENB)	عبارة عن وصلة jumper لتفعيل مخرجي البوابتين الثالثة والرابعة (OUT3, OUT4). إذا كانت موجودة فهذا يعني أنه تم تطبيق 5V+ على رجل تفعيل البوابتين الثالثة والرابعة EN 3,4.
13	OUT3	مخرج OUT3 البوابة الثالثة. يتم ربطه مع أحد أطراف المحرك الثاني DC أو المحرك الخطوي +B.
14	OUT4	مخرج OUT4 البوابة الرابعة. يتم ربطه مع الطرف الآخر للمحرك الثاني DC أو المحرك الخطوي -B.

يمكن إجمال مزايا وحدة التحكم بالمحركات L298N بما يلي:

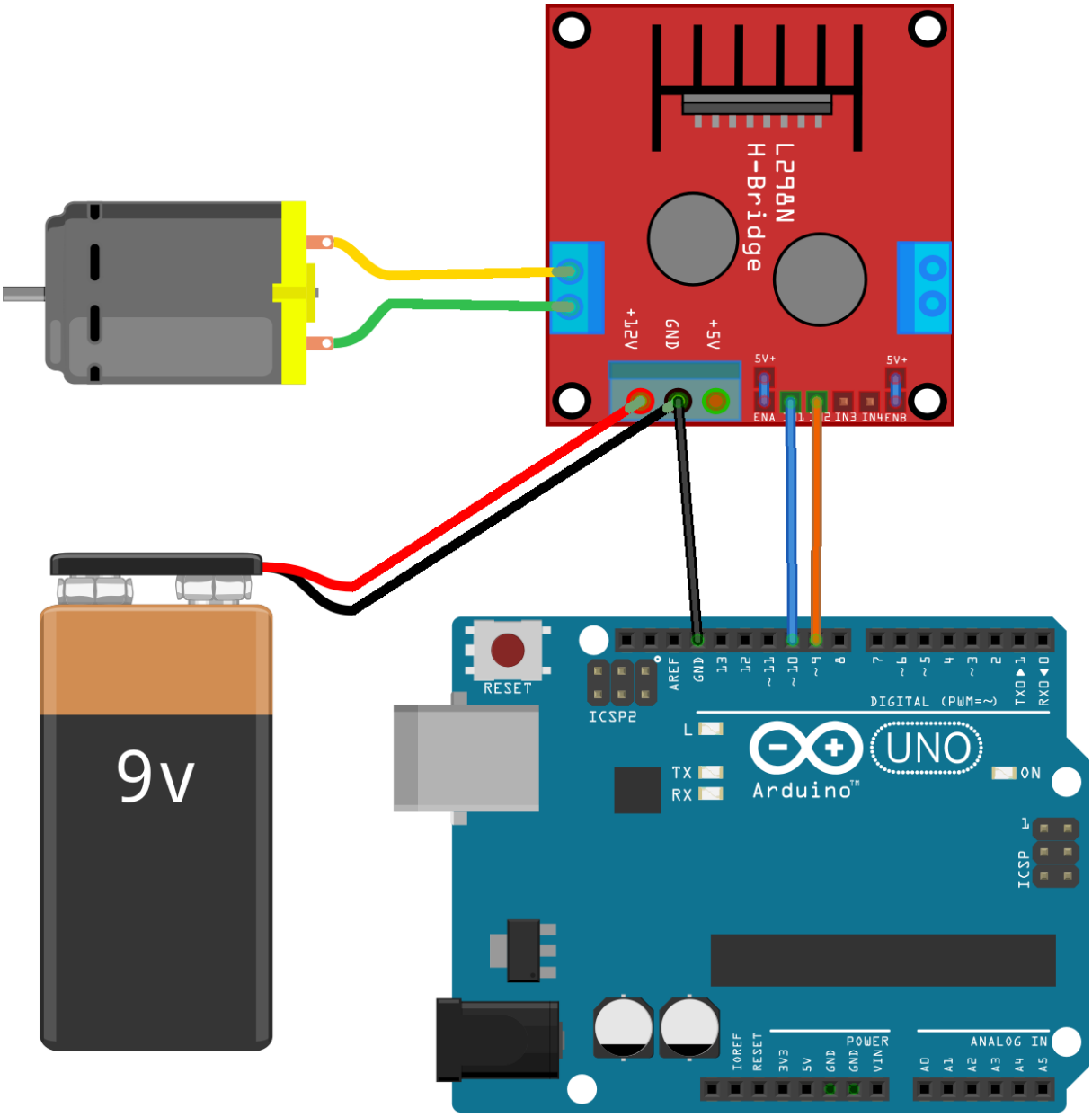
- يمكن من خلال هذه الوحدة قيادة محركي تيار مستمر DC، أو محرك خطوي واحد.
- جهد تغذية المحركات ما بين (7V-35V).
- التيار الذي يمكن استجراره للمحرك يصل إلى 2A.
- مستوى التحكم (الجهد المنطقي عند وصل لوحة الأردوينو): المستوى المنخفض: 0.3V...1,5V- ، المستوى المرتفع 2,3V.....5V.
- التيار المنطقي: 0~36mA.
- أعظم تبديد للاستطاعة: 20W.
- درجة حرارة العمل: 25°C- وحتى 130°C+.
- الأبعاد: 60mm*54mm
- الوزن : 48g~.

3-7-2-2-1-التحكم بالمحرك المستمر من خلال لوحة الأردوينو مع استخدام للوحة التحكم L298N

يبين الشكل (3-47) مخطط التوصيل العملي للوحة الأردوينو مع لوحة التحكم L298N والمحرك المستمر. تم ربط المنفذين 9, 10 للوحة الأردوينو مع الأرجل 8,9 للوحة (IN1,IN2). كذلك تم ربط المحرك DC مع الأرجل 1,2 للوحة (OU1,OUT2). تم تزويد اللوحة بمنبع تغذية 9V+ على الرجل 4 والذي من خلاله يتم تغذية المحركات. تم المحافظة على الوصلة 3 (jumper)، وهذا يعني أنه سيتم تفعيل منظم الجهد الداخلي للوحة 5V+ وبالتالي لا يوجد حاجة لتغذيتها بجهد 5V+. إذا تم نزع هذه الوصلة لابد من تأمين تغذية 5V+ على الرجل 6. تم المحافظة على الوصلة 7 (jumper)، وهذا يعني أن المخارج OU1,OUT2 مفعلة. بين الجدول (3-14) حالة المحرك المستمر من حيث الحركة وجهة الدوران تبعاً للقيم المطبقة على الأرجل الرقمية pin9, pin10 .

الجدول (3-14): حالة المحرك المستمر تبعاً للقيم المطبقة على الأرجل الرقمية pin9, pin10.

حالة المحرك	Pin10	pin9
متوقف	Low	Low
يدور عكس عقارب الساعة	Low	High
يدور مع عقارب الساعة	High	Low
متوقف	High	High



الشكل (3-47): مخطط وصل لوحة الأردوينو مع المحرك المستمر باستخدام وحدة L298N.

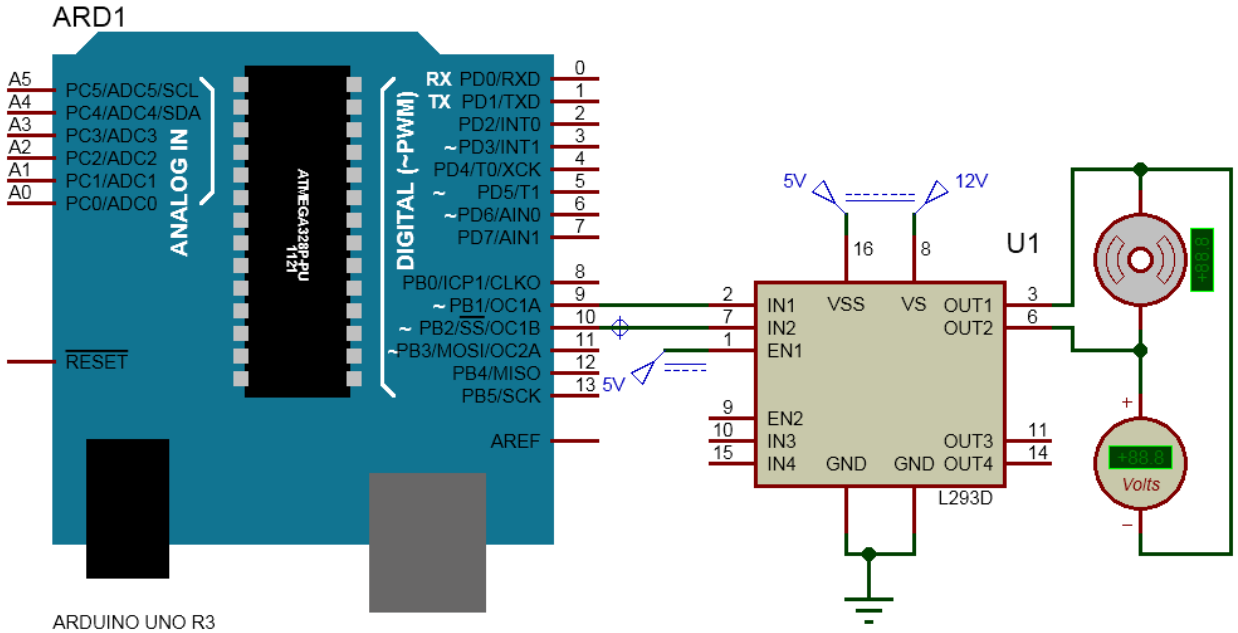
3-2-7-3-الكود البرمجي

فيما يلي الكود البرمجي للتحكم بحركة المحرك DC سواء باستخدام الدارة المتكاملة L293D أو لوحة L298N وبشكل يتوافق مع الشكلين (3-43) و (3-47). في هذا المثال سنعمل على تحرك المحرك لليمين لمدة خمس ثوانٍ، ولليسار لمدة خمس ثوانٍ، ومن ثم إيقافه لمدة خمس ثوانٍ.

```
void setup() {
pinMode(9,OUTPUT);
pinMode(10,OUTPUT);
}

void loop() {
move_motor('L');    // move left
```

```
delay(5000);
move_motor('R');   // move right
delay(5000);
move_motor('S');    // stop
delay(5000);
}

void move_motor(char Direction)
{
if (Direction=='L' )
{
 digitalWrite(9,HIGH);
 digitalWrite(10,LOW);
}
if (Direction=='R' )
{
 digitalWrite(9,LOW);
 digitalWrite(10,HIGH);
}
if (Direction=='S' )
{
 digitalWrite(9,LOW);
 digitalWrite(10,LOW);
}}
```

3-7-2-4-محاكاة التحكم بمحرك مستمر dc من خلال برنامج Proteus

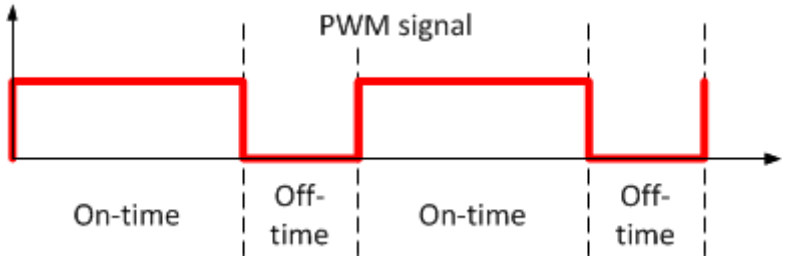


الشكل (3-48): محاكاة التحكم بمحرك مستمر dc باستخدام الدارة المتكاملة L293 من خلال برنامج Proteus.

3-7-2-5-التحكم بسرعة المحرك المستمر من خلال لوحة الأردوينو

يوجد عدة طرق للتحكم بسرعة المحرك المستمر DC، ولكن من أبسط هذه الطرق هو أن تطبق إشارة تعديل عرض النبضة Pulse width modulation(PWM) على المحرك.

إشارة PWM هي إشارة مربعة ترددها ثابت، ويتغير فيها الفترة الزمنية لنبضة ON (المستوى العالي HIGH)، والفترة الزمنية لنبضة OFF (المستوى المنخفض LOW) كما هو موضح في الشكل (3-49).



الشكل (3-49): إشارة تعديل عرض النبضة PWM

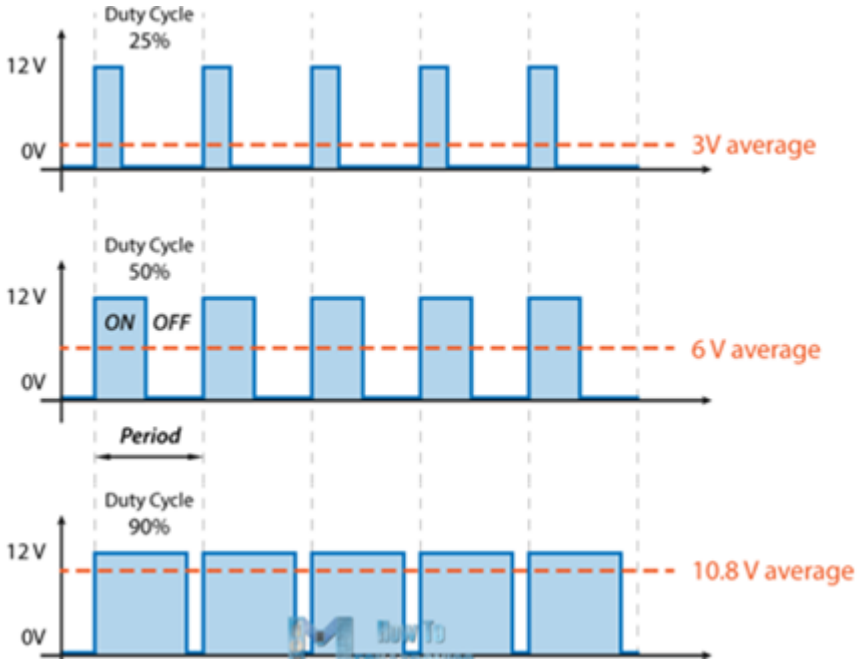
يعرف بارامتر دورة التشغيل duty cycle وفق العلاقة التالية:

$$D = \frac{PW}{T} \times 100\ \%$$

PW: عرض نبضة المستوى العالي (الفترة الزمنية للنبضة)، T: دور إشارة PWM (الفترة الزمنية لنبضة المستوى العالي ونبضة المستوى المنخفض).

تعتمد القيمة المتوسطة (المستمرة) لجهد إشارة PWM على دورة التشغيل. بفرض أن جهد المستوى العالي 12V+ نستنتج ما يلي كما في الشكل (3-50):

- إذا كانت دورة التشغيل 0% فإن القيمة المتوسطة لجهد إشارة PWM هي صفر.
- إذا كانت دورة التشغيل 25% فإن القيمة المتوسطة لجهد إشارة PWM هي 25% من 12V (أي 3V).
- إذا كانت دورة التشغيل 50% فإن القيمة المتوسطة لجهد إشارة PWM هي 50% من 12V (أي 6V).
- إذا كانت دورة التشغيل 90% فإن القيمة المتوسطة لجهد إشارة PWM هي 90% من 12V (أي 10.8V).
- إذا كانت دورة التشغيل 100% فإن القيمة المتوسطة لجهد إشارة PWM هي 100% من 12V (أي 12V).



الشكل (3-50): تغير القيمة المتوسطة لإشارة PWM مع تغير دورة التشغيل.

إذا تم تطبيق إشارة PWM على أحد طرفي محرك مستمر، والطرف الآخر تم وصله للأرضي، وتم تغيير عرض النبضات (دورة التشغيل) ستتغير القيمة المتوسطة لإشارة PWM وهذا ما يؤدي إلى تغيير سرعة المحرك، وبالتالي التحكم بها.

تتضمن لوحة الأردوينو أونو على 6 منافذ تسمح بتوليد إشارة PWM هي 3,5,6,9,10,11. يولد المنفذان 5,6 إشارة PWM بتردد 980Hz تقريباً، وبقية المنافذ 490 Hz تقريباً.

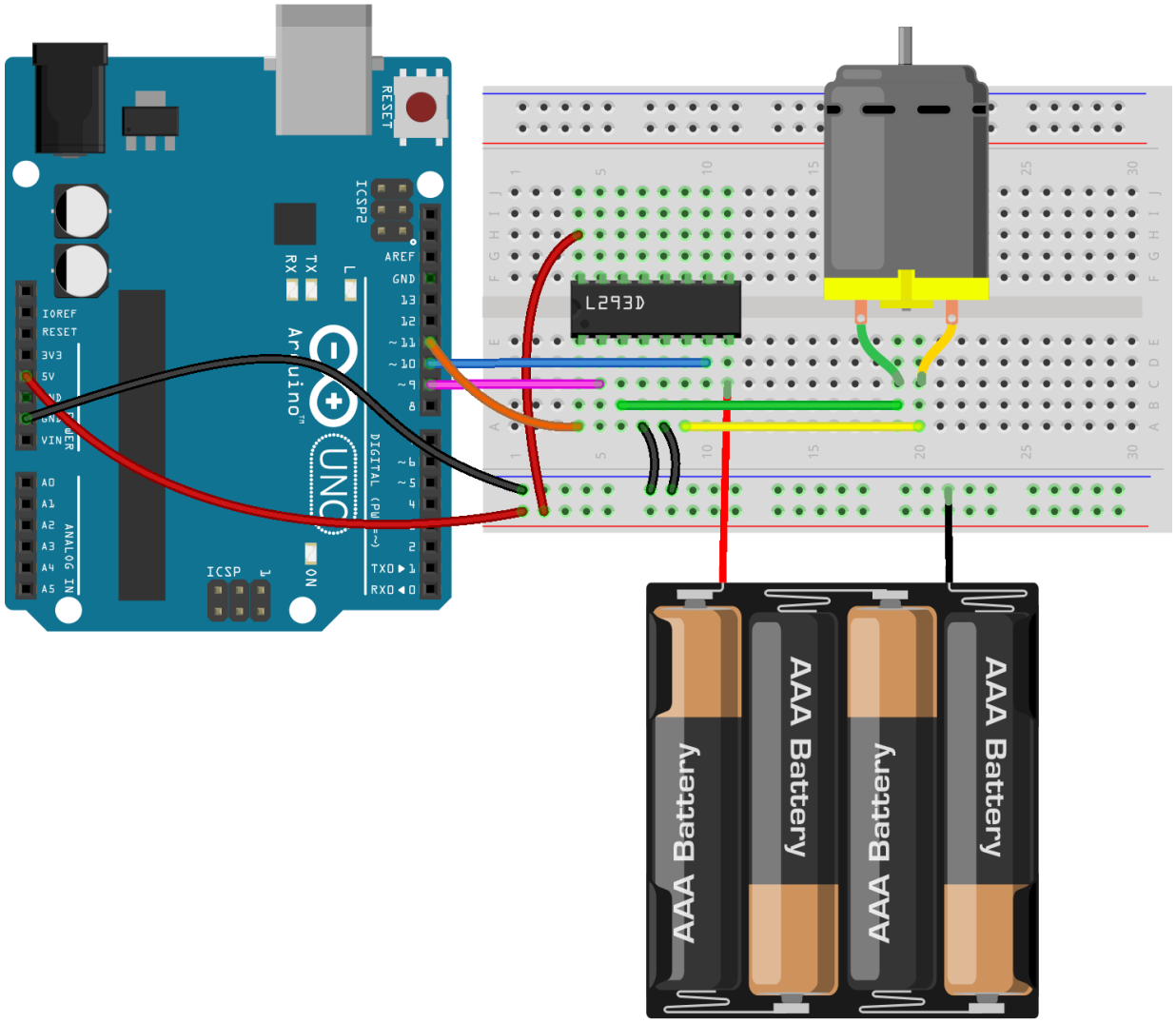
تستخدم التعليمة analogWrite(pin, value) في برنامج Arduino IDE لتوليد إشارة PWM حيث:

pin: رقم المنفذ الذي سيولد إشارة PWM.

value: تحدد دورة التشغيل. تأخذ قيمة من 0 وحتى 255. مثلاً من أجل value=0 فإن دورة التشغيل 0% ، ومن أجل value=64 فإن دورة التشغيل 25%، ومن أجل value=127 فإن دورة التشغيل 50% ، ومن أجل value=191 فإن دورة التشغيل 75% ، ومن أجل value=255 فإن دورة التشغيل 100%.

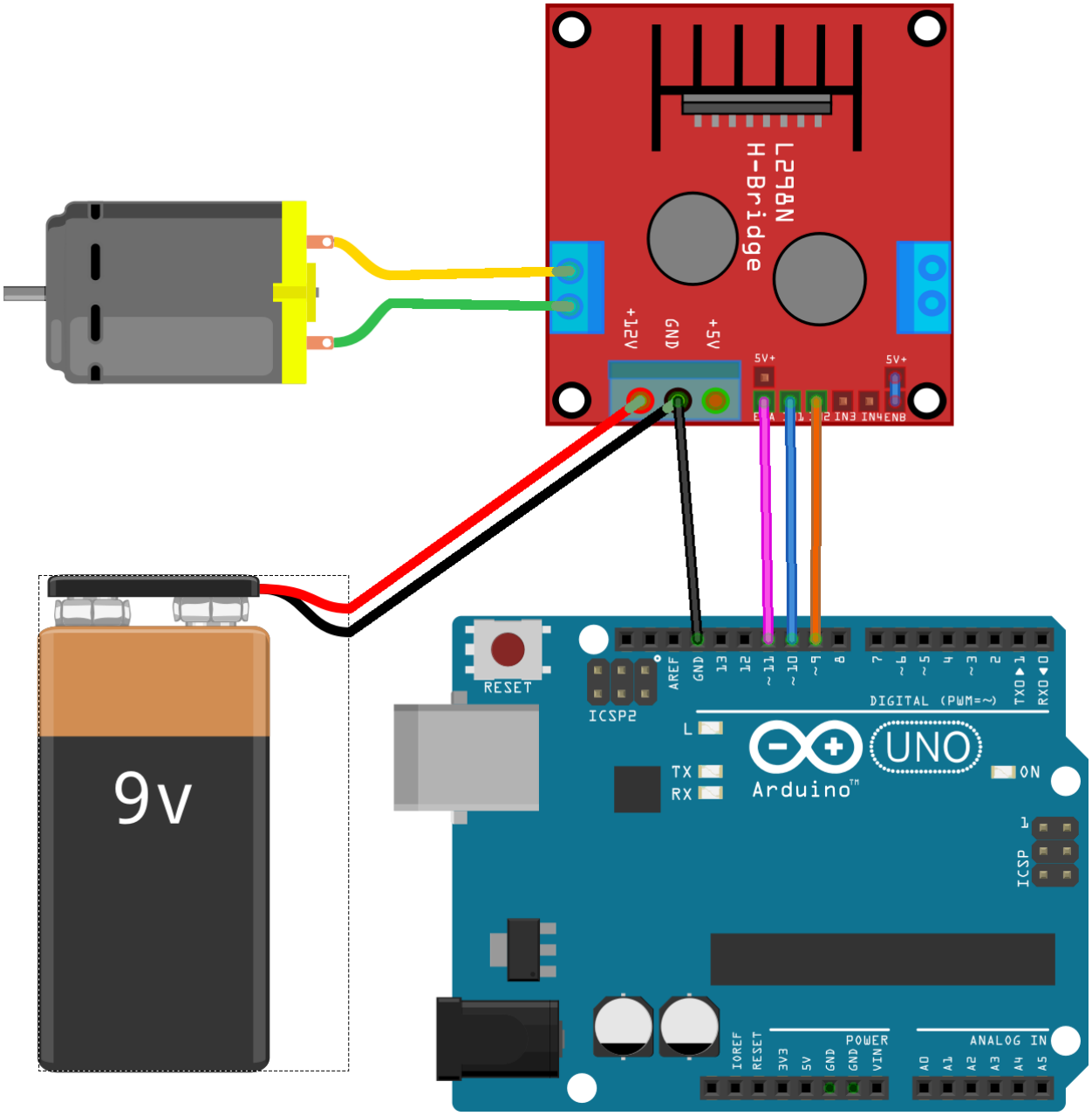
لا يوجد داعٍ لاستدعاء pinMode() لضبط المنفذ كمخرج قبل استدعاء التعليمة analogWrite(pin, value).

يوضح الشكل (3-51) الدارة العلمية للتحكم بسرعة المحرك المستمر DC باستخدام الدارة المتكاملة L293D. تشبه الدارة الموضحة في الشكل (3-43) باستثناء أن الرجل الأولى للدارة المتكاملة (E 1,2) يتم وصلها إلى إحدى أرجل الأردوينو التي تولد إشارة PWM (الرجل 11) بدلاً من وصلها إلى 5V+.



الشكل (3-51): التحكم بسرعة المحرك المستمر DC باستخدام الدارة المتكاملة L293D.

يوضح الشكل (3-52) الدارة العلمية للتحكم بسرعة المحرك المستمر DC باستخدام لوحة L298N. تشبه الدارة الموضحة في الشكل (3-47) باستثناء أن الوصلة 7 (Jumper) تم إزالتها، ووصلها إلى إحدى أرجل الأردوينو التي تولد إشارة PWM (الرجل 11).



الشكل (3-52): التحكم بسرعة المحرك المستمر DC باستخدام لوحة L298N.

3-7-2-5-1-الكود البرمجي

فيما يلي الكود البرمجي للتحكم بسرعة المحرك DC سواء باستخدام الدارة المتكاملة L293D أو لوحة L298N وبشكل يتوافق مع الشكلين (3-51) و (3-52). يشبه الكود البرمجي هنا الكود البرمجي السابق (الفقرة 3-2-7-3) مع إضافة قيمة تتراوح من 0 وحتى 255 أثناء استدعاء البرنامج الفرعي الخاص بالتحكم بالمحرك. هذه القيمة من خلالها يتم التحكم بسرعة المحرك. في البرنامج الفرعي تم إضافة التعليمة analogWrite(11,speed) والتي من خلالها يتم توليد إشارة PWM على الرجل 11 التي تتصل مع رجل التفعيل، وبالتالي التحكم بسرعة المحرك.

في هذا المثال سنعمل على تحرك المحرك لليمين بنصف السرعة بالمقارنة مع المثال السابق لمدة خمس ثوانٍ، ولليسار بنصف السرعة لمدة خمس ثوانٍ، ومن ثم إيقافه لمدة خمس ثوانٍ.

```
void setup() {
  pinMode(9, OUTPUT);
  pinMode(10, OUTPUT);
}

void loop() {
  move_motor('L', 127);     // L:move left, 127: half speed
  delay(5000);
  move_motor('R', 127);     // R:move right, 127: half speed
  delay(5000);
  move_motor('S', 0);       // stop
  delay(5000);
}

void move_motor(char Direction, byte Speed)
{
analogWrite(11,Speed);    // speed:0....255
if (Direction=='L' )
{
 digitalWrite(9,HIGH);
 digitalWrite(10,LOW);
}
if (Direction=='R' )
{
 digitalWrite(9,LOW);
 digitalWrite(10,HIGH);
}
if (Direction=='S' )
{
 digitalWrite(9,LOW);
 digitalWrite(10,LOW);
}
}
```

3-7-3- المحركات الخطوية

تتميز المحركات الخطوية بأنها تتحرك على شكل خطوات متقطعة حيث يدور المحور بزاوية محددة مع كل نبضة كهربائية يتلقاها أحد ملفات المحرك من دون أية آلية تغذية عكسية feedback كما في محرك السيرفو، وهذا ما يجعل المحرك الخطوي أبسط وأقل تكلفة. تتغير زاوية دوران المحور تبعاً لبنية المحرك الداخلية ولكنها محصورة عموماً ما بين 0.9° حتى 90°. تبعاً لذلك تستخدم

المحركات الخطوية في التطبيقات التي تتطلب دقة في الحركة مثل طابعات 3D، والبلوتر plotter، ومحركات الأقراص الصلبة والليزرية، والماسح الضوئي scanner، وآلات CNC والروبوتات.

يتألف المحرك الخطوي من جزأين رئيسيين هما: الدوار rotor، والجزء الثابت stator. الدوار عبارة عن مغناطيس دائم، أما الجزء الثابت فيتألف من عدة ملفات تعمل كمغناطيس كهربائي عندما يمر بها تيار كهربائي. تسبب ملفات المغناطيس الكهربائي إلى أن ينحاز الجزء الدوار عندما يتم شحنها كهربائياً. يتم دفع الدوار بالتناوب مع كل ملف يمر من خلاله تيار كهربائي.

يتغير جهد تغذية المحرك كثيراً تبعاً لنوعه فقد يكون محصوراً ضمن المجال 3 V وحتى 10 V. و يتغير أيضاً التيار المستهلك في المحرك عموماً حسب المقاومة الخاصة بملفات المحرك.

يمكن إجمال مزايا المحرك الخطوي بما يلي :

- يقدم المحرك الخطوي دقة في تحديد الموقع حيث تصل دقة المحرك الخطوي الجيد إلى 3-5% وهذا الخطأ ليس تراكمياً من خطوة إلى الأخرى.
- المحرك الخطوي رخيص التكلفة وسهل الاستخدام.
- المحرك الخطوي له عمر استخدام طويل جداً.
- يحافظ المحرك الخطوي على مكانه عندما لا يتم تطبيق نبضات كهربائي.
- لا يتأذى المحرك الخطوي بزيادة التحميل overloading، ويتوقف عن العمل فقط.
- تمتاز المحركات الخطوية بعزم torque مرتفع عند السرعات المنخفضة.

مساوئ :

- يحدث اهتزاز الرنين Resonance إذا لم يتم التحكم بشكل صحيح.
- عزم منخفض عند السرعات العالية.
- من الصعوبة أن يعمل بسرعات عالية.
- فاعلية منخفضة Low Efficiency: تستهلك استطاعة أكثر مما تقدمه، لذلك تميل للعمل مع وجود حرارة.

يمكن تصنيف المحركات الخطوية إلى:

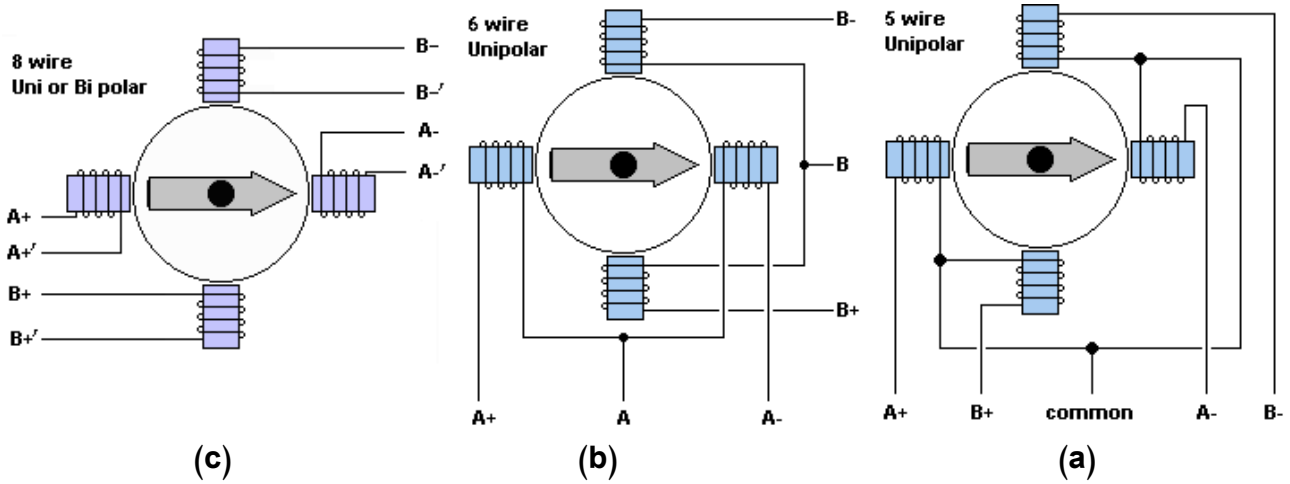
- محركات خطوية أحادية القطبية Unipolar.
- محركات خطوية ثنائية القطبية Bipolar.

سنتكلم في الفقرات التالية عن المحركات الخطوية التي تتحرك بخطوة 90 درجة للتبسيط، مع العلم أن المحركات الأخرى لن تختلف كثيراً عنها وخصوصاً أثناء قيادتها بلوحة الأردوينو.

3-7-3-1- المحرك الخطوي أحادي القطبية Unipolar Stepper Motor

يتحرك التيار في ملفات المحرك الخطوي أحادي القطبية باتجاه واحد لهذا تم تسميته بهذا الاسم. يتميز بأنه بسيط جداً، ومن مساوئه أن عزمه أقل لأنه لا يمكن تنشيط أكثر من نصف الملفات

بنفس الوقت. يخرج من المحرك خمسة أو ستة أسلاك أو ثمانية أسلاك. في المحرك خماسي الأسلاك ترتبط الملفات بأحد أطرافها معاً ويخرج منها سلك مشترك، والأسلاك الأربعة المتبقية لأطراف الملفات الأخرى. في المحرك سداسي الأسلاك يتشارك ملفان بأطرافهما بسلك مشترك، والملفان الآخران يتشاركان بسلك مشترك، والأسلاك الأربعة المتبقية لأطراف الملفات الأربعة الأخرى. عند وصل السلكين المشتركين معاً سنحصل على محرك خماسي الأسلاك. في المحرك ثماني الأسلاك تكون الملفات مستقلة وكل سلكين في الخرج يمثلان ملف من الملفات الأربعة. يوضح الشكل (3-53) محركات خطوية أحادية القطبية بأربعة ملفات خماسية وسداسية وثمانية الأسلاك.



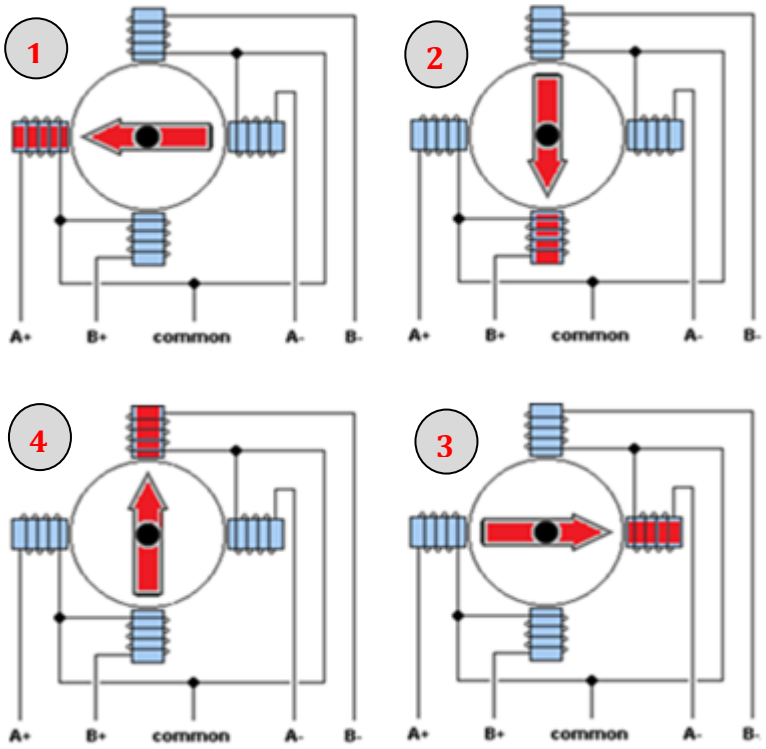
الشكل (3-53): محركات خطوية أحادية القطبية بأربعة ملفات: (a) خماسية الأسلاك، (b) سداسية الأسلاك، (c) ثمانية الأسلاك.

يتم تنشيط الملفات الأربعة بطرق مختلفة هي:

- إثارة ملف واحد Wave drive or Single-Coil Excitation.
- خطوة كاملة Full step drive.
- نصف خطوة Half stepping.
- خطوة دقيقة Micro-stepping.

طريقة إثارة ملف واحد: في هذه الطريقة يتم تطبيق جهد على ملف واحد (تنشيط ملف واحد) في كل مرة. نادراً ما تستخدم هذه الطريقة لأنها تقدم أقل من نصف عزم الدوران الاسمي للمحرك، بالتالي فإن حمولة المحرك لا يمكن أن تكون كبيرة. عموماً تستخدم هذه الطريقة لحفظ الطاقة. يتم إتمام دورة كاملة على 4 مراحل. يبين الشكل (3-54) كيفية عمل هذه الطريقة، والقيم المطلوب تطبيقها على الملفات الأربعة في كل مرحلة.

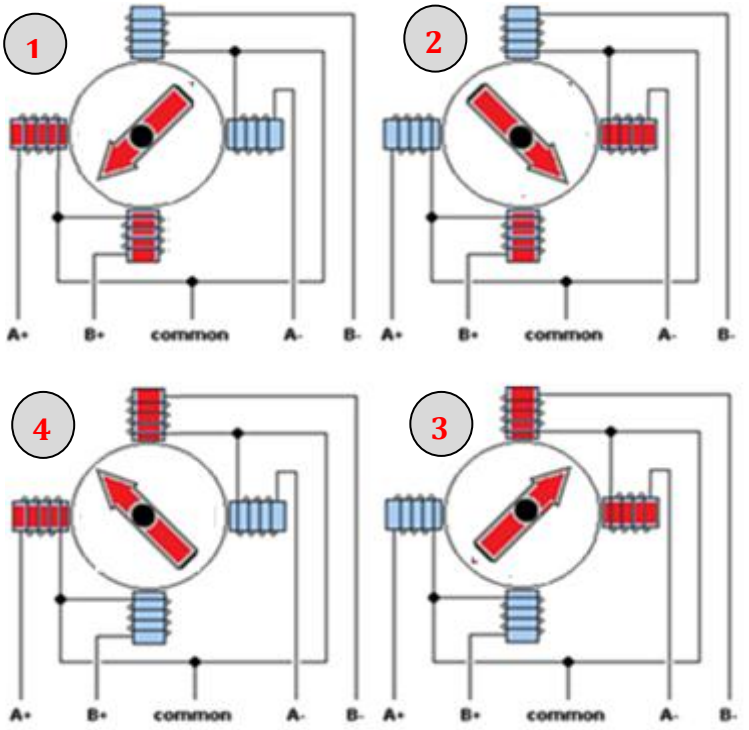
Step Number	A+	B+	A-	B-
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1



الشكل (3-54): تحريك المحرك الخطوي أحادي القطبية وفق طريقة إثارة ملف واحد.

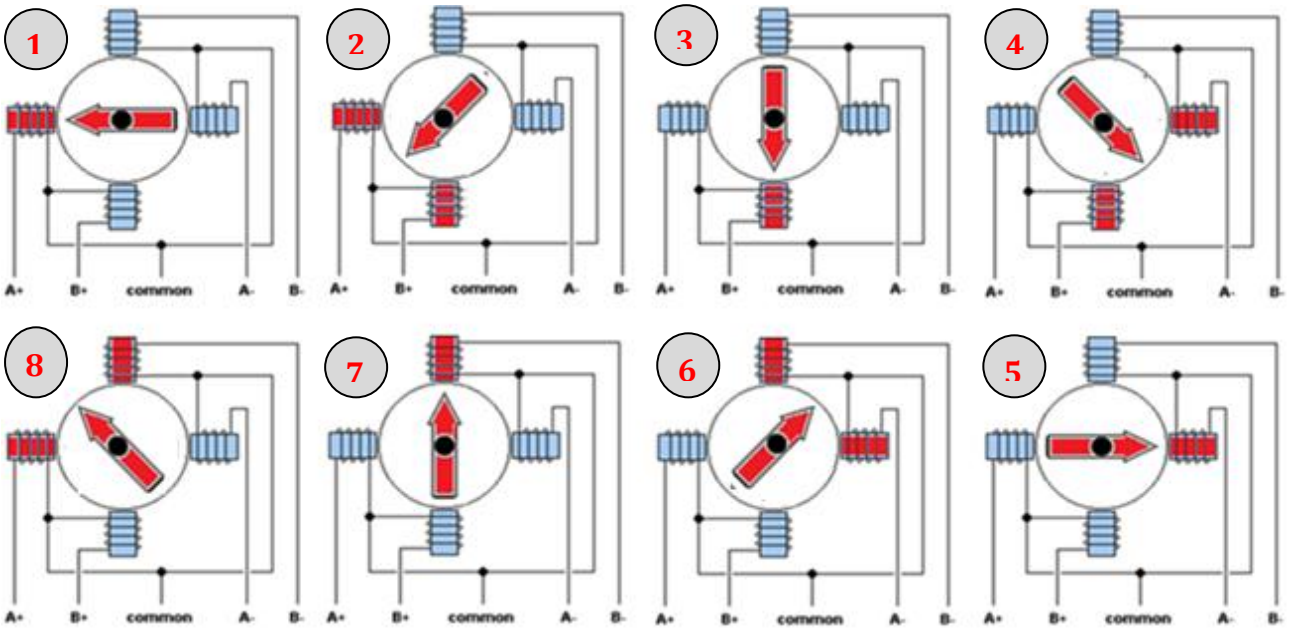
طريقة تحريك بخطوة كاملة: هذه الطريقة الأكثر استخداماً. يتم في هذه الطريقة تطبيق جهد على ملفين (أي تنشيط ملفين) في كل مرة، وهو ما يؤدي إلى الحصول على عزم اسمي كامل (100%) للمحرك. يتم إتمام دورة كاملة على 4 مراحل. يبين الشكل (3-55) كيفية عمل هذه الطريقة، والقيم المطلوب تطبيقها على الملفات الأربعة في كل مرحلة.

Step Number	A+	B+	A-	B-
1	1	1	0	0
2	0	1	1	0
3	0	0	1	1
4	1	0	0	1



الشكل (3-55): تحريك المحرك الخطوي أحادي القطبية وفق طريقة خطوة كاملة.

طريقة تحريك بنصف خطوة: تسمح هذه الطريقة بالحصول على دقة موقع مضاعفة من دون أي تغيير في التركيبية المادية للمحرك، حيث يتحرك المحرك بمقدار نصف خطوة بالمقارنة مع الطريقتين السابقتين. يتم إتمام دورة كاملة على 8 مراحل. يبين الشكل (3-56) كيفية عمل هذه الطريقة، والقيم المطلوب تطبيقها على الملفات الأربعة في كل مرحلة.



Step Number	A+	B+	A-	B-
1	1	0	0	0
2	1	1	0	0
3	0	1	0	0
4	0	1	1	0
5	0	0	1	0
6	0	0	1	1
7	0	0	0	1
8	1	0	0	1

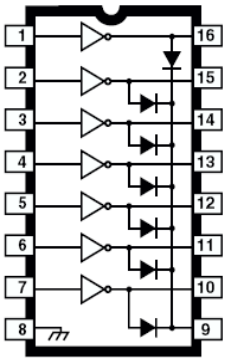
الشكل (3-56): تحريك المحرك الخطوي أحادي القطبية وفق طريقة نصف الخطوة.

طريقة Microstepping: لا يتم في هذه الطريقة تغذية الملفات بنبضات وإنما بإشارة تشبه إشارة sin. بهذا الأسلوب يتم الانتقال من خطوة إلى الأخرى بشكل أنعم، وهذا ما يجعل المحرك الخطوي مناسباً للتطبيقات ذات الدقة العالية مثل أنظمة CNC. يدور المحرك الخطوي في هذه الطريقة بشكل مستمر تقريباً مثل محركات DC البسيطة.

يستجر المحرك الخطوي -كما في محركات التيار المستمر- تياراً لا بأس به لا تستطيع لوحة الأردوينو تأمينه لهذا لابد من استخدام دارة عزل ما بين المتحكم و المحرك تؤمن له هذا التيار. هناك طرق عديدة يمكن بها تنفيذ دارة العزل: ترانزستورات أو دارات متكاملة مثل L293 أو UL2003. سنستخدم دارة ULN2003 التي تمتاز برخصها مقارنة مع L293.

3-7-3-1-1- الدارة المتكاملة ULN2003

تتكون دارة ULN2003 من سبع بوابات التي تتصف بأنها تعمل كمصب للتيار فقط. القيمة العظمى للتيار 500mA. يوضح الشكل (3-57) بنية دارة ULN2003 وتوزع أرجلها:



الشكل (3-57): البنية الداخلية لدارة ULN2003 ، وتوزع الأرجل.

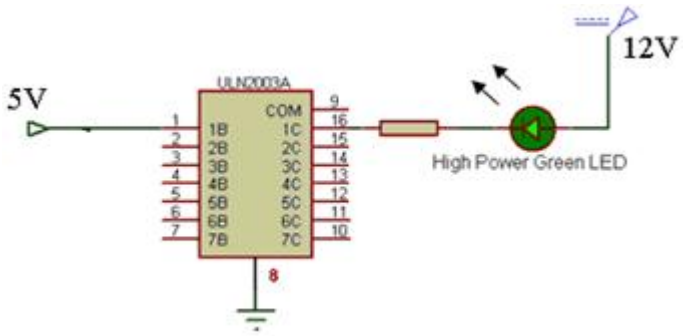
تعمل الأرجل من 1 و حتى 7 كمداخل لبوابات الدارة، والأرجل من 10 وحتى 16 كمخارج للبوابات. الرجل رقم 8 يتم وصلها إلى الأرضي المشترك. الرجل 9 تتصل مع مهابط ثنائيات، من الممكن عدم وصل هذه الرجل، أو قد يتم وصلها مع تغذية المحرك كما سنرى لاحقاً. يوضح الجدول (3-15) كيفية عمل البوابة.

الجدول (3-15): طريقة عمل البوابة في دارة ULN2003

INPUT	OUTPUT
High	Low
Low	Z (high impedance)

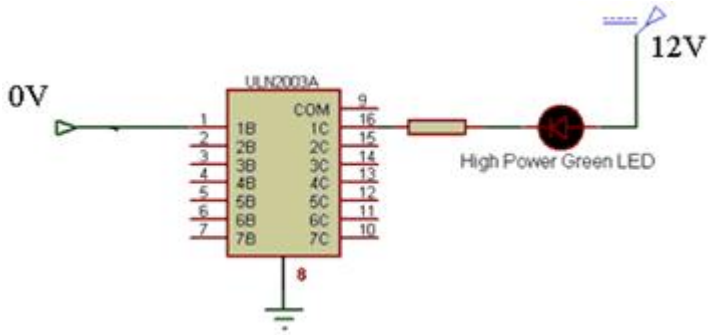
لإيضاح عمل بوابة دارة ULN2003 لندرس الحالتين التاليتين والتي سنتحكم فيها بثنائي ضوئي من خلال الجهد المطبق على مدخل البوابة الأولى.

1-بفرض أن الجهد المطبق على مدخل البوابة الأولى 5V (High level) كما هو موضح في الشكل (3-58). عندئذ فإن خرج البوابة الأولى على الرجل 16 هو 0V (Low level). تبعاً لذلك سيضيئ الثنائي الضوئي حيث عملت بوابة دارة ULN2003 على تأمين مصب للتيار إلى الأرضي .



الشكل (3-58): الجهد المطبق على مدخل البوابة الأولى 5V.

2- بفرض أن الجهد المطبق على مدخل البوابة الأولى 0V (Low level) كما هو موضح في الشكل (3-59). عندئذ فإن خرج البوابة الأولى على الرجل 16 هو ممانعة عالية high impedance. تبعاً لذلك لن يضيء الثنائي الضوئي حيث لم يتم تأمين مصب للتيار.



الشكل (3-59): الجهد المطبق على مدخل البوابة الأولى 0V.

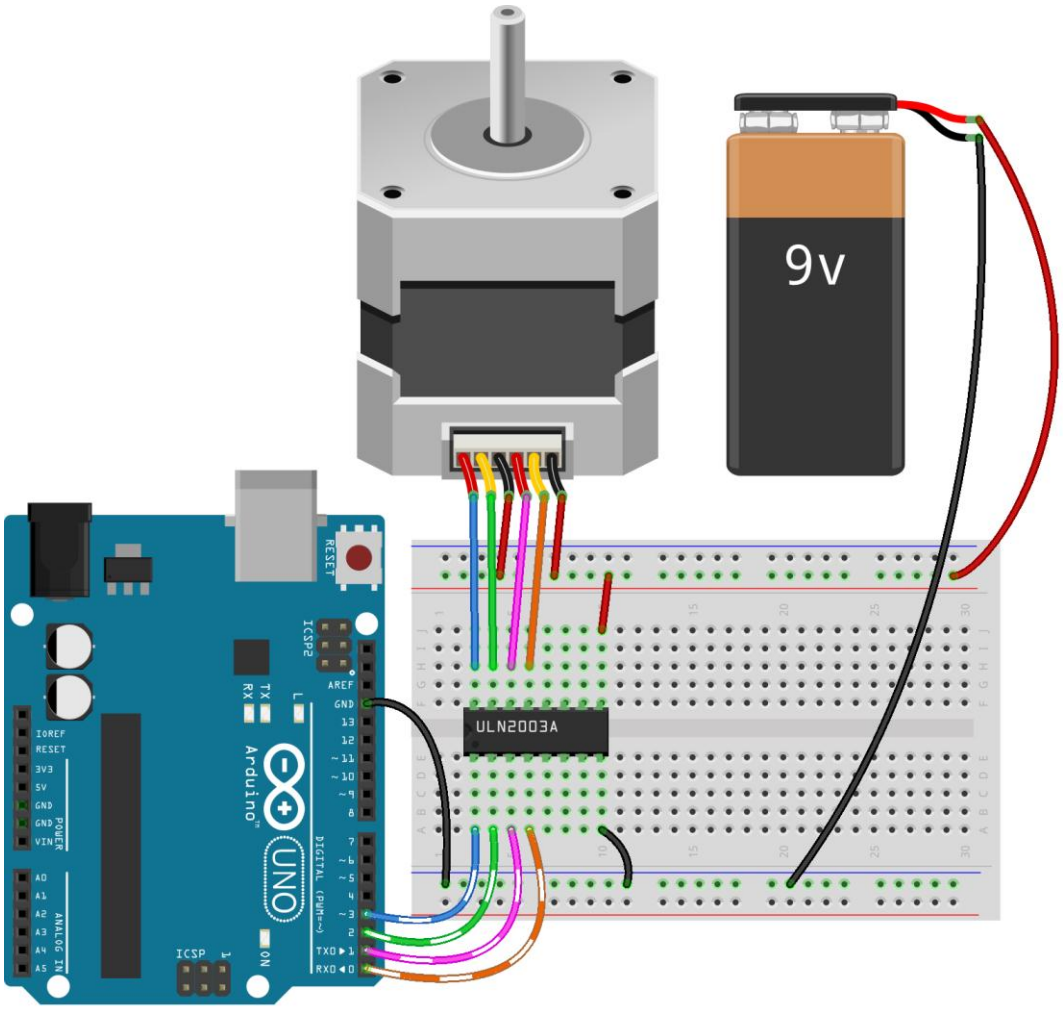
الجهد المطبق على الرجل 1 (مدخل البوابة الأولى) هو 0V (Low) و بالتالي الخرج على الرجل 16 (مخرج البوابة الأولى) هو High Z أي كأنها رجل عائمة . الثنائي الضوئي لا يعمل لأنه لم يتم تأمين مصب للتيار.

لا يمكن استخدام دارة ULN2003 مع المحرك المستمر لكي يتحرك باتجاهين وذلك لأنها تسمح بمرور التيار باتجاه واحد فقط وبالتالي فهي مناسبة للمحركات الخطوية أحادية القطبية.

3-7-3-2-1-التحكم بالمحرك الخطوي أحادي القطبية وسرعته من خلال لوحة الأردوينو

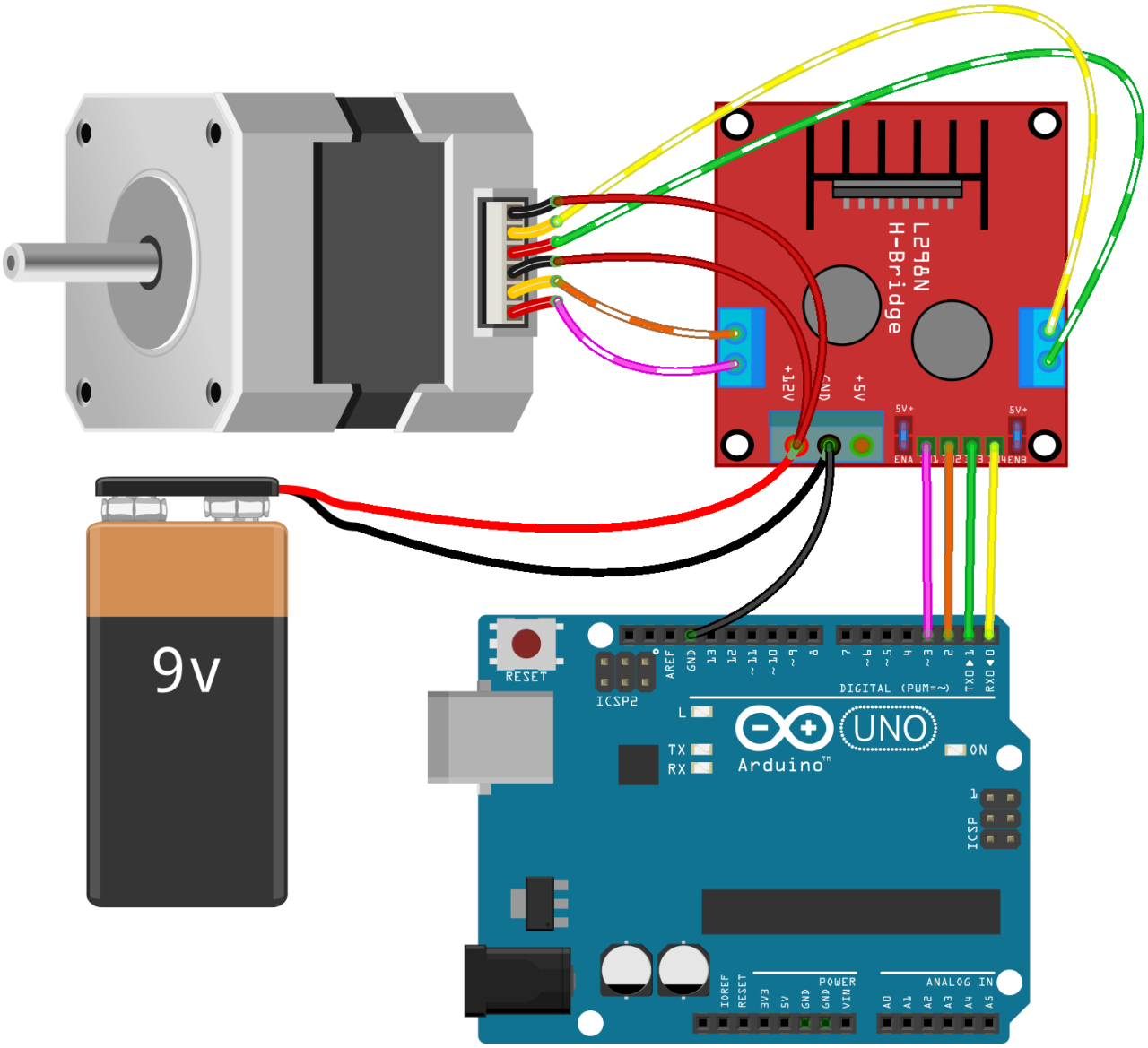
يبين الشكل (3-60) مخطط التوصيل العملي للوحة الأردوينو مع دارة القيادة ULN2003 والمحرك الخطوي أحادي القطبية سداسي الأسلاك. تم ربط المنافذ 0, 1, 2, 3 مع مداخل بوابات دارة ULN2003، والمخارج مع أربعة أسلاك للمحرك. السلكان المشتركان تم وصلهما إلى التغذية 9V+. تم وصل الرجل 8 لدارة ULN2003 مع الأرضي، والرجل 9 إلى التغذية 9V+ وذلك للتخلص من الحقل الكهربائي المغناطيسي العودي. عند تطبيق 5V+ (1 منطقي) على المنافذ 0, 1, 2, 3 سيكون خرج البوابات 0V مما يسمح بمرور تيار في ملفات المحرك، أما عند تطبيق 0V (0 منطقي) على المنافذ 0, 1, 2, 3 سيكون خرج البوابات ممانعة عالية ولن يمر تيار في الملفات. يتم اختيار إحدى طرق تنشيط الملفات المذكرة سابقاً (ملف واحد، خطوة كاملة، نصف خطوة) وبرمجة لوحة الأردوينو تبعاً لذلك.

يتم التحكم بجهة دوران المحرك (مع أو عكس عقارب الساعة) من خلال ترتيب تنشيط الملفات الواحد تلو الآخر، ويتم التحكم بسرعة المحرك من خلال تغيير التأخير الزمني ما بين كل مرحلة من مراحل الدورة.



الشكل (3-60): مخطط وصل لوحة الأردوينو مع المحرك الخطوي أحادي القطبية من خلال الدارة المتكاملة ULN2003.

يبين الشكل (3-61) مخطط التوصيل العملي للوحة الأردوينو مع لوحة L298N والمحرك الخطوي أحادي القطبية. تم ربط المنافذ 0, 1, 2, 3 مع أرجل الدخل المنطقية للوحة (IN1,IN2,IN3,IN4)، والمخارج (OUT1,OUT2,OUT3,OUT4) مع أربعة أسلاك للمحرك. السلكان المشتركان للمحرك تم وصلهما إلى التغذية 9V+ تم المحافظة على الوصلة 7 (jumper)، والوصلة 12 (jumper) بحيث يتم تفعيل جميع المخارج. تم تزويد اللوحة بمنبع تغذية 9V+ على الرجل 4 والذي من خلاله يتم تغذية المحرك. تم المحافظة على الوصلة 3 (jumper)، وهذا يعني أنه سيتم تفعيل منظم الجهد الداخلي للوحة 5V+ وبالتالي لا يوجد حاجة لتغذيتها بجهد 5V+. إذا تم نزع هذه الوصلة لابد من تأمين تغذية 5V+ على الرجل 6.



الشكل (3-61): مخطط وصل لوحة الأردوينو مع المحرك الخطوي أحادي القطبية من خلال لوحة L298N.

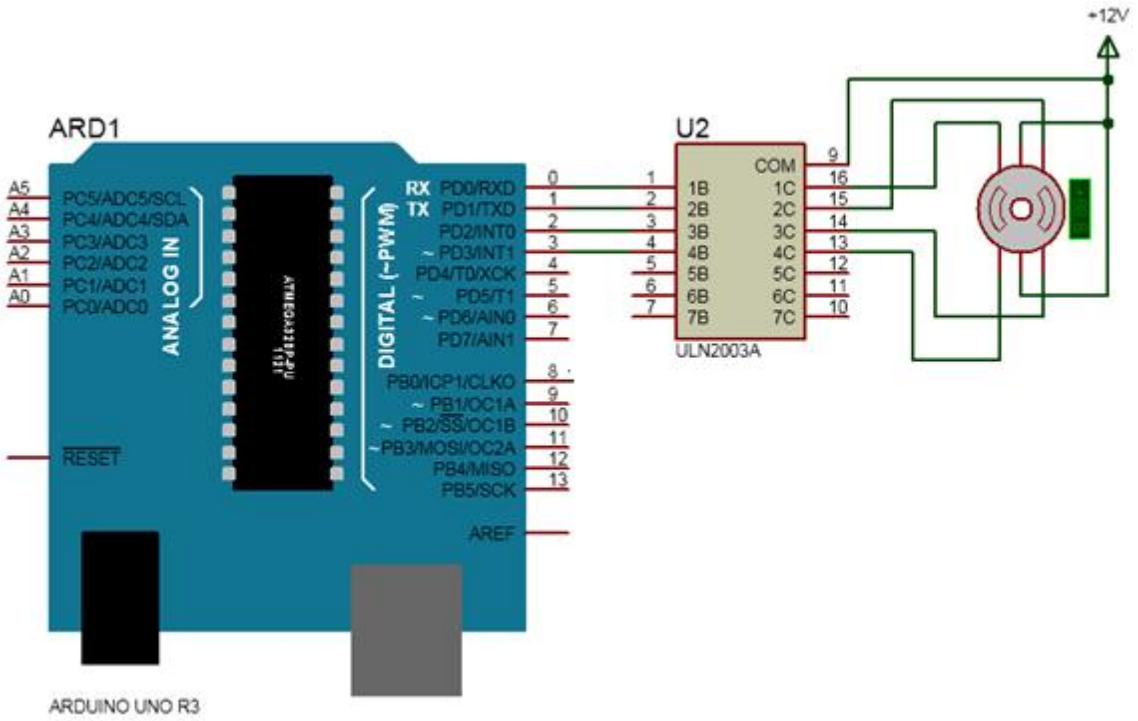
3-7-3-1-3-الكود البرمجي

تم كتابة الكود البرمجي الخاص بطريقة التحريك بخطوة كاملة، مع إمكانية اختيار جهة الدوران وسرعة المحرك. يصلح هذا الكود استخدامه مع دارة القيادة UL2003، ولوحة L298N.

```
void setup() {
  byte i;
  for (i = 0; i <= 7; i++)
  {
    pinMode(i, OUTPUT);
  }
}
void loop() {
  move_motor('L', 250); // move left, speed
  move_motor('R', 250); // move left, speed
```

```
  move_motor('L', 500); // move right, speed
  move_motor('R', 500); // move right, speed
}

void move_motor(char Direction, int Speed)
{
  if (Direction == 'L')
  { PORTA(0x03);
    delay(Speed);
    PORTA(0x06);
    delay(Speed);
    PORTA(0x0C);
    delay(Speed);
    PORTA(0x09);
    delay(Speed);
  }
  if (Direction == 'R')
  {
    PORTA(0x09);
    delay(Speed);
    PORTA(0x0C);
    delay(Speed);
    PORTA(0x06);
    delay(Speed);
    PORTA(0x03);
    delay(Speed);
  }
}

void PORTA(byte value)
{
  byte i;
  for (i = 0; i <= 7; i++)
  {
    digitalWrite(i, bitRead(value, i));
  }
}
```

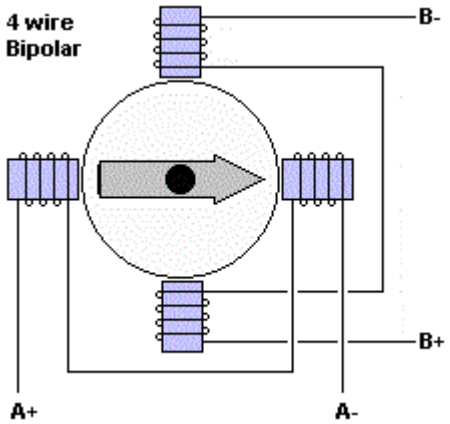
3-7-3-1-4-محاكاة التحكم بمحرك أحادي القطبية من خلال برنامج Proteus



الشكل (3-62): محاكاة التحكم بمحرك أحادي القطبية من خلال برنامج Proteus.

3-7-3-2- المحرك الخطوي ثنائي القطبية Bipolar Stepper Motor

في هذا النمط من المحركات الخطوية يرتبط ملفان داخلياً مع بعضهما بشكل تسلسلي، ويرتبط الملفان الآخران أيضاً مع بعضهما بشكل تسلسلي كما هو موضح في الشكل (3-63). تبعاً لهذا الوصل ستكون البينة الداخلية للمحرك الخطوي ثنائي القطبية عبارة عن ملفين. تمثل أطراف الملفين الأسلاك الأربعة في الخرج.



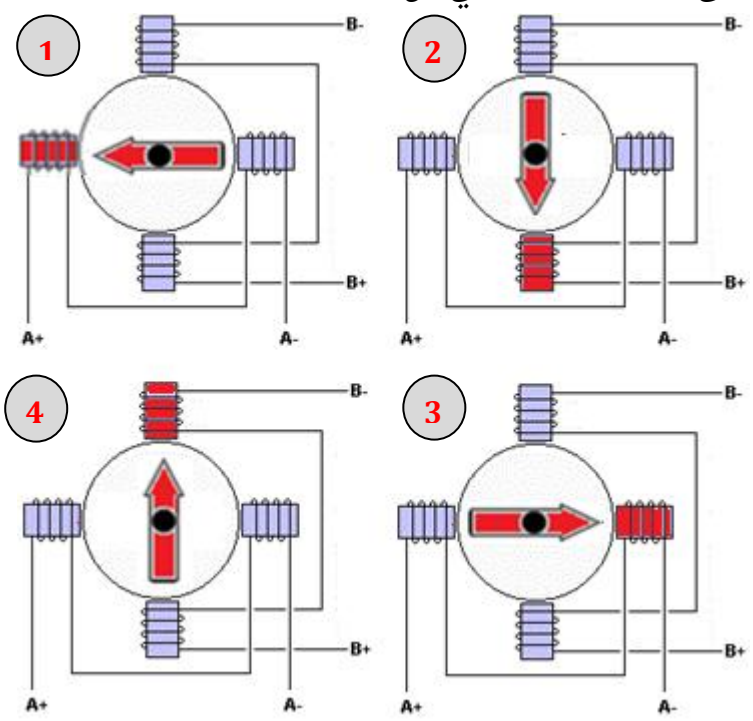
الشكل (3-63): محرك خطوي ثنائي القطبية ذو أربعة أسلاك في الخرج.

يتم تنشيط الملفات الأربعة لقيادة المحرك الخطوي ثنائي القطبية بنفس طريقة المحرك أحادي القطبية المذكور سابقاً:

- إثارة ملف واحد Wave drive or Single-Coil Excitation.
- خطوة كاملة Full step drive.
- نصف خطوة Half stepping.
- خطوة دقيقة Micro-stepping.

طريقة إثارة ملف واحد: في كل مرة يتم تطبيق جهد على ملف واحد، وبقية الملفات يطبق عليها 0V. عزم الدوران أقل من بقية الطرق، بالتالي فإن حمولة المحرك لا يمكن أن تكون كبيرة. يتم إتمام دورة كاملة على 4 مراحل. يبين الشكل (3-64) كيفية عمل هذه الطريقة، والقيم المطلوب تطبيقها على الملفات الأربعة في كل مرحلة.

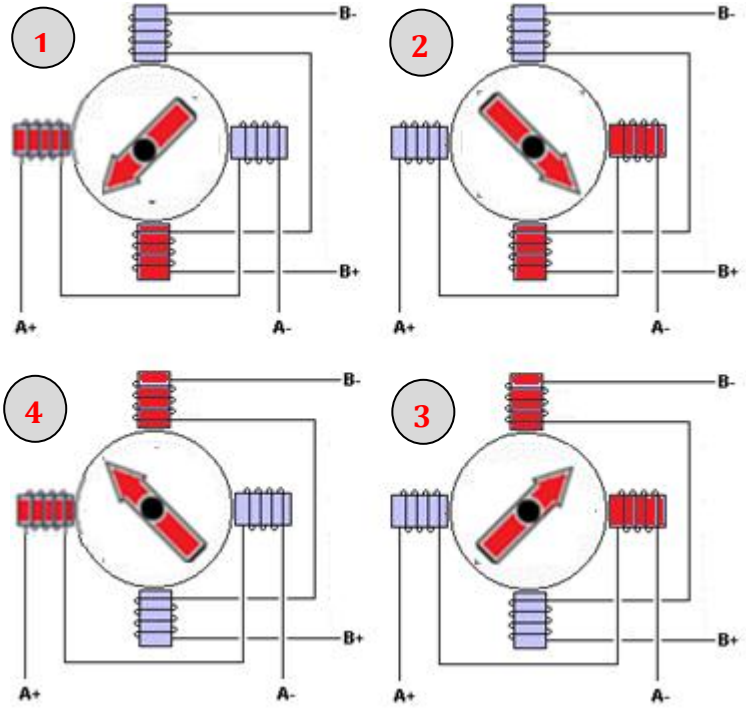
Step Number	A+	B+	A-	B-
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1



الشكل (3-64): تحريك المحرك الخطوي ثنائي القطبية وفق طريقة إثارة ملف واحد.

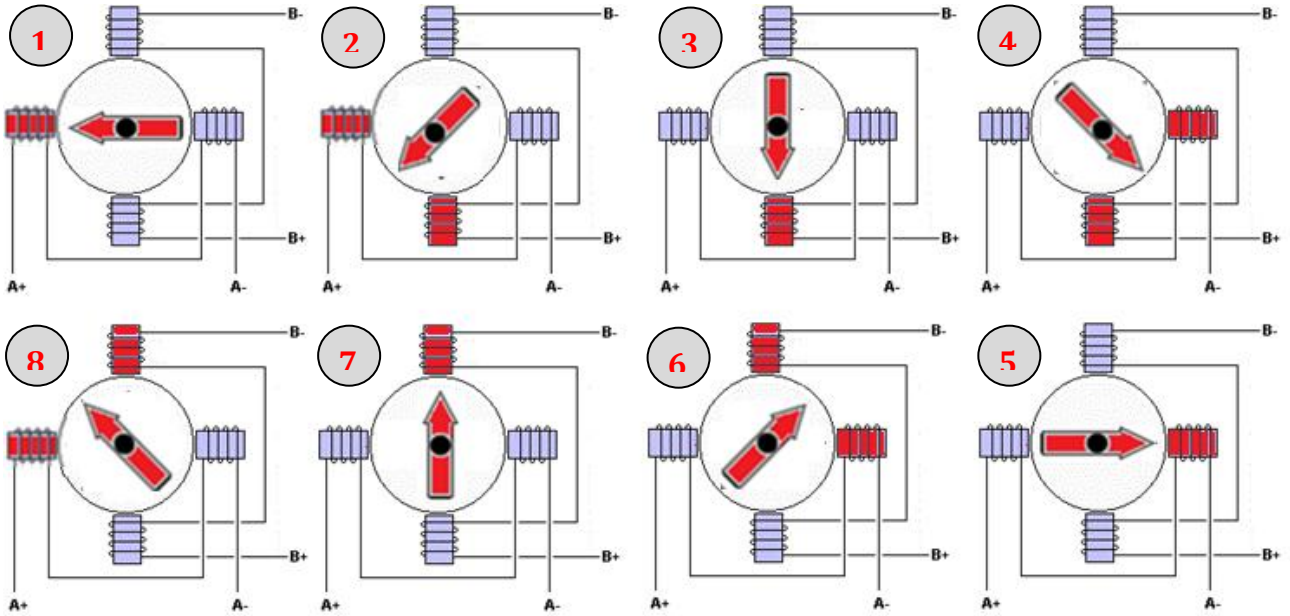
طريقة تحريك بخطوة كاملة: في كل مرة يتم تطبيق جهد على ملفين، والملفان الآخران يطبق عليهما 0V. عزم الدوران أعلى من الطريقة السابقة. يتم إتمام دورة كاملة على 4 مراحل. يبين الشكل (3-65) كيفية عمل هذه الطريقة، والقيم المطلوب تطبيقها على الملفات الأربعة في كل مرحلة.

Step Number	A+	B+	A-	B-
1	1	1	0	0
2	0	1	1	0
3	0	0	1	1
4	1	0	0	1



الشكل (3-65): تحريك المحرك الخطوي ثنائي القطبية وفق طريقة خطوة كاملة.

طريقة تحريك بنصف خطوة: تسمح هذه الطريقة بالحصول على دقة موقع مضاعفة من دون أي تغيير في التركيبية المادية للمحرك، حيث يتحرك المحرك بمقدار نصف خطوة بالمقارنة مع الطريقتين السابقتين. يتم إتمام دورة كاملة على 8 مراحل. يبين الشكل (3-66) كيفية عمل هذه الطريقة، والقيم المطلوب تطبيقها على الملفات الأربعة في كل مرحلة.



Step Number	A+	B+	A-	B-
1	1	0	0	0
2	1	1	0	0
3	0	1	0	0
4	0	1	1	0
5	0	0	1	0
6	0	0	1	1
7	0	0	0	1
8	1	0	0	1

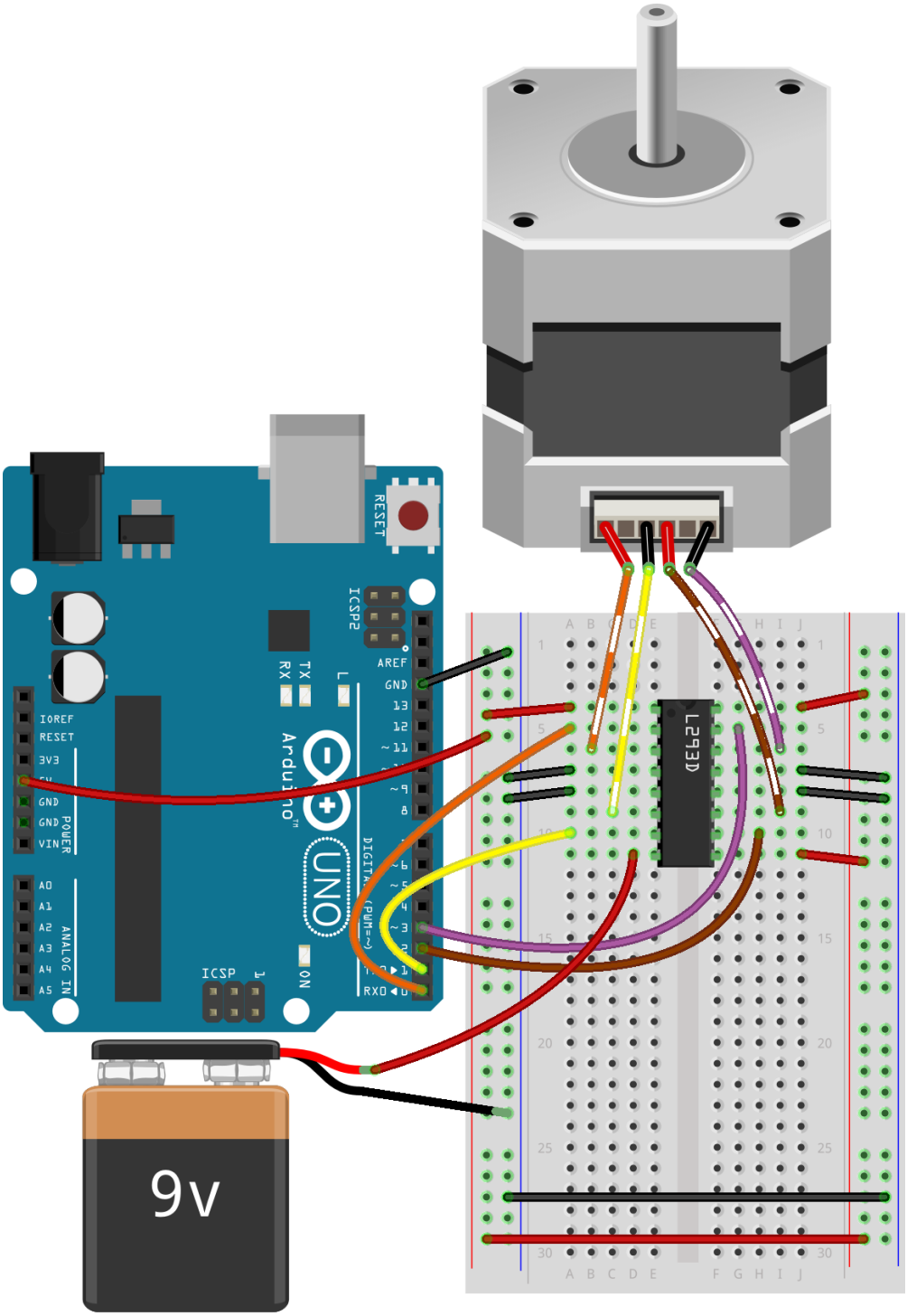
الشكل (3-66): تحريك المحرك الخطوي ثنائي القطبية وفق طريقة نصف الخطوة.

طريقة Microstepping: لا يتم في هذه الطريقة تغذية الملفات بنبضات وإنما بإشارة تشبه إشارة sin. بهذا الأسلوب يتم الانتقال من خطوة إلى الأخرى بشكل أنعم، وهذا ما يجعل المحرك الخطوي مناسباً للتطبيقات ذات الدقة العالية.

3-7-2-3-1-التحكم بالمحرك الخطوي ثنائي القطبية وسرعته من خلال لوحة الأردوينو

لا بد من استخدام دارة عزل ما بين لوحة الأردوينو والمحرك ثنائي القطبية لتأمين التيار المطلوب. لا يمكن استخدام الدارة المتكاملة ULN2003 مع المحرك ثنائي القطبية لأنها تعمل كمصب للتيار فقط (أي تسمح بمرور التيار باتجاه واحد)، في حين الدارة المتكاملة L293 والدارة L298N ستفي بالغرض المطلوب منها، وقد تم شرحهما سابقاً في الفقرتين (3-7-2-1)، (3-7-2-2).

يبين الشكل (3-67) مخطط التوصيل العملي للوحة الأردوينو مع دارة L293 والمحرك الخطوي ثنائي القطبية. تم ربط المنافذ 0, 1, 2, 3 مع مداخل بوابات دارة L293، والمخارج مع أربعة أسلاك للمحرك. يتم اختيار إحدى طرق تنشيط الملفات المذكرة سابقاً (ملف واحد، خطوة كاملة، نصف خطوة) وبرمجة لوحة الأردوينو تبعاً لذلك. يتم التحكم بجهة دوران المحرك (مع أو عكس عقارب الساعة) من خلال ترتيب تنشيط الملفات الواحد تلو الآخر، ويتم التحكم بسرعة المحرك من خلال تغيير التأخير الزمني ما بين كل مرحلة من مراحل الدورة.

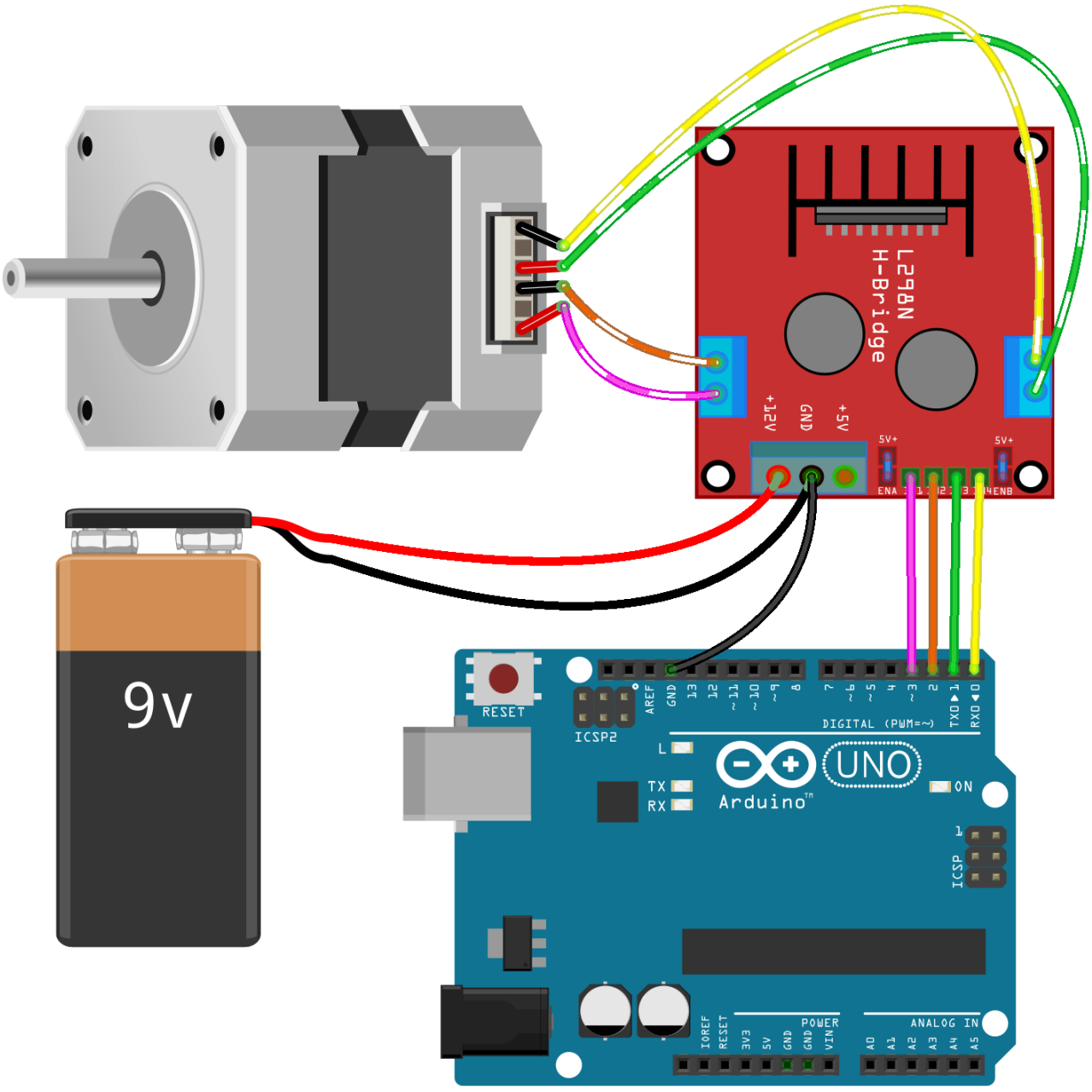


الشكل (3-67): مخطط وصل لوحة الأردوينو مع المحرك الخطوي ثنائي القطبية من خلال الدارة المتكاملة L298D.

يبين الشكل (3-68) مخطط التوصيل العملي للوحة الأردوينو مع لوحة L298N والمحرك الخطوي ثنائي القطبية. تم ربط المنافذ 3 ,2 ,1 ,0 مع أرجل الدخل المنطقية للوحة (IN1,IN2,IN3,IN4)، والمخارج (OUT1,OUT2,OUT3,OUT4) مع أربعة أسلاك للمحرك. تم المحافظة على الوصلة 7 (jumper)، والوصلة 12 (jumper) بحيث يتم تفعيل جميع المخارج. تم تزويد اللوحة بمنبع تغذية 9V+ على الرجل 4 والذي من خلاله يتم تغذية المحرك. تم المحافظة على

الوصلة 3 (jumper)، وهذا يعني أنه سيتم تفعيل منظم الجهد الداخلي للوحة 5V+ وبالتالي لا يوجد حاجة لتغذيتها بجهد 5V+. إذا تم نزع هذه الوصلة لابد من تأمين تغذية 5V+ على الرجل 6.

يتم اختيار إحدى طرق تنشيط الملفات المذكرة سابقاً (ملف واحد، خطوة كاملة، نصف خطوة) وبرمجة لوحة الأردوينو تبعاً لذلك. يتم التحكم بجهة دوران المحرك (مع أو عكس عقارب الساعة) من خلال ترتيب تنشيط الملفات الواحد تلو الآخر، ويتم التحكم بسرعة المحرك من خلال تغيير التأخير الزمني ما بين كل مرحلة من مراحل الدورة.



الشكل (3-68): مخطط وصل لوحة الأردوينو مع المحرك الخطوي ثنائي القطبية من خلال لوحة L298N.

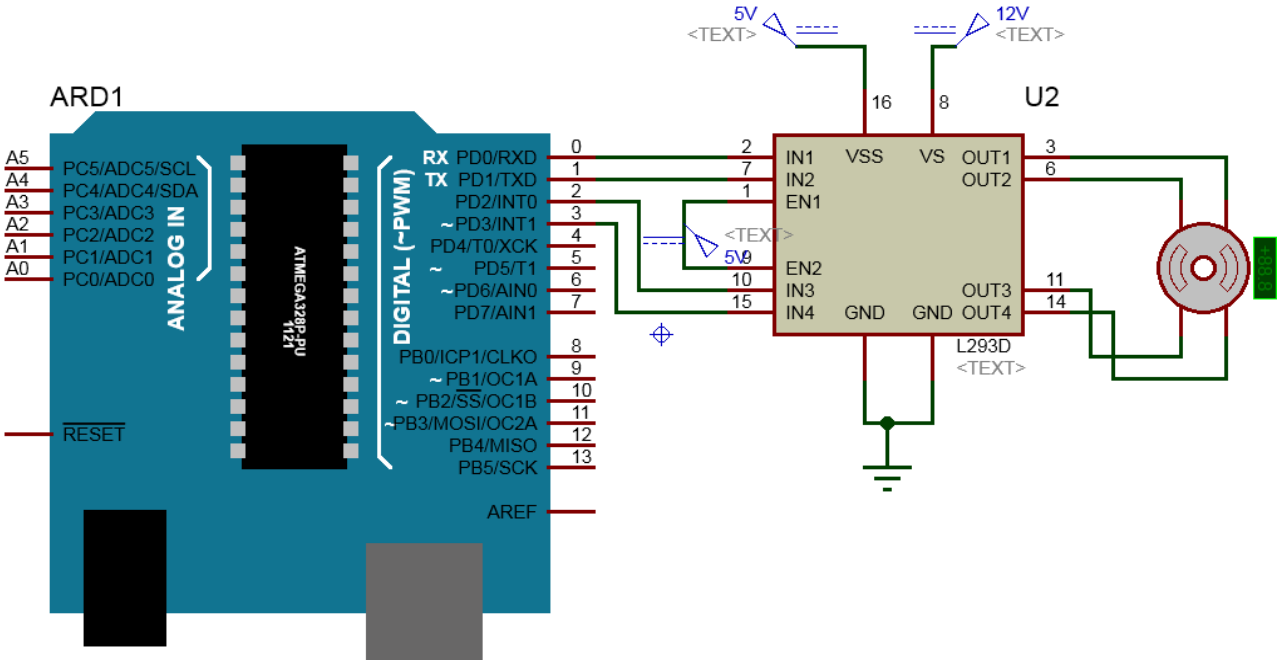
3-7-3-2-2-الكود البرمجي

تم كتابة الكود البرمجي الخاص بطريقة التحريك بخطوة كاملة، مع إمكانية اختيار جهة الدوران وسرعة المحرك. يصلح هذا الكود مع كل من الدارة المتكاملة L293، واللوحة L298N الموضحتين في الشكلين (3-67)، و(3-68).

```
void setup() {
 byte i;
 for (i = 0; i <= 7; i++)
 {
  pinMode(i, OUTPUT);
 }
}

void loop() {
 move_motor('L', 250); // move left, speed
 move_motor('R', 250); // move left, speed
 move_motor('L', 500); // move right, speed
 move_motor('R', 500); // move right, speed
}

void move_motor(char Direction, int Speed)
{
 if (Direction == 'L')
 { PORTA(B00000101);
  delay(Speed);
  PORTA(B00000110);
  delay(Speed);
  PORTA(B00001010);
  delay(Speed);
  PORTA(B00001001);
  delay(Speed);
 }
 if (Direction == 'R')
 { PORTA(B00000101);
  delay(Speed);
  PORTA(B00001001);
  delay(Speed);
  PORTA(B00001010);
  delay(Speed);
  PORTA(B00000110);
  delay(Speed);
 }
}
```

```
void PORTA(byte value)
{
 byte i;
 for (i = 0; i <= 7; i++)
 {
  digitalWrite(i, bitRead(value, i));
 }
}
```

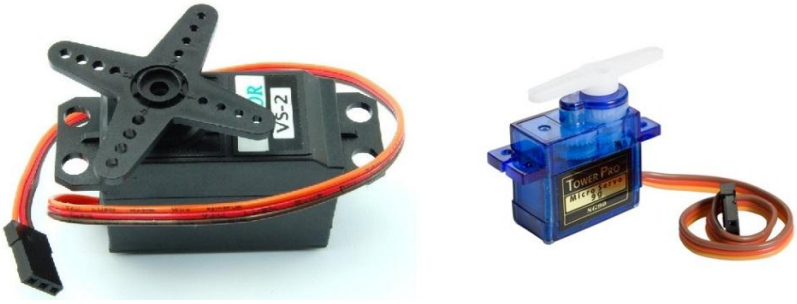
3-7-2-3-محاكاة التحكم بمحرك ثنائي القطبية من خلال برنامج Proteus.



الشكل (3-69): محاكاة التحكم بمحرك ثنائي القطبية من خلال برنامج Proteus.

3-7-4- محرك السيرفو Servo Motor

هو عبارة عن محرك مستمر DC يتم التحكم بزاوية دورانه بدقة من خلال نظام تحكم بحلقة تغذية عكسية مغلقة نموذجية. يدور محرك السيرفو فقط بقدر ما نريد ومن ثم يتوقف وينتظر الإشارة التالية بشكل يخالف المحرك المستمر DC التقليدي الذي يبدأ بالدوران عندما يتم تطبيق تغذية عليه ويستمر بالدوران حتى تتوقف التغذية عليه، بالتالي لا يمكننا التحكم بزاوية دورانه إلى مكان محدد. يوضح الشكل (3-70) نماذج مختلفة لمحركات سيرفو.



الشكل (3-70): أمثلة لمحركات سيرفو.

لمحرك السيرفو ثلاث أسلاك: اثنان لتزويده بالطاقة (ground، V_{cc})، والسلك الثالث يتم من خلاله تطبيق إشارة التحكم (نبضات PWM). في العادة لون سلك V_{cc} أحمر، ولون سلك الأرضي ground أسود أو بني، أما لون سلك إشارة التحكم أصفر أو أبيض أو برتقالي.

يتكون محرك السيرفو من الأجزاء التالية كما هو موضح في الشكل (3-71) :

1-محرك مستمر DC motor.

2-مقبض الخرج output shaft (ذراع محرك السرفو).

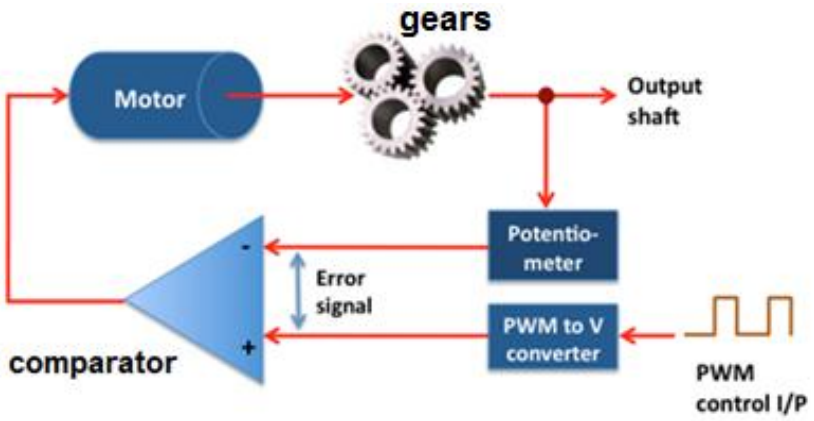
3-سلسلة من التروس gears التي تصل محرك DC بمقبض الخرج. الهدف منها تقليل سرعة محرك DC لتصبح مناسبة أكثر عملياً.

4-نظام تحكم الكتروني بمكان المقبض ذو تغذية عكسية مغلقة، ويتكون مما يلي:

أ-مقاومة متغيرة potentiometer: تتصل مع مقبض الخرج أو في مكان ما من سلسلة التروس، تقدم جهداً متناسباً مع مكان المقبض.

ب-مبدل نبضات تعديل عرض النبضة PWM إلى جهد.

ج-مقارن comparator: يعمل على مقارنة الجهد الناتج عن المقاومة المتغيرة مع الجهد الناتج عن المبدل.



الشكل (3-71): مخطط صندوقي لمحرك السيرفو.

يعمل محرك السيرفو كما يلي:

1-يطبق على المحرك إشارة تحكم على شكل إشارة تعديل عرض النبضة pulse width modulation (PWM) والتي ترددها 50Hz (أي تتكرر الإشارة كل $\frac{1}{50} = 20ms$). عرض هذه النبضات هو الذي سيتحكم بزاوية الدوران كما سنرى.

2-يتم تحويل هذه النبضات إلى جهد مكافئ عن طريق مبدل PWM. كلما كان عرض هذه النبضات أكبر كلما كان الجهد الناتج أكبر.

3-تقدم المقاومة المتغيرة جهداً يتناسب مع مكان مقبض الخرج.

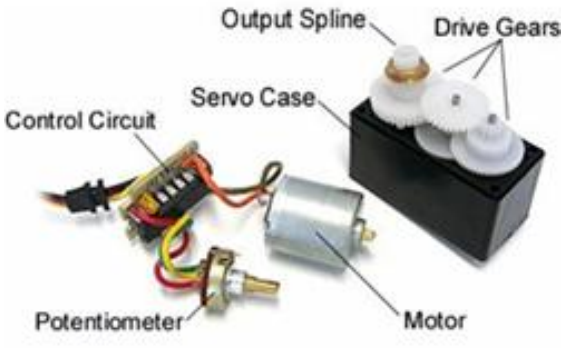
4-يعمل المقارن على مقارنة الجهد الناتج عن المبدل والجهد الناتج عن المقاومة المتغيرة. الاختلاف ما بين الإشارتين يعرف بإشارة خطأ error signal، والتي يتم تضخيمها، وتستخدم لقيادة محرك المستمر DC.

5-يتحرك محرك DC، ومقبض الخرج، وبالتالي الجهد الناتج عن المقاومة المتغيرة بحيث تؤدي العملية إلى إنقاص أو حذف إشارة الخطأ.

6-عندما يتم حذف إشارة الخطأ يكون مقبض الخرج قد وصل إلى المكان أو الزاوية المطلوبة، ويتوقف محرك DC تبعاً لذلك.

7-عند تغيير عرض نبضات PWM، سيتغير الجهد الناتج عن مبدل PWM وتنتج إشارة خطأ، ليتحرك المحرك ومقبض الخرج إلى زاوية أخرى مطلوبة، ويتوقف عندها.

يبين الشكل (3-72) العناصر الداخلية لمحرك السيرفو.



الشكل (3-72): العناصر الداخلية لمحرك السيرفو.

معظم محركات السيرفو تدور ضمن º180، وبعض منها ضمن º360. يتم التحكم بالزاوية الدقيقة من خلال عرض نبضات PWM ذات التردد 50Hz كما تم ذكره. تعتمد العلاقة ما بين عرض النبضة والزاوية على الشركات المصنعة للمحركات ولكن المبدأ الأساسي هو نفسه في كل محرك.

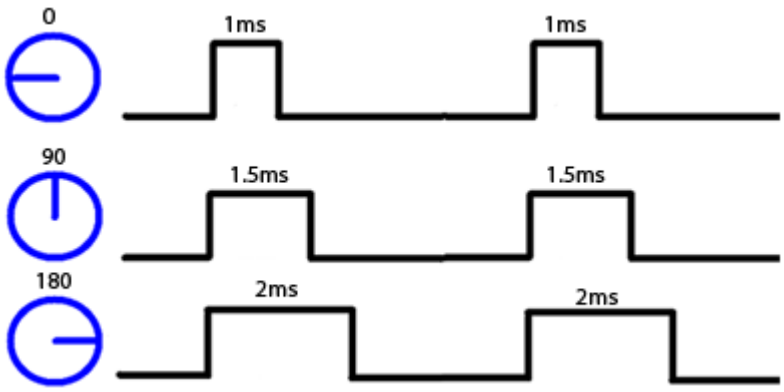
يوضح الشكل (3-73) مثالاً على العلاقة ما بين عرض النبضات وزاوية الدوران:

من أجل عرض نبضة مقداره 1ms تكون زاوية المحرك º0.

من أجل عرض نبضة مقداره 1.5ms تكون زاوية المحرك º90.

من أجل عرض نبضة مقداره 2ms تكون زاوية المحرك º180.

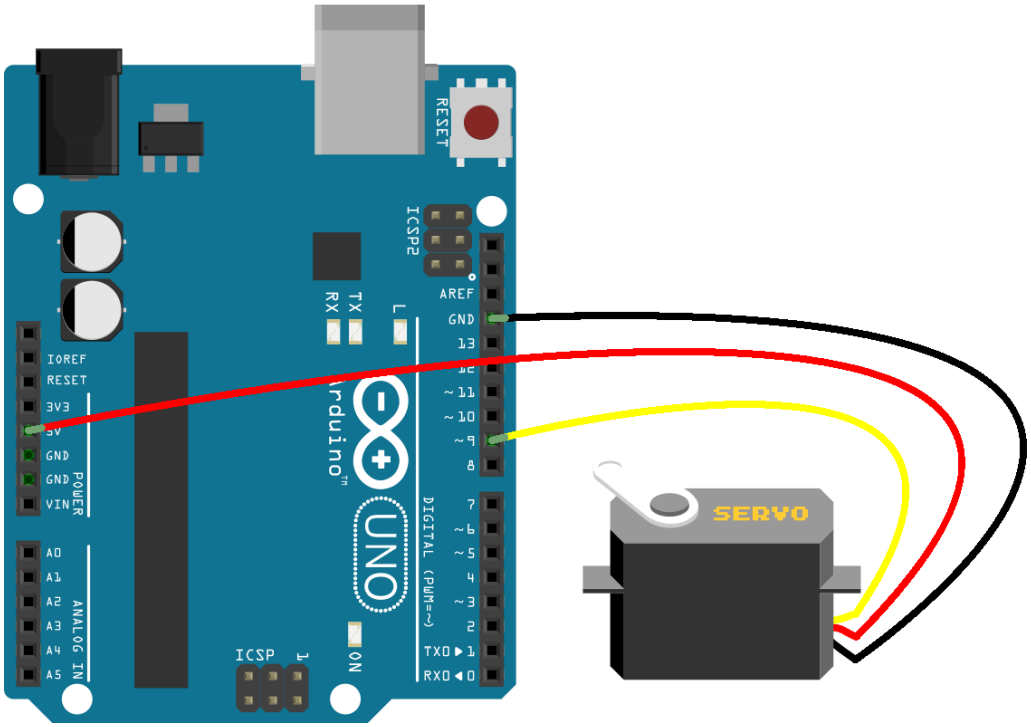
يتحرك محرك السيرفو ما بين 0 و180 درجة، ولا يتحرك في دوران مستمر كما في محرك DC التقليدي.



الشكل (3-73): من خلال التحكم بعرض نبضات إشارة PWM المطبقة على محرك السيرفو يتم ضبط زاوية دورانه.

3-7-4-1-التحكم بالمحرك السيرفو من خلال لوحة الأردوينو

يبين الشكل (3-74) مخطط التوصيل العملي للوحة الأردوينو مع محرك السيرفو بشكل مباشر. تم وصل سلك التغذية للمحرك V_{cc} (عادة لونه أحمر) إلى جهد التغذية للوحة 5V+، وتم وصل سلك الأرضي للمحرك (عادة أسود أو بني) إلى منفذ الأرضي للوحة، سلك إشارة التحكم (عادة لونه أصفر أو برتقالي أو أبيض) إلى الرجل رقم 9 للوحة والتي من خلالها يمكن توليد إشارة PWM.



الشكل (3-74): مخطط وصل لوحة الأردوينو مع المحرك الخطوي أحادي القطبية.

3-7-4-2-الكود البرمجي

يمكن استخدام التعليمة analogWrite(pin,value) لتوليد إشارة PWM كما تم ذكره سابقاً في الفقرة (3-2-7-3)، وبالتالي التحكم بمحرك السيرفو. يوجد مكتبة ضمن بيئة البرمجة Arduino IDE هي <Servo.h> تبسط عملية التحكم أكثر. فيما يلي تعليمات هذه المكتبة:

التعليمة	الوظيفة
Servo myservo;	تستخدم لإنشاء عنصر للتحكم بتعليمات محرك السيرفو. اسم العنصر myservo (ويمكن اختيار اسم آخر). يتم وضع هذه التعلمية بعد استدعاء المكتبة. مثال: Servo myservo;
myservo.attach(pin);	تحدد هذه التعليمة الرجل pin التي سيتم ربط محرك السيرفو معها. مثال: myservo.attach(9);
myservo.write(angle);	تعمل على ضبط زاوية محور المحرك تبعاً للبارامتر angle (بالدرجات من 0 وحتى 180). مثال: myservo.write(90); // set servo to mid-point
myservo. writeMicroseconds(uS);	يتم تحديد عرض نبضة إشارة PWM بواحدة الميكرو ثانية وبالتالي تحريك المحرك إلى زاوية محددة كما في الشكل (3-71). myservo.writeMicroseconds(1500); // set servo to mid-point
myservo.read()	قراءة الزاوية الحالية لمحرك السيرفو، ويعيد تبعاً لذلك قيمة من 0 وحتى 180.

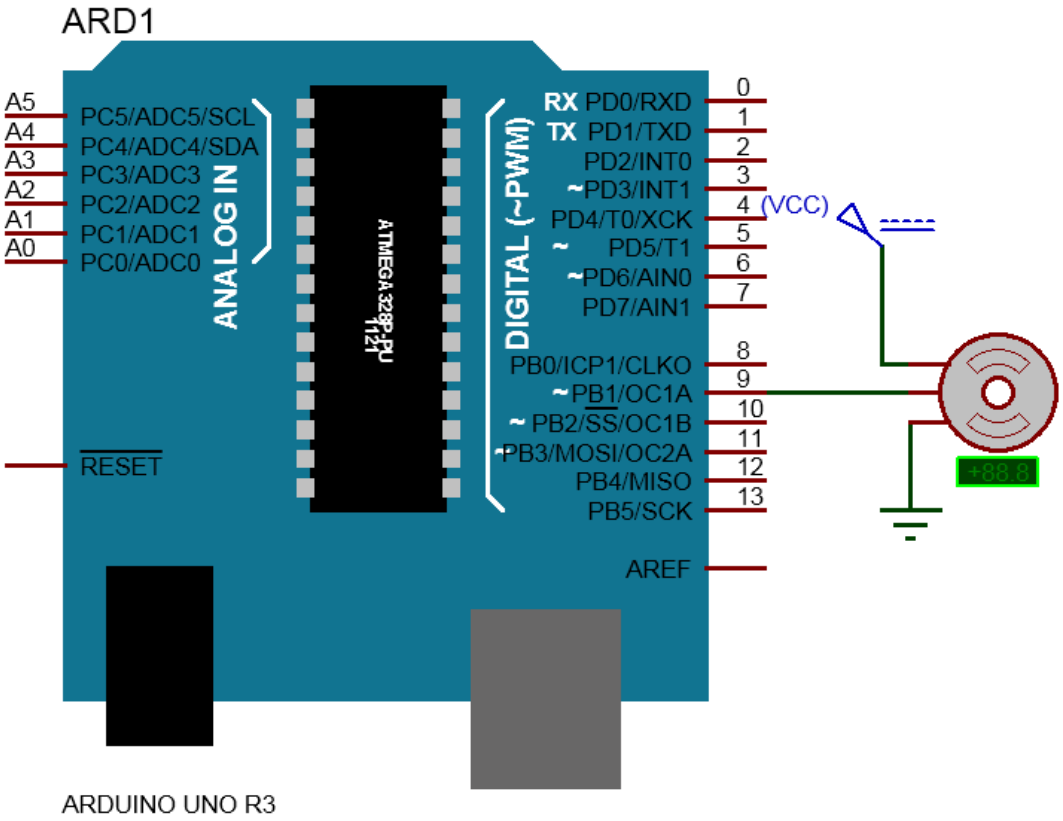
ملاحظة : اترك فاصلاً زمنياً مقداره 15ms على الأقل للانتقال من زاوية لزاوية أخرى.

تم كتابة كود برمجي لتحريك محرك السيرفو إلى الزوايا 180° ,90°, 0º.

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

void setup()
{
 myservo.attach(9);
}

void loop() {
 myservo.write(0);
 delay(1000);

 myservo.write(90);
 delay(1000);
```

```
  myservo.write(180);
  delay(1000);
}
```

3-4-7-3-محاكاة التحكم بمحرك السيرفو من خلال برنامج Proteus

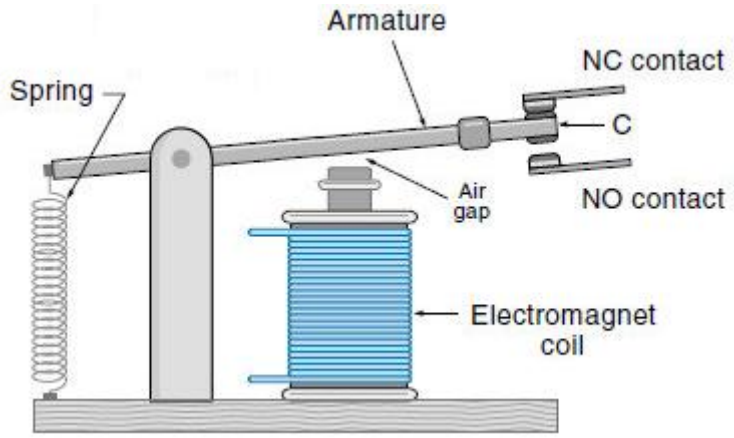


الشكل (3-75): محاكاة التحكم بمحرك السيرفو من خلال برنامج Proteus.

التحكم بالأجهزة التي تعمل بجهود عالية

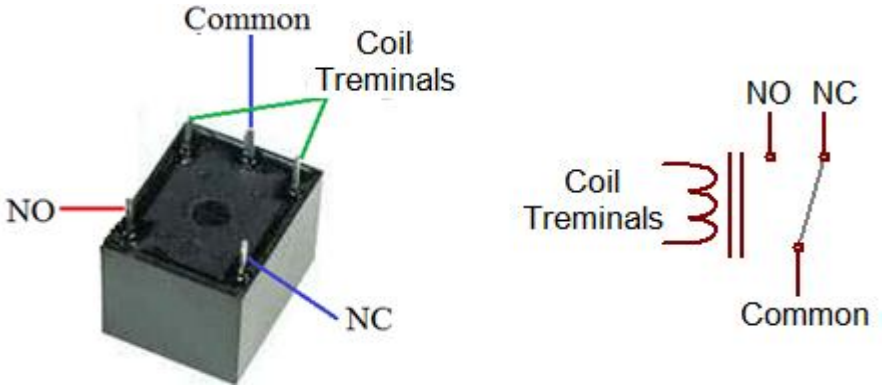
3-8-1-مقدمة

كما هو معلوم لدينا تعمل لوحة الأردوينو بجهد مقداره 5V+ لذلك لا يمكن لها أن تتحكم مباشرةً بأجهزة الجهد العالي (120-240V) كمصابيح الإضاءة، والمراوح، والسخانات، لتستخدم أداة تعرف بالمرحل relay التي تقدم اتصالاً ما بين دارة الجهد المنخفض مع دارة الجهد العالي. يتألف المرحل كما هو موضح في الشكل (3-76) من ملف coil، وعمود أو ذراع حديدي متحرك movable armature، وتماس ثابت مغلق بشكل طبيعي Normal close (NC) وتماس مفتوح بشكل طبيعي Normal Open (NO)، وقد يكون أكثر من تماسين ثابتين. يعمل الملف كمغناطيس كهربائي electromagnet. بدون تطبيق جهد كهربائي على طرفي الملف يكون الذراع المتحرك متصلاً مع التماس المغلق بشكل طبيعي NC. عند تطبيق جهد كهربائي منخفض نسبياً على طرفي الملف سيتولد حقل مغناطيسي يؤدي إلى جذب الذراع المتحرك إليه، ليتحرك تبعاً لذلك من تماس NC إلى تماس NO ويتصل معه. عند إزالة الجهد الكهربائي يعود الذراع المتحرك إلى موضع الاسترخاء وذلك من خلال نابض spring، ليتصل مع تماس NC من جديد. يتم ربط دارة الجهد المنخفض مع طرفي الملف، في المقابل يستطيع الذراع المتحرك والتماسين الثابتين تحمل تطبيق جهود مرتفعة نسبياً، لهذا يتم وصلها مع دارة الجهد العالي.



الشكل (3-76): بنية المرحل.

للمرحل استناداً لما سبق خمسة أطراف هي: طرفا الملف، وطرف مشترك (الذراع المتحرك)، وطرف NC، وطرف NO كما هو موضح في الشكل (3-77).



الشكل (3-77): أطراف المرحل العملية، والرمز الالكتروني المكافئ.

يوجد بعض المصطلحات الخاصة بالمرحل هي:

جهد الملف الاسمي أو المعياري (Nominal Coil Voltage (Rated Coil Voltage: الجهد الذي تم تصميم الملف من خلاله ليعمل به. تطبيق جهد أعلى سيكون مهدراً للطاقة ومن الممكن أن يتلف المرحل بسبب الحرارة أو الإجهاد الميكانيكي.

جهد المسك pickup voltage : هو أصغر جهد مطبق على الملف يتم من خلاله ضمان سحب الذراع المتحرك. إذا كانت قيمة الجهد المطبق أقل من جهد المسك فقد يتم سحب الذراع ولكن بشكل غير مضمون.

جهد الانسحاب Dropout voltage: أعلى جهد مطبق على الملف لا يتم به جذب الذراع المتحرك.

مثال: بفرض أن جهد الملف الاسمي 5 V، وجهد المسك 70% من جهد الملف الاسمي، وجهد الانسحاب 10% من جهد الملف الاسمي. حدد الجهود التي من خلالها يتم جذب التماس، وعدم جذبه بشكل مضمون به.

يمكن جذب الذراع المتحرك عند تطبيق جهد أعلى 3.5 V (70% of 5V)، ويمكن ضمان عدم جذب (ترك) الذراع المتحرك عند تطبيق جهد أقل من 0.5V (10% of 5V).

تيار التشغيل الاسمي Nominal Operating Current : قيمة التيار المتدفق في الملف عندما يتم تطبيق الجهد الاسمي عليه.

استطاعة التشغيل الاسمية Nominal Operating Power : قيمة الاستطاعة المستخدمة من قبل الملف عند تطبيق الجهد الاسمي.

مقاومة الملف Coil Resistance.

جهد التبديل الأعظمي Maximum Switching Voltage : أعظم جهد يمكن تطبيقه على التماسات بشكل آمن، وتختلف قيمته حالة كونه مستمر DC أو متناوب AC في معظم الحالات.

تيار التبديل الأعظمي Maximum Switching Current: أعظم تيار يمكن تطبيقه على التماسات بشكل آمن، وتختلف قيمته في حالة كونه مستمر DC أو متناوب AC في معظم الحالات.

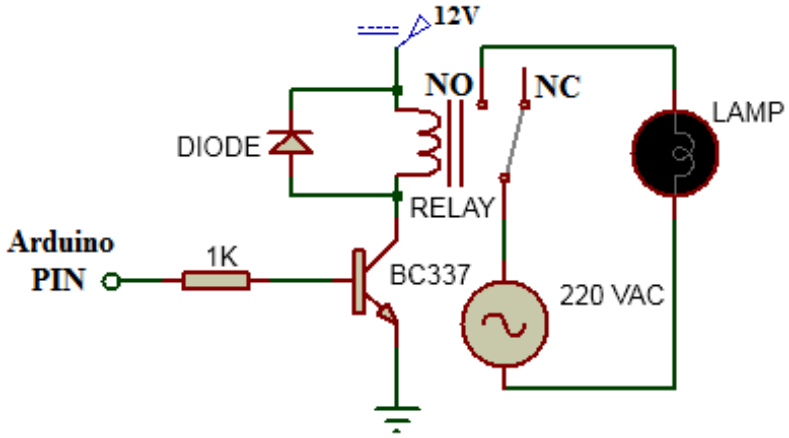
استطاعة التبديل العظمى **Maximum Switching Power :** الحد الأعلى للاستطاعة الممكن تبديلها من قبل التماسات.

يوضح الشكل (3-78) مثالاً على مرحل مع الخواص الخاصة به والتي تكتب عادة على الغلاف الخارجي. الجهد الاسمي له 12V، وجهد التبديل الأعظمي في حالة كونه متناوباً 240 VAC مع تيار تبدل أعظمي 15 A، وفي حالة كونه مستمراً 24VDC مع تيار أعظمي 15A.

الشكل(3-78): مثال على مرحل مع الخواص الخاصة به. في الدائرة الحمراء الجهد الاسمي، في الدائرة الخضراء أعظم جهد وتيار تبديل في حالة تطبيق إشارة متناوبة، في الدائرة الصفراء أعظم جهد وتيار تبديل عند تطبيق إشارة مستمرة.

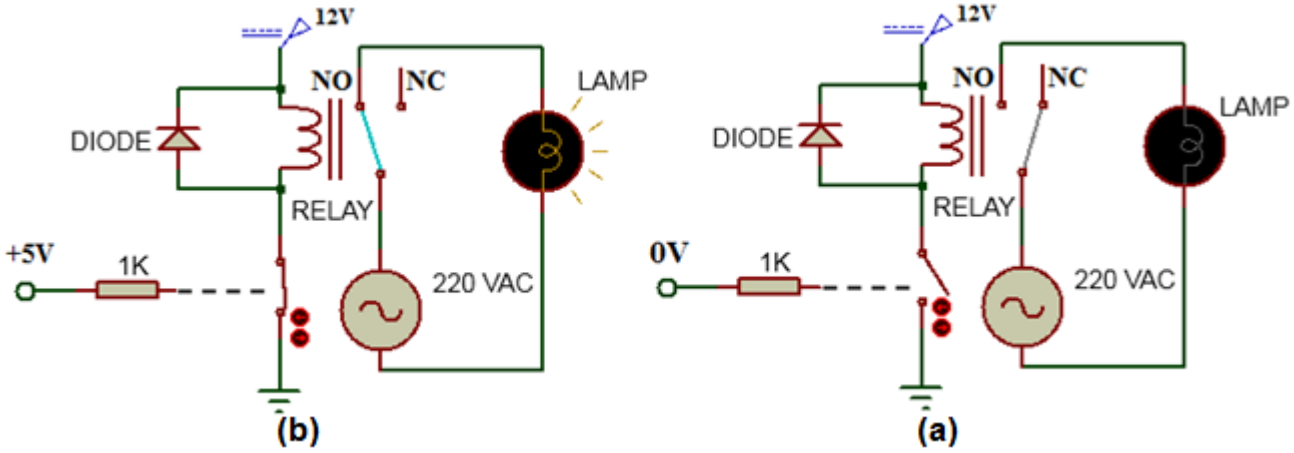
3-8-2-ربط المرحل مع لوحة الأردوينو

تم التوصل من خلاله الفقرة السابقة إلى أن دارة الجهد المنخفض يتم وصلها مع طرفي الملف، ودارة الجهد العالي مع الذراع المتحرك، والتماسات الثابتة. الجهد المطبق على طرفي الملف قد يكون 6V أو 9V أو 12V أو 24V، لهذا لا يمكن وصل أحد أقطاب الأردوينو مع الملف مباشرة، أضف إلى ذلك إلى أن تيار التشغيل الاسمي لا يمكن تأمينه في معظم الحالات عن طريق قطب الأردوينو. لهذا لابد من استخدام أداة الكترونية تؤمن الجهد والتيار ويتم التحكم بها عن طريق الأردوينو كما تم دراسته في فقرة المحركات. أبسط أداة يمكن استخدامها هي الترانزستور. يوضح الشكل (3-79) كيفية وصل لوحة الأردوينو مع طرفي ملف المرحل عن طريق الترانزستور. وكيفية وصل دارة الجهد العالي المتمثلة بمنبع جهد متناوب 220V ومصباح LAMP مع الذارع المتحرك والتماسات الثابتة.



الشكل (3-79): وصل لوحة الأردوينو مع المرحل لتشغيل أجهزة الجهد العالي.

يعمل الترانزستور كمفتاح switch، بدون تطبيق جهد (0 منطقي) على قاعدة الترانزستور يكون في حالة فتح OFF وتصبح الدارة المكافئة له كما هو موضح في الحالة (3-80-a)، وعندئذ لن يمر تيار كهربائي ضمن ملف المرحل، وبالتالي سيكون الذراع متصل مع التماس NC، وتكون دارة الجهد العالي مفتوحة ولن يعمل المصباح تبعاً لذلك. عند تطبيق جهد مقداره 5V+ عن طريق أحد أرجل لوحة الأردوينو على قاعدة الترانزستور يصبح في هذه الحالة بحالة تشغيل ON وتصبح الدارة المكافئة كما هو موضح في الحالة (3-80-b)، ليمر تيار ضمن ملف المرحل، وسيجذب الذراع إليه، مما يؤدي إلى أن يتصل مع التماس NO، بالتالي ستغلق دارة الجهد العالي وسيعمل المصباح عندئذ. يمكن استخدام ترانزستور BC548 أو BC337. يمكن ملاحظة وجود ثنائي diode على طرفي ملف المرحل، وهو ضروري لأنه عندما إزالة الجهد المطبق على طرفي الملف سيحدث ارتفاع مفاجئ للجهد (voltage spike) بسبب التيار الذي يحتفظ به الملف، مما قد يؤدي إلى تلف الترانزستور أو قد يقلل من عمر تماسات التبديل. قد يصل هذا الجهد إلى 1000V من أجل مرحل 12V. يوفر هذا الثنائي مساراً للتيار المخزن في الملف، وبالتالي تبديد وتفريغ الطاقة المغناطيسية بأمان.

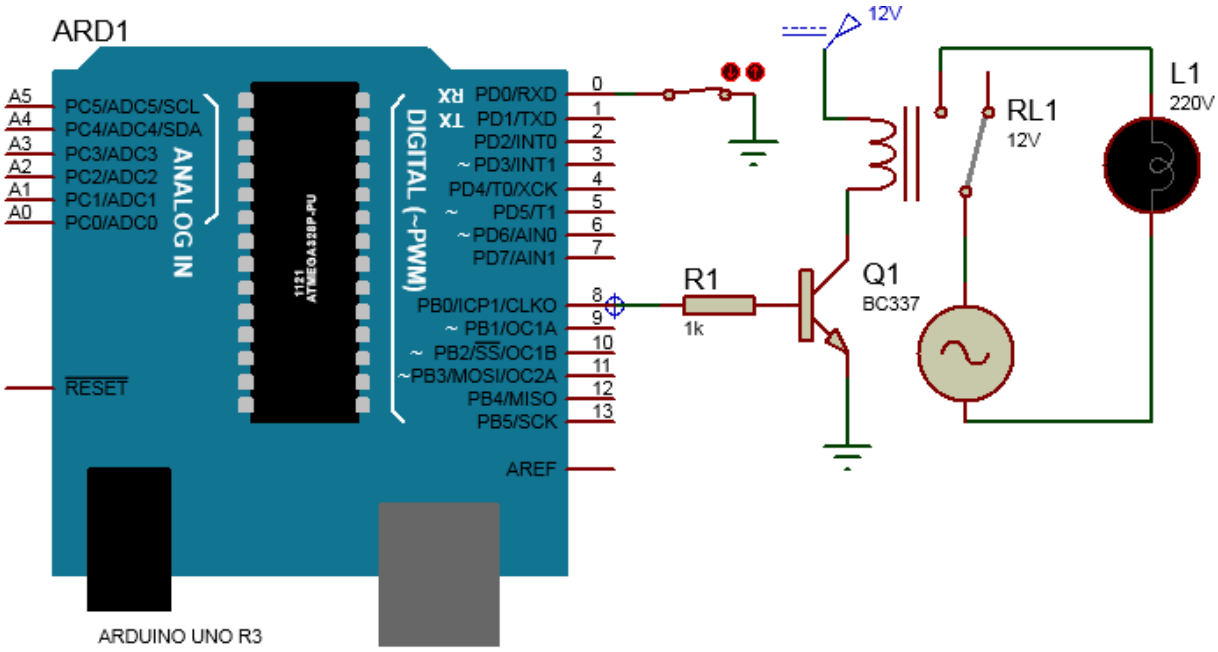


الشكل (3-80): الحالة (a) عند تطبيق 0V لا يعمل المصباح، الحالة (b) عند تطبيق 5V+ يعمل المصباح.

3-8-3-الكود البرمجي

```
void setup() {
pinMode(0,INPUT_PULLUP);
pinMode(8,OUTPUT);
}

void loop() {
boolean x;
x=digitalRead(0);
if (x==LOW) {digitalWrite(8,HIGH);}
else {digitalWrite(8,LOW);}
}
```

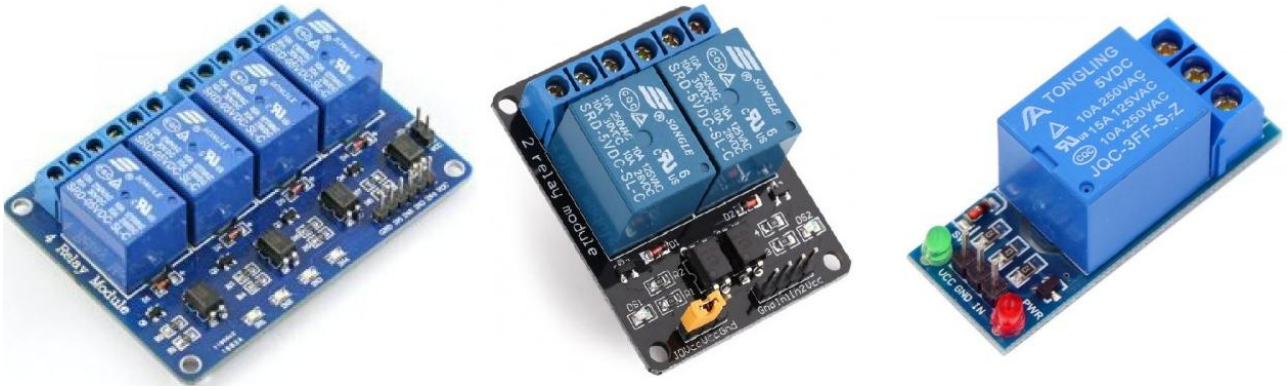
3-8-4- محاكاة التحكم بمصباح باستخدام المرحل في برنامج Proteus



الشكل (3-81): محاكاة التحكم بمصباح باستخدام المرحل في برنامج Proteus.

3-8-5- ملحقات

يوضح الشكل (3-82) بعض الدرات الجاهزة التي تحتوي على مرحل، ويتم وصلها مع لوحة الأردوينو ودارة الجهد العالي بشكل مباشر.

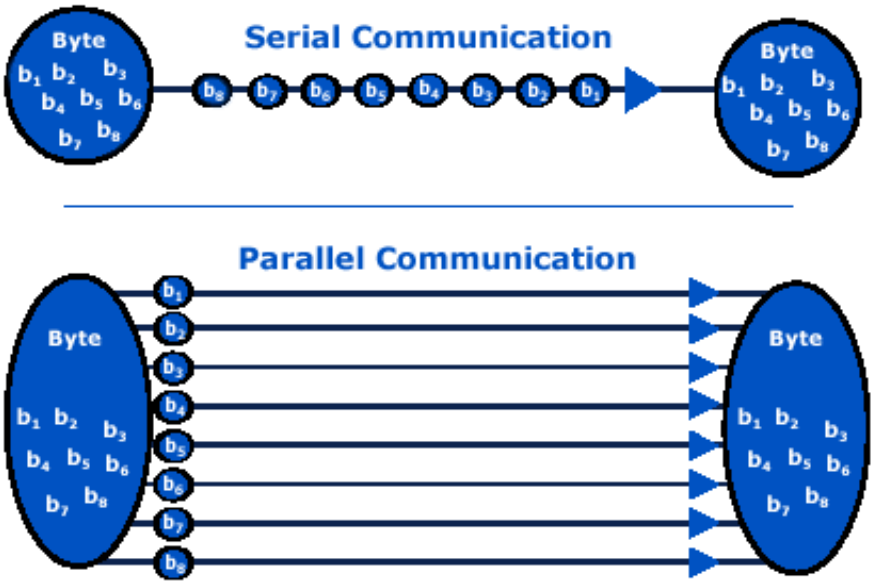


الشكل (3-82): نماذج لبعض دارات المرحل الممكن وصلها مع لوحة الأردوينو.

نقل المعلومات وفق بروتوكول الاتصال التسلسلي UART

3-9-1- مقدمة

يشار إلى عملية تبادل المعلومات ما بين الأجهزة بالاتصال communication. يمكن للطرفيات أن تتبادل المعلومات إما بشكل تسلسلي serial أو بشكل تفرعي parallel كما هو موضح في الشكل (3-83). في الاتصال التفرعي يتم إرسال مجموعة من النبضات الرقمية في نفس الوقت، بينما في الاتصال التسلسلي يتم إرسال نبضة رقمية كل فترة زمنية، بالتالي يتم إرسال النبضات على التتالي. على الرغم من أن الاتصال التفرعي قد يحقق سرعة نقل مرتفعة، إلا أنه أكثر تكلفة، لذلك يستخدم في اتصالات المسافات القصيرة مثل مسارات أنظمة الحواسيب وبعض الدارات المتكاملة. في المقابل يتطلب الاتصال التسلسلي عدد خطوط نقل أقل من الاتصال التفرعي، وبالتالي تكلفة أقل، كما أنه في الوقت الحاضر تم التوصل إلى معدلات نقل عالية للاتصال التسلسلي كما هو الحال في USB واتصالات الليف الضوئية، وبعض الدارات المتكاملة الحديثة.



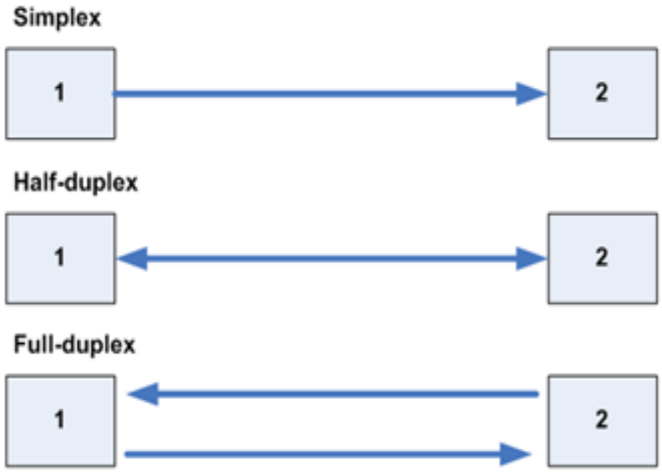
الشكل (3-83): الاتصال بشكل تسلسلي، وبشكل تفرعي.

يمكن تصنيف الاتصالات التسلسلية كما هو موضح في الشكل (3-84) حسب اتجاه إرسال المعطيات إلى :

1- **simplex** : يتم إرسال المعطيات باتجاه واحد فقط من المرسل إلى المستقبل.

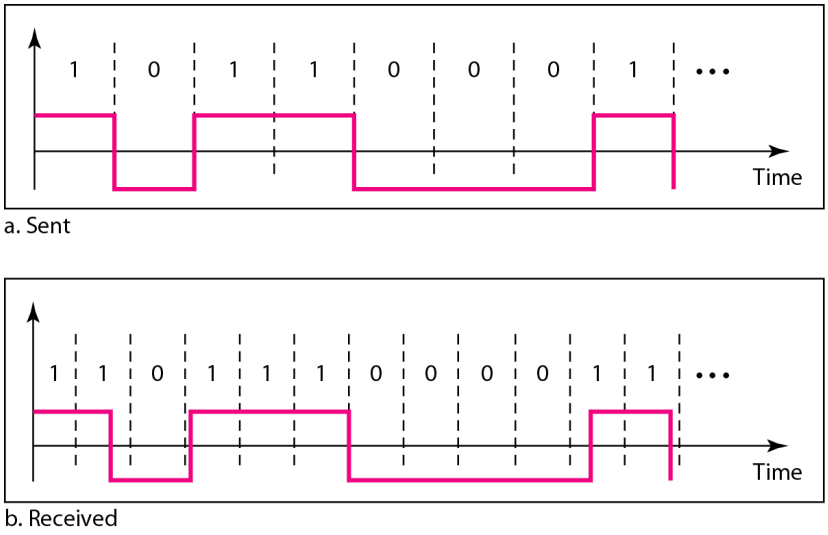
2- **half – duplex** : يتم إرسال المعطيات باتجاهين ولكن لا يمكن نقلها بالاتجاهين في نفس الوقت.

3- **full – duplex** : يتم إرسال المعطيات باتجاهين وبنفس الوقت.



الشكل (3-84): تصنيف الاتصالات التسلسلية حسب اتجاه نقل المعطيات.

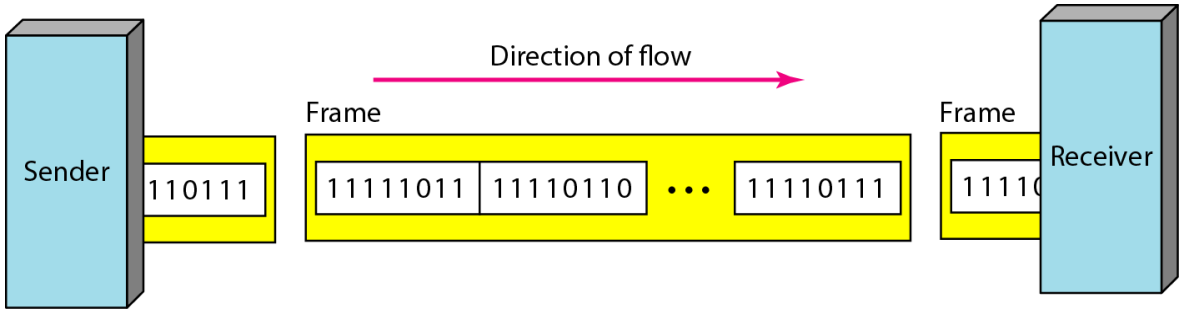
لابد من وجود إيقاع معين ما بين جهازي الإرسال والاستقبال، بمعنى أنه يجب على جهاز الإرسال أن يكون لديه المقدرة على تحديد بداية ونهاية كل بت ومعدل الإرسال المستخدم وهو ما يعرف بتزامن الإرسال. يوضح الشكل (3-85) تأثير عدم وجود تزامن ما بين المرسل والمستقبل.



الشكل (3-85): تأثير عدم وجود تزامن ما بين المرسل والمستقبل.

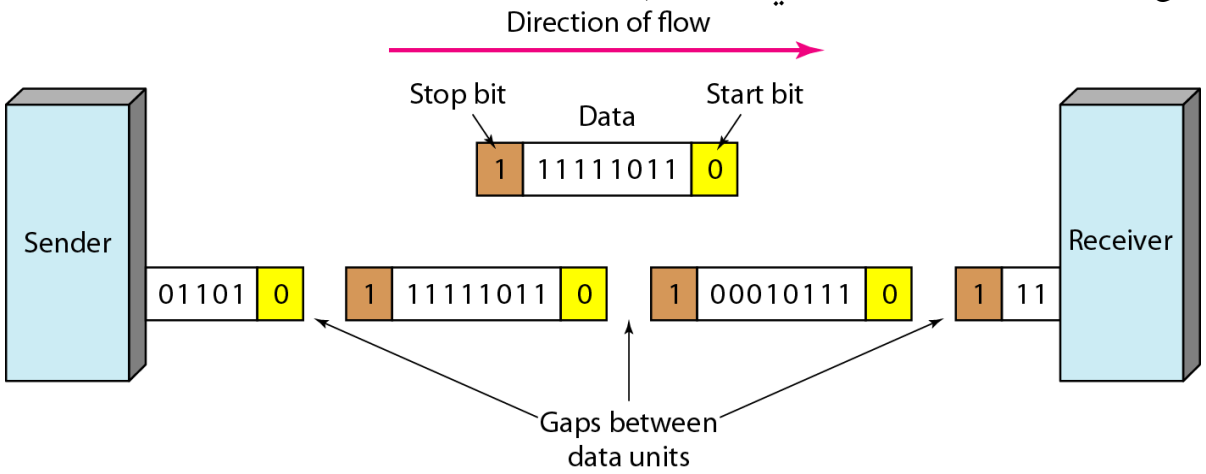
يوجد طريقتان مختلفتان لتزامن الإرسال:

1-الإرسال المتزامن synchronous transmission: يستخدم إشارة ساعة خارجية لإجراء عملية التزامن كما هو موضح في الشكل (3-86). قد ترسل هذه الإشارة على خط منفصل عن المعطيات وتعرف ب Clock أو Strobe، أو يمكن تضمينها مع إشارة المعطيات واستعادة إشارة clock عند المستقبل. من الأمثلة على البروتوكولات التي تستخدم إشارة ساعة clock على خط منفصل: I2C , SPI. من الأمثلة على تضمين إشارة المعطيات بإشارة ساعة استخدام ترميز Manchester. يستخدم الإرسال المتزامن لنقل كتل كبيرة من المعطيات، ويتم إرسالها على شكل إطارات frames.



الشكل (3-86): إرسال المعطيات وفق الأسلوب المتزامن.

2- الإرسال غير المتزامن asynchronous transmission: تستخدم في هذه الطريقة إشارات خاصة على طول وسط النقل. يتم في البداية إرسال بت بداية ومن ثم بتات المعطيات (نموذجياً 8 أو 7 بتات) ومن ثم بت توقف كما موضح في الشكل (3-87). يوجد ثغرات Gaps ما بين إرسال البايتات المرسلة توفر للمستقبل فترة من أجل التزامن. يعتبر هذا النمط من الإرسال مناسباً لإرسال معطيات بمعدلات بت منخفضة. ويستخدم عموماً مع معطيات مرسلة على فترات غير منتظمة (كما في لوحة المفاتيح). من أمثلة البروتوكولات التي تستخدم الإرسال غير المتزامن: RS232, RS485.



الشكل (3-87): إرسال المعطيات وفق الأسلوب غير المتزامن.

أشهر بروتوكولات الاتصال التسلسلية المستخدمة في المتحكمات الصغرية: RS232, RS422, RS485, I2C, SPI, CAN, Ethernet.

3-9-2- بروتوكول الاتصال التسلسلي RS-232 (recommended standard)

يعد بروتوكول RS-232 من أشهر بروتوكولات الاتصال التسلسلية المستخدمة لنقل المعلومات ما بين طرفيتين، مثل ربط الحاسب مع أجهزة محيطية أخرى كالمودم والطابعة والفارة. تعرف الطرفية الأولى DTE (data terminal equipment) ، والطرفية الثانية DCE (data communication equipment). فيما يلي عرض لخصائص هذا البروتوكول:

1-يتم نقل المعطيات على خطين : خط يتم فيه إرسال المعطيات من الجهاز إلى الجهاز الآخر و خط يتم فيه إرسال المعطيات في الاتجاه الآخر، أي أن الاتصال بازدواجية تامة full duplex. يمكن أن ترتبط الطرفيتان أيضاً بخطوط تحكم أخرى (نمط المصافحة handshake). لا يوجد خط تزامن.

2-المستويات المنطقية على خطوط المعطيات: يحدد للواحد منطقي (يعرف أيضاً بالعلامة mark) مستوي الجهد الأكثر سلبية من 3V-، ويحدد للصفر منطقي (يعرف أيضاً بالفراغ space) مستوي الجهد الأكثر إيجابية من 3V+. قد يكون هذا غريب ولكنه يزيد من المسافة و وثوقية نقل المعطيات.
مجال الجهد من 3V- إلى 3V+ غير مسموح به.

3-ترسل المعطيات على شكل مجموعة من البتات وفق الترتيب التالي:

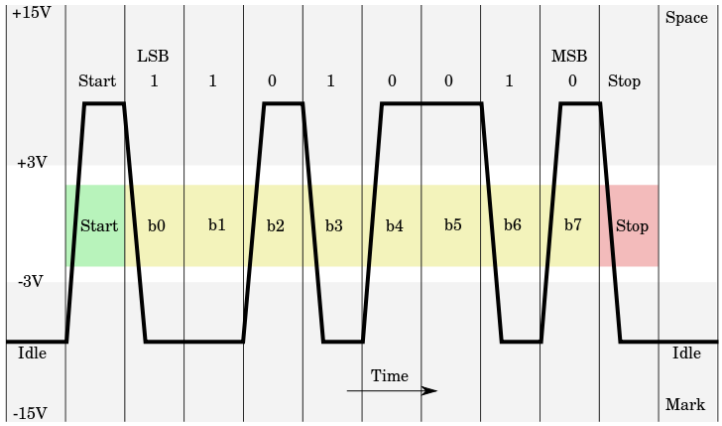
بت البداية start bit: بت يدل على بداية عملية إرسال المعطيات ويكون على حالة الجهد المرتفع (logic 0).

بتات الرمز أو الحرف Character Bits: عدد هذه البتات 5 أو 6 أو 7 أو 8. البت الأقل أهمية هو البت المرسل أولاً.

بت الفحص Parity Bit: بت اختياري يمكن عدم إرساله. ويستخدم لكشف الأخطاء الممكن حدوثها على بتات الحروف وهناك نموذجان لهذا البت : فحص زوجي Even Parity: يأخذ بت الفحص صفر منطقي أو واحد منطقي بحيث يكون مجموع بتات الحروف المساوية للواحد المنطقي مع بت الفحص مساوية لعدد زوجي. فحص فردي Odd Parity: يأخذ بت الفحص صفر منطقي أو واحد منطقي بحيث يكون مجموع بتات الحروف المساوية للواحد المنطقي مع بت الفحص مساوية لعدد فردي

بت توقف Stop Bits : بت توقف واحد أو أكثر يتم إضافته في نهاية بتات الحروف أو بت الفحص. ويكون على حالة الجهد المنخفض (logic 1) . بت التوقف يعطي للمستقبل فترة زمنية كافية ليكون مستعداً لاستقبال حرف أخر.

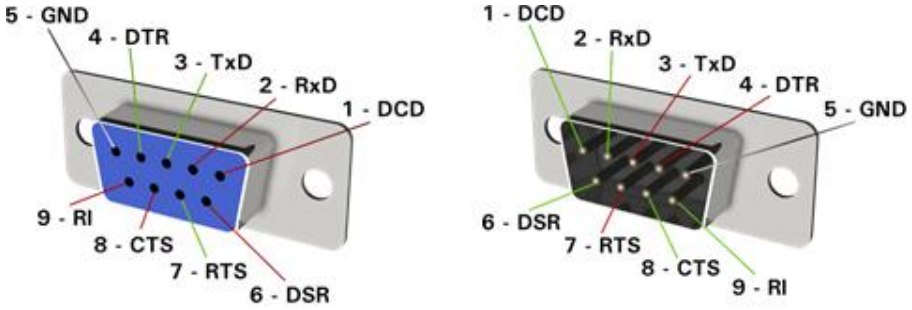
يوضح الشكل (3-88) كيفية إرسال الحرف "K" وفق بروتوكول الاتصال RS232.



الشكل (3-88): إرسال الحرف "K" وفق بروتوكول الاتصال RS232.

4-لابد من ضبط كل من المرسل والمستقبل على نفس معدل أو سرعة الإرسال (عدد البتات في الثانية) لكي تتم عملية التزامن بشكل صحيح. يعرف معدل الإرسال بـ baud rate و أكثر المعدلات شهرة و يعمل بها : Baud rate: 50 , 110 , 300 , 1200 , 2400 , 4800 , 9600 , 14400, 28800, 33600, 56000, 115000, rarely (330000)

5-وصلات RS-232 : يتطلب المعيار RS-232 لإجراء عمليات الاتصال ما بين الطرفيات وصلات خاصة إما 9 pins أو 25 pins. الوصلة 25 pins يطلق عليها DB-25 ، أما الوصلة 9 pins يطلق عليها DB-9. إضافة لذلك هناك الوصلة RJ-45 المستخدمة في الإثرنت. يبين الشكل (3-89) الوصلة DB-9 ، ووظيفة كل قطب pin.



الشكل (3-89): وصلة DB-9 ، ووظيفة كل قطب pin.

يمكن تصنيف الإشارات المرسلة على الأرجل السابقة إلى :

1- إشارات معطيات data signal وتضم كلاً من :

received data(RxD) – transmitted data(TxD)

2- إشارات تحكم control signal وتضم كلاً من :

clear to send(CTS) – request to send(RTS) - ring indicator(RI) - data terminal ready(DTR) – data set ready(DSR) - carrier detect(CD)

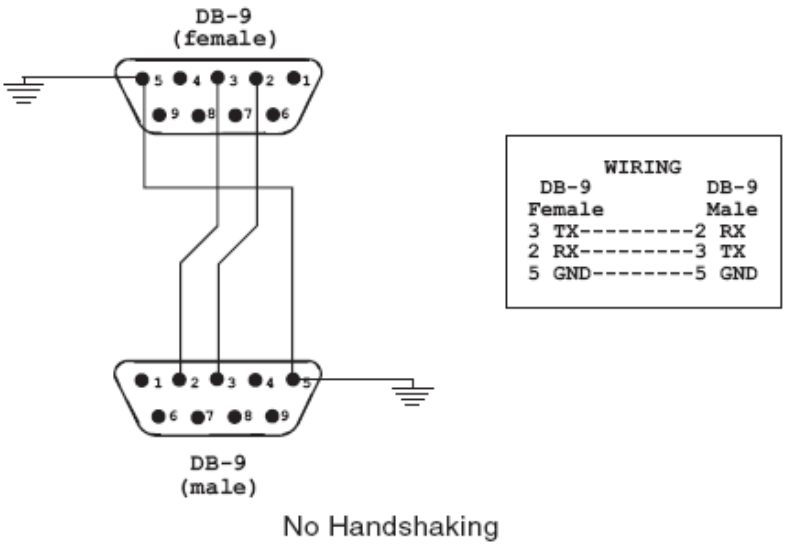
يمكن العمل تبعاً لذلك بإحدى الشكلين التاليين:

1- بدون مصافحة no handshaking : باستخدام إشارات المعطيات فقط.

2- مصافحة full handshaking : باستخدام إشارات المعطيات والتحكم.

سنهتم هنا بالنمط الأول من دون مصافحة فهو أبسط بكثير.

يوضح الشكل (3-90) كيفية ربط جهازين (مثلاً حاسوبين) وفق المعيار RS232 ونمط عدم المصافحة.



الشكل (3-90): ربط حاسوبين معاً من خلال منافذهما التسلسلية DB-9 ووفق نمط بدون مصافحة.

3-9-3-نقل المعطيات ما بين لوحة الأردوينو والطرفيات الأخرى من خلال بروتوكول الاتصال UART

يسمح بروتوكول الاتصال universal asynchronous receiver-transmitter (UART) للوحة الأردوينو من أن تتصل مع طرفيات أخرى كلوحة أردوينو أخرى أو حاسوب وغير ذلك. خواص بروتوكول UART مشابهة لبروتوكول RS232 باستثناء مستوى جهود 1 و0 منطقي، حيث تأخذ 5V+ و0V، لذلك لا يستخدم UART مع المسافات الطويلة. لتبديل مستوى جهود المعطيات الرقمية من UART إلى RS232 والعكس تستخدم بعض الدارات المتكاملة مثل max232. تُدعم بعض المتحكمات الصغرية ببنية مادية للاتصال وفق هذا البروتوكول. إذا لم يتضمن المتحكم الصغري وحدة اتصال UART يمكن عندئذ بناؤها برمجياً باستخدام أرجل الدخل والخرج الرقمية.

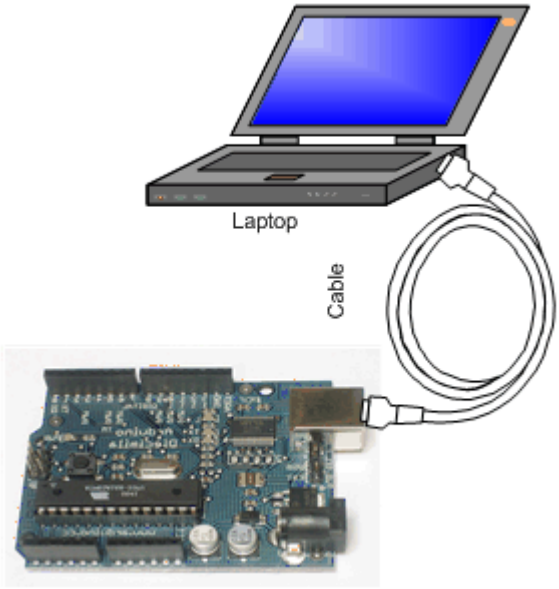
في لوحة الأردوينو أونو يتم الاتصال وفق بروتوكول UART ببنية مادية من خلال رجل إرسال TX (pin 1)، ورجل استقبال RX (pin 0).

لم يعد في الوقت الحاضر المنفذ التسلسلي الموضح في الشكل (3-89) منتشراً في أجهزة الحواسيب وتم استبداله بمنفذ USB. تبعاً لذلك نحتاج لربط لوحة الأردوينو مع منفذ USB للحاسب إلى استخدام دارة الكترونية تعمل على التبديل ما بين UART و USB. يوجد خياران لتنفيذ ذلك:

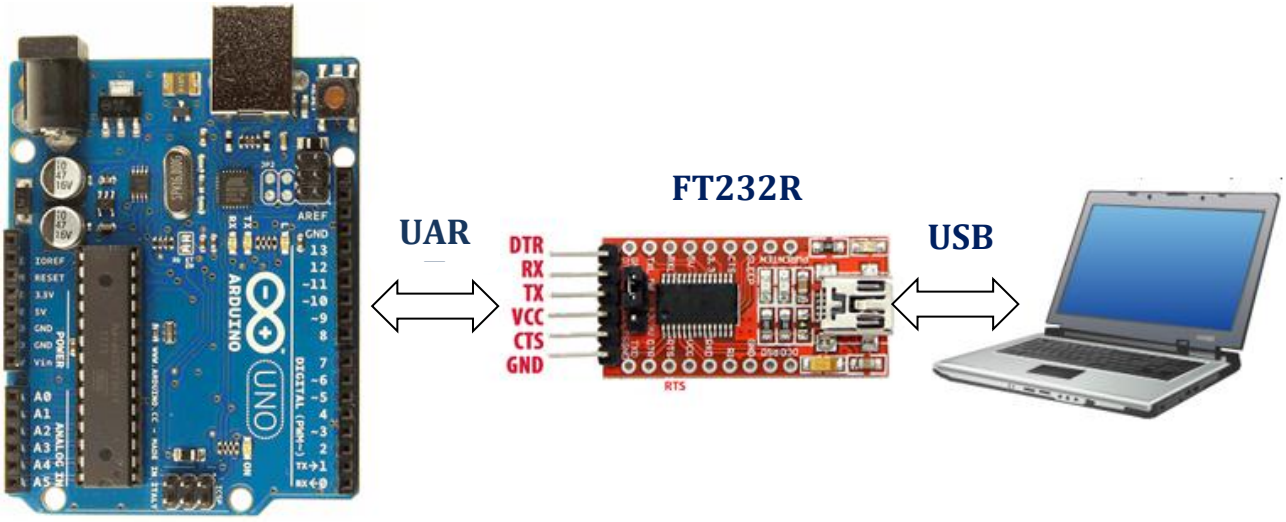
- من خلال المتحكم الصغري الثانوي ATmega16U2 المدمج ضمن لوحة الأردوينو أونو، حيث يعمل على تحويل المعطيات المرسلة من المتحكم الرئيسي وفق بروتوكول UART إلى بروتوكول USB والعكس، بالتالي ربط لوحة الأردوينو بشكل مباشر مع الحاسب كما هو موضح في الشكل (3-91).
- من خلال استخدام لوحة الكترونية خارجية تحتوي على دارة متكاملة تعرف بـ FT232R كما هو موضح في الشكل (3-92). تعمل FT232R كوسيط اتصال ما بين UART و USB بشكل مشابه للمتحكم الثانوي ATmega16U2. لهذه اللوحة رجل إرسال TX يتم وصلها مع رجل استقبال لوحة الأردوينو(0 RX (pin ، ورجل استقبال RX يتم وصلها مع رجل إرسال لوحة الأردوينو(1 TX (pin. يوجد أيضاً رجل تغذية Vcc تعطي جهداً 5V+، ورجل أرضي GND، يمكن استخدامهما لتغذية لوحة الأردوينو أو أية دارات أخرى. الرجل DTR و CTS لا داعي لوصلها في تطبيقاتنا.

عند وصل لوحة الأردينو أو لوحة FT232R مع منفذ USB للحاسب، يطلب نظام التشغيل برنامج تعريف driver. بالنسبة للوحة الأردوينو يكون التعريف موجود مع حزمة Arduino IDE التي تم شرحها في الفصل الثاني. أما شريحة FT232R يمكن تنصيب التعريف Virtual COM port (VCP) driver من موقع المنتج http://www.ftdichip.com . في كلا الحالتين ستظهر لوحة الأردوينو أو لوحة FT232R على الحاسب كمنفذ COM إضافي عند فتح نافذة إدارة الأجهزة. تبعاً لذلك فإن البرنامج المستخدم في الحاسب للتواصل مع لوحة الأردوينو (مثل نافذة المراقبة التسلسلية serial

monitor في برنامج Arduino IDE) سيتعامل مع لوحة الأردوينو أو لوحة FT232R على أنهما منفذ COM .



الشكل (3-91): نقل المعلومات ما بين لوحة الأردوينو والحاسب المحمول عبر منفذ USB عن طريق المتحكم الثانوي المدمج ضمن لوحة الأردوينو.



الشكل (3-92): نقل المعلومات ما بين لوحة الأردوينو والحاسب عبر منفذ USB عن طريق شريحة FT232R. يتم وصل رجل TX في لوحة FT232R مع رجل RX (pin 0) للوحة الأردوينو، والعكس.

للوحة الأردوينو ميغا Mega أربع منافذ تسلسلية:

- serial 0: رجل إرسال TX (pin 1)، ورجل استقبال RX(pin 0)،
- serial 1: رجل إرسال TX (pin 18)، ورجل استقبال RX (pin 19).
- serial 2: رجل إرسال TX (pin 16)، ورجل استقبال RX (pin 17).
- serial 3: رجل إرسال TX (pin 14)، ورجل استقبال RX (pin 15).

يبين الجدول التالي التعليمات المستخدمة في برنامج Arduino IDE للاتصال وفق البنية المادية لبروتوكول UART، أي في لوحة الأردوينو UNO مع رجل الاستقبال (RX) pin0، ورجل الإرسال (TX) pin1، ومنفذ USB في نفس الوقت عن طريق المتحكم الثانوي ATmega16U2

التعلمية	الوظيفة

while (!serial) { }
 	
if (serial)
	تعد serial شرطاً منطقياً، حيث تعيد القيمة true إذا كان المنفذ التسلسلي جاهز للاتصال، وتعيد false في حالة عدم جهوزيته. لذلك يمكن استخدامها بحيث تنتظر لوحة الأردوينو حتى يصبح المنفذ جاهزاً. مثال:
	while (!Serial) {
	// wait for serial port to connect. Needed for native USB
	}

Serial.available()	تعيد هذه التعليمة عدد البايتات (الحروف) المستقبلة والمخزنة في ذاكرة الاستقبال التسلسلية والجاهزة للقراءة. مثال:
	if (Serial.available() > 0) {
	// يتحقق الشرط في حال وجود بيانات مستقبلة
	}

Serial.begin(speed) Serial.begin(speed, config) من أجل لوحة *Arduino Mega* فقط : Serial1.begin(speed) Serial2.begin(speed) Serial3.begin(speed) Serial1.begin(speed, config) Serial2.begin(speed, config) Serial3.begin(speed, config)	تستخدم هذه التعليمة لتحديد معدل الإرسال من خلال البارامتر speed. يمكن أيضاً وبشكل اختياري تحديد عدد بتات المعطيات data، الزوجية parity، بتات التوقف stop من خلال البارامتر config. بشكل افتراضي عدد البتات 8، بدون بت فحص زوجية، بت توقف واحد. مثال: Serial.begin(9600);

Serial.print(val)
Serial.print(val, format)
	تسمح هذه التعليمة بإرسال معطيات على المنفذ التسلسلي على شكل نص ASCII من خلال البارامتر val. يأخذ البارامتر val أي نمط من المعطيات (أرقام أو حروف) وكلها ترسل على شكل ASCII. البارامتر format اختياري ومن خلاله يتم ضبط إرسال الأرقام كما في الأمثلة التالية:
	Serial.print(78) //gives "78"
	Serial.print(1.23456) //gives "1.23"
	Serial.print('N') //gives "N"
	Serial.print("Hello world.") //gives "Hello world."
	Serial.print(78, BIN) //gives "1001110"
	Serial.print(78, OCT) //gives "116"
	Serial.print(78, DEC) //gives "78"
	Serial.print(7, HEX) //gives "4E"
	Serial.println(1.23456, 0) // gives "1"
	Serial.println(1.23456, 2)) // gives "1.23"
	Serial.println(1.23456, 4)) // gives "1.2346"

التعلمية	الوظيفة
Serial.println(val) Serial.println(val, format)	نفس وظيفة التعليمة السابقة، ولكن يتبع النص التسلسلي برمز عودة النقل ('\r' or ASCII 13)، ورمز السطر الجديد ('\n' or ASCII 10)

Serial.write(val)
Serial.write(str)
	يتم إرسال معطيات ثنائية. المعطيات قد ترسل كبايت واحد (البارامتر val) أو سلسلة من البايتات (البارامتر str). مثال:
	Serial.write(82); // send 82 (1010010)
	Serial.write("hello"); //send the string "hello"

Serial.read()
Arduino Mega only:
Serial1.read()
Serial2.read()
Serial3.read()
	يتم قراءة البايت الأول للمعطيات التسلسلية القادمة والمتاحة. في حالة عدم وجود معطيات تعيد التعليمة 1-. مثال:
	int incomingByte = 0;
	.....
	incomingByte = Serial.read();

Serial.readString()	تعمل على قراءة سلسلة الرموز الموجودة من الذاكرة التسلسلية. تنتهي هذه الوظيفية في حالة انتهاء المهلة (تحددها التعليمة setTimeout ومدة المهلة افتراضياً ثانية).
Serial.parseInt()	تعمل على قراءة أول رقم صحيح (موجب أو سالب) من الذاكرة التسلسلية serial buffer. تنتهي هذه الوظيفية في حالة انتهاء المهلة أو قراءة قيمة ليست عدد صحيح، وتعيد قيمة 0 عندئذ.
Serial.parseFloat()	تعمل على قراءة أول رقم حقيقي (نقطة عائمة floating point) من الذاكرة التسلسلية serial buffer. تنتهي هذه الوظيفية في حالة انتهاء المهلة أو قراءة قيمة ليست عدد صحيح، وتعيد قيمة 0 عندئذ.
Serial.setTimeout(*time*)	يتم تحديد مهلة الانتظار العظمى عند استخدام بعض التعليمات التسلسلية من خلال البارامتر time وبواحدة الميلي ثانية ms.

يمكن أيضاً بناء بروتوكول UART برمجياً وعندئذ يمكن استخدام أي أرجل رقمية للوحة الأردوينو. من الممكن بناء عدة أرجل لتعمل وفق بروتوكول UART وبسرعة تصل إلى 115200 bps. لتفعيل البناء البرمجي بروتوكول UART برمجياً نتبع الخطوات التالية:

1-نستدعي مكتبة SoftwareSerial.h في البداية.

```
#include < SoftwareSerial.h>
```

2-بعد استدعاء المكتبة مباشرةً ننشئ كائن تسلسلي جديد (نسميه مثلاً mySerial)، ونحدد رجل الاستقبال ثم رجل الإرسال كما يلي:

```
SoftwareSerial mySerial(rxPin,txPin);
```

مثال:

```
SoftwareSerial mySerial(2,3);
```

3-داخل الإجرائية void setup() ندخل معدل نقل البتات كما يلي:

```
mySerial.begin(speed);
```

مثال:

```
mySerial.begin(9600);
```

4-يمكن الآن استخدام التعليمات:

mySerial.read(), mySerial.print(), mySerial.write(), mySerial.available(),

والتي تم شرحها سابقاً في الجدول.

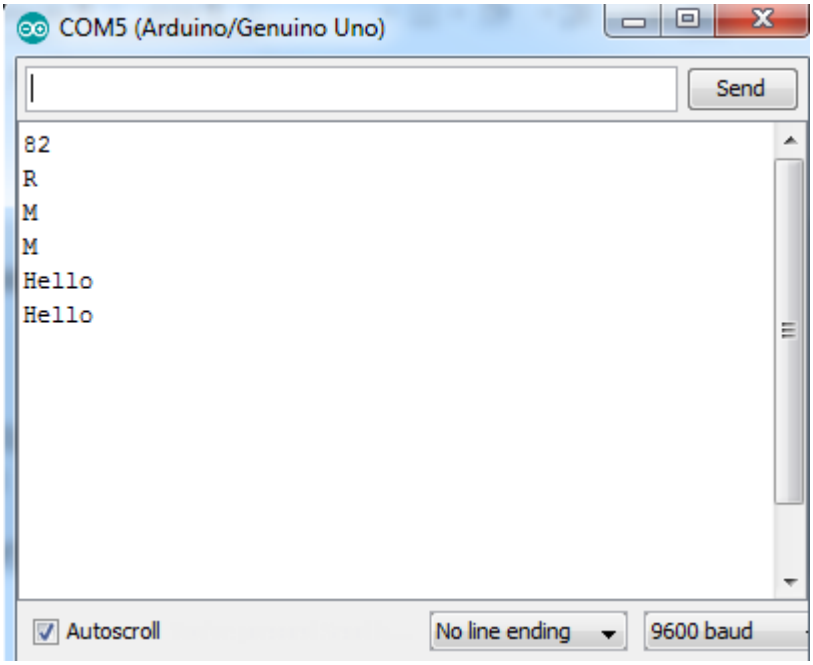
3-9-4-الكود البرمجي

التطبيق الأول

يهدف هذا التطبيق إلى معرفة الفرق ما بين تعليمتي الإرسال Serial.println ، Serial.write.

```
void setup() {
  Serial.begin(9600); // opens serial port, sets data rate to 9600 bps
}

void loop() {
  Serial.println(82);
  Serial.write(82);
  Serial.println();
  Serial.println('M');
  Serial.write('M');
  Serial.println();
  Serial.println("Hello");
  Serial.write("Hello");
  while (1) { }
}
```

بعد تحميل البرنامج وفتح نافذة المراقبة التسلسلية serial monitor في برنامج Arduino IDE، تم التوصل إلى النتيجة الموضحة في الشكل (3-93). نلاحظ أن التعليمتين تقدمان نفس النتيجة عند إرسال حرف 'M' وسلسلة حروف "Hello"، ولكن يختلفان عند إرسال رقم 82، التعليمة Serial.println (أو Serial.print) ترسل الرقم على شكل حرفين أو رمزين '2' '8' ، بينما التعليمة Serial.write فترسل الحرف المكافئ للرقم 82 وفق ترميزات أسكي وهو حرف 'R'.



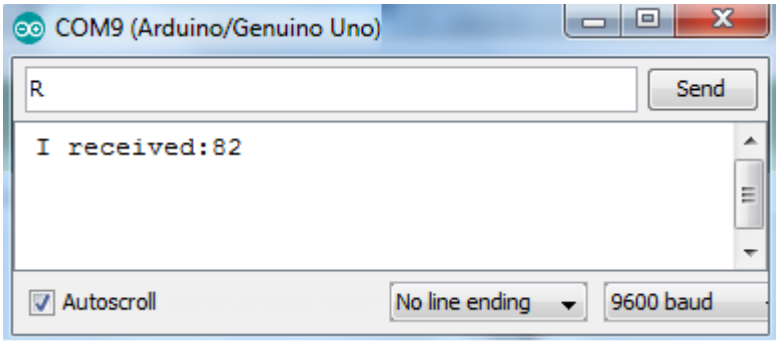
الشكل (3-93): نافذة المراقبة التسلسلية للتطبيق الأول.

التطبيق الثاني

يهدف هذا التطبيق إلى معرفة كيفية استخدام تعليمة قراءة بايت واحد وهي Serial.read().

```
int incomingByte = 0;

void setup() {
  Serial.begin(9600); // opens serial port, sets data rate to 9600 bps
}

void loop() {
  if (Serial.available() > 0) {
    incomingByte = Serial.read(); // read the incoming byte:
    Serial.print(" I received:");
    Serial.println(incomingByte);
  }
}
```

بعد تحميل البرنامج وفتح نافذة المراقبة التسلسلية serial monitor في برنامج Arduino IDE، تم التوصل إلى النتيجة الموضحة في الشكل (3-94). عند إرسال حرف 'R' من الحاسب تكون لوحة الأردوينو قد استقبلت ترميزة أسكي المكافئة له، لتخزنها في المتغير incomingByte. بعد ذلك ترسل لوحة الأردوينو للحاسب رمزين '2' '8' لأنه تم استخدام تعليمة الإرسال Serial.println كما تم شرحه في التطبيق الأول.



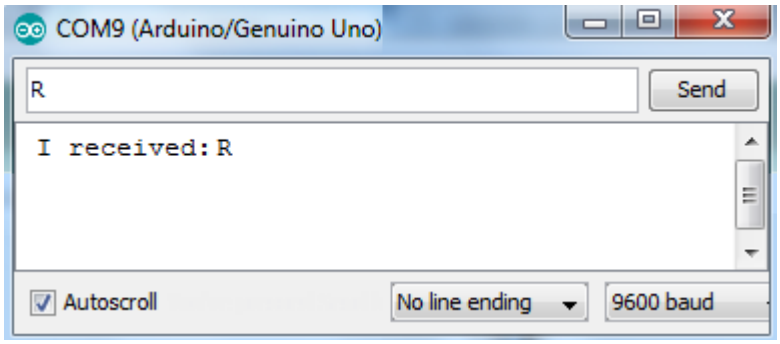
الشكل (3-94): نافذة المراقبة التسلسلية للتطبيق الثاني.

التطبيق الثالث

سنعيد نفس التطبيق السابق مع استبدال Serial.println بالتعليمة Serial.write.

```
int incomingByte = 0;

void setup() {
  Serial.begin(9600); // opens serial port, sets data rate to 9600 bps
}

void loop() {
  if (Serial.available() > 0) {
    incomingByte = Serial.read(); // read the incoming byte:
    Serial.print(" I received:");
    Serial.write(incomingByte);
  }
}
```

بعد تحميل البرنامج وفتح نافذة المراقبة التسلسلية serial monitor في برنامج Arduino IDE، تم التوصل إلى النتيجة الموضحة في الشكل (3-95). عند إرسال حرف 'R' من الحاسب تكون لوحة الأردوينو استقبلت ترميزة أسكي المكافئة له، لتخزنها في المتغير incomingByte. بعد ذلك ترسل لوحة الأردوينو للحاسب الحرف 'R' لأنه تم استخدام تعليمة الإرسال Serial.write كما تم شرحه في التطبيق الأول.



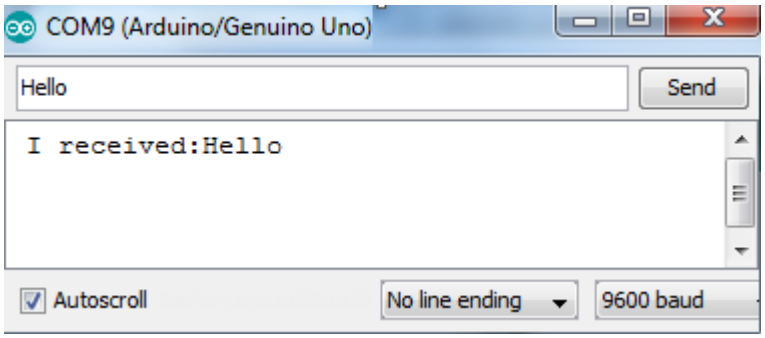
الشكل (3-95): نافذة المراقبة التسلسلية للتطبيق الثالث.

التطبيق الرابع

يهدف هذا التطبيق معرفة كيفية استخدام تعليمة استقبال سلسلة من المحارف وهي Serial.readString(). تم تخزين السلسلة المستقبلة في متغير اسمه a تم التصريح عنه على شكل String.

```
String a;
void setup() {
  Serial.begin(9600); // opens serial port, sets data rate to 9600 bps
}

void loop() {
  if (Serial.available() > 0) {
    a = Serial.readString(); // read the incoming data as string
    Serial.print(" I received:");
    Serial.println(a);
  }
}
```

بعد تحميل البرنامج وفتح نافذة المراقبة التسلسلية serial monitor في برنامج Arduino IDE، تم التوصل إلى النتيجة الموضحة في الشكل (3-96). عند إرسال سلسلة من المحارف"Hello " من الحاسب إلى لوحة الأردوينو، يتم تخزين السلسلة في المتغير a، ومن ثم يعاد إرسالها من جديد باستخدام التعليمة Serial.println. إذا استخدمت التعليمة Serial.write سنصل إلى نفس النتيجة.



الشكل (3-96): نافذة المراقبة التسلسلية للتطبيق الرابع.

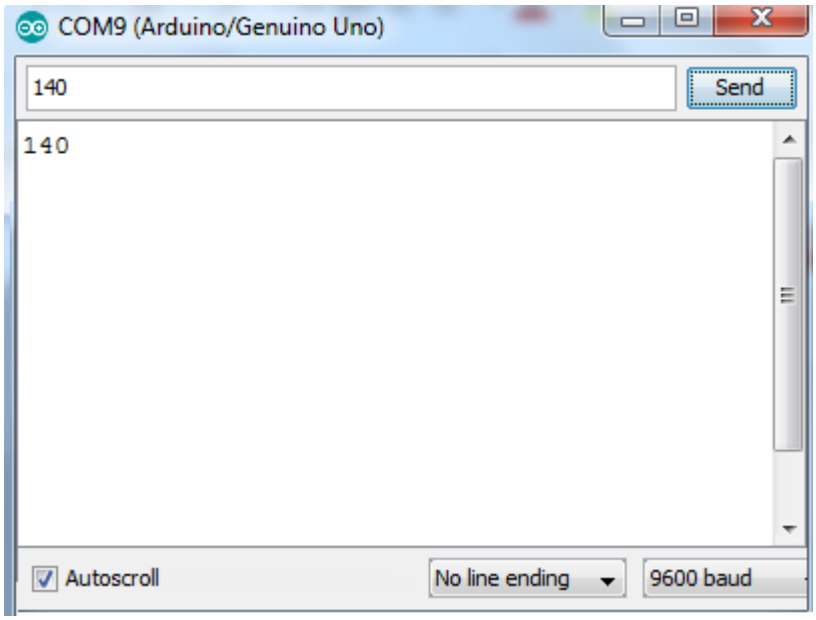
التطبيق الخامس

يهدف هذا التطبيق معرفة كيفية استخدام تعليمة استقبال رقم صحيح وهي Serial.parseInt(). تم تخزين هذا الرقم في متغير اسمه a تم التصريح عنه على شكل int.

```
int a;

void setup()
{
  Serial.begin(9600);
}
void loop()
{
```

```
 //waiting for input
 if (Serial.available() > 0) {
 a = Serial.parseInt(); //read int or parseFloat for ..float...
 Serial.println(a);}
}
```

بعد تحميل البرنامج وفتح نافذة المراقبة التسلسلية serial monitor في برنامج Arduino IDE، تم التوصل إلى النتيجة الموضحة في الشكل (3-97). عند إرسال رقم قيمته 140 من الحاسب إلى لوحة الأردوينو، فإنها تخزن هذا الرقم في المتغير a الذي تم تعريفه على أنه عدد صحيح. تم إعادة إرسال هذا الرقم للحاسب باستخدام تعليمة Serial.println.



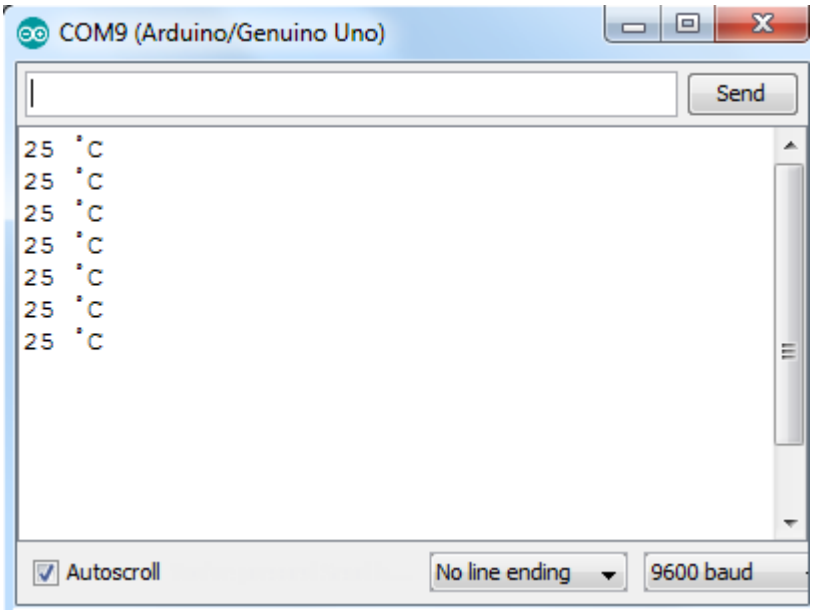
الشكل (3-97): نافذة المراقبة التسلسلية للتطبيق الخامس.

التطبيق السادس

تم في هذا التطبيق ربط حساس درجة الحرارة LM35 مع لوحة الأردوينو كما تم دراسته في الفقرة (3-6-2)، والمطلوب إرسال قيمة درجة الحرارة إلى الحاسب.

```
void setup()
{
Serial.begin(9600);
}

void loop() {
unsigned int sensorValue;
sensorValue=analogRead(0)/2;
Serial.print(sensorValue);
Serial.println("° C");
delay(1000);
}
```

بعد تحميل البرنامج وفتح نافذة المراقبة التسلسلية serial monitor في برنامج Arduino IDE، تم التوصل إلى النتيجة الموضحة في الشكل (3-98). نلاحظ أنه تم إرسال قيمة درجة الحرارة بشكل صحيح على شكل رمزين ('5' , '2') إلى الحاسب من خلال تعليمة Serial.print.



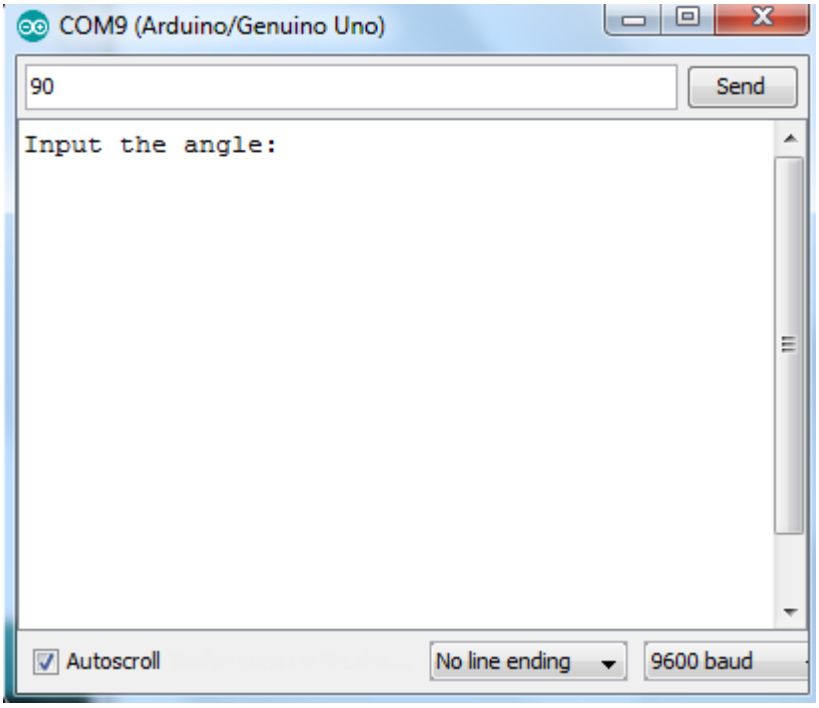
الشكل (3-98): نافذة المراقبة التسلسلية للتطبيق السادس.

التطبيق السابع

تم في هذا التطبيق ربط محرك سيرفو مع لوحة الأردوينو كما تم دراسته في الفقرة (3-7-4)، والمطلوب إرسال قيمة زاوية تحريك المحرك من الحاسب. يتم إرسال قيمة الزاوية من الحاسب، ليتم استقبالها كرقم من قبل لوحة الأردوينو باستخدام تعليمة Serial.parseInt() . من خلال تعليمة servo.write يتم تحريك المحرك إلى الزاوية المطلوبة.

```
#include <Servo.h>
Servo servo;
int x;

void setup()
{
 servo.attach(9);
 Serial.begin(9600);
 Serial.println("Input the angle:");
}

void loop() {
 //waiting for input
 if (Serial.available() > 0) {
   x = Serial.parseInt(); //read int
   servo.write(x);      // moving servo motor
 }
}
```

بعد تحميل البرنامج وفتح نافذة المراقبة التسلسلية serial monitor في برنامج Arduino IDE، تم التوصل إلى النتيجة الموضحة في الشكل (3-99).



الشكل (3-99): نافذة المراقبة التسلسلية للتطبيق السابع.

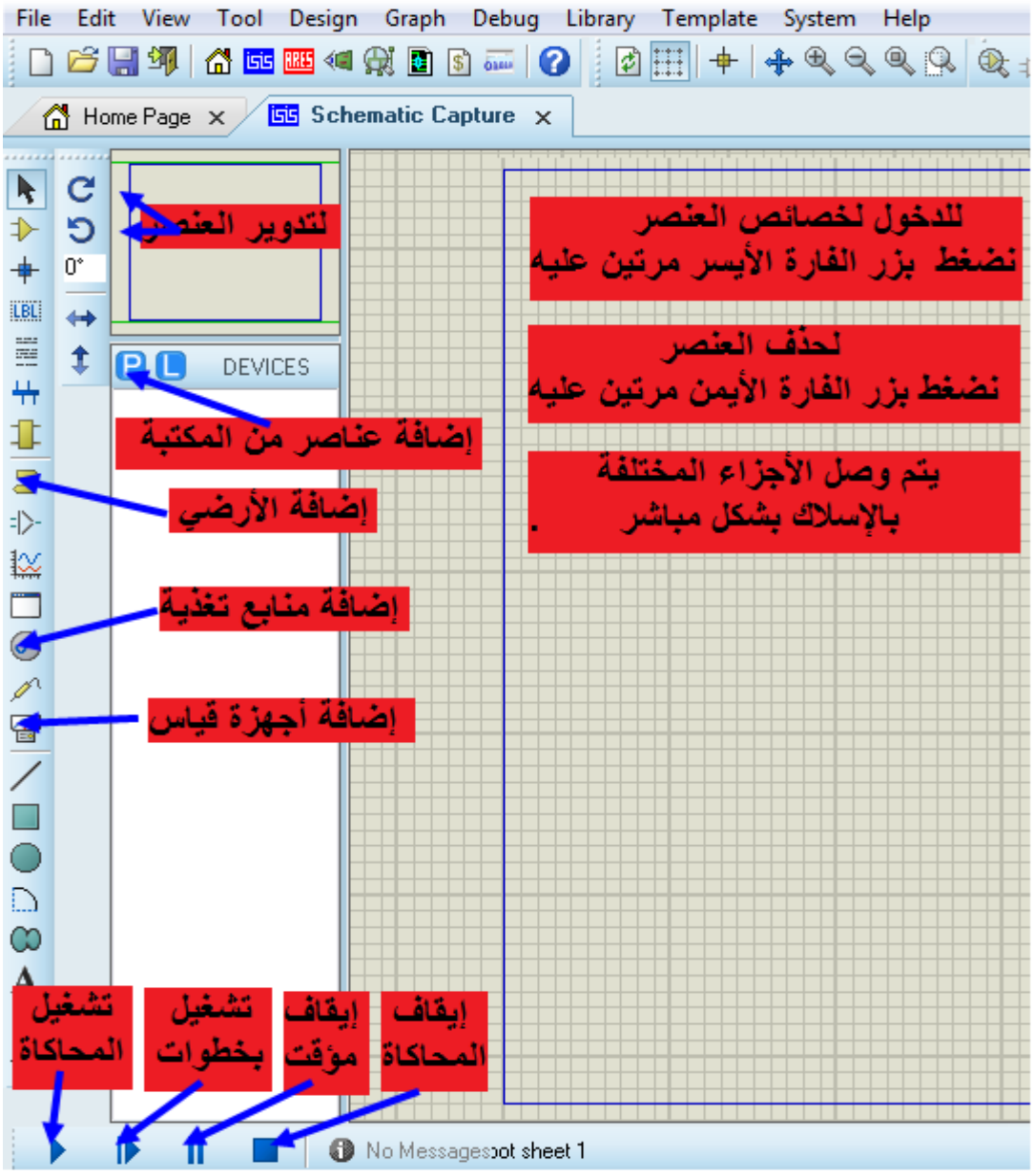
الملحق (1)

برامج محاكاة الأردوينو

بعد أن تتم برمجة لوحة الأردوينو وتنفيذ الدارة العملية، من الممكن أن لا تعمل الدارة. هذا يعود لسببين إما خطأ في كتابة الكود البرمجي أو في الوصل الالكتروني. يعتبر تحديد المشكلة أمراً هاماً، من هنا تأتي أهمية برامج المحاكاة التي من خلالها يتم اختبار صحة الكود ، والتأكد من خلوه من الأخطاء. يوجد العديد من برامج محاكاة المتحكمات، سنتكلم عن أشهرها : Proteus و VBB.

1-برنامج Proteus

يعد برنامج Proteus أشهر أداة برمجية لمحاكاة الدوائر الالكترونية بمختلف أنواعها التشابهية والرقمية بما فيها المتحكمات الدقيقة. يمتاز البرنامج بسهولة التعامل معه، وسنكتفي في الشكل بشرح مبسط لعناصر الواجهة الرئيسية له.



لتحميل نسخة البرنامج Proteus 8.6 :

http://download894.mediafire.com/vewihot42d2g/4m04g555lln13n6/proteus+8.6+SP2.rar

يتطلب البرنامج التشغيل عن طريق administrator.

لا تحتوي مكتبة عناصر البرنامج على الأردوينو لذلك يتم إضافته على الشكل التالي:

1-في البداية قم بتحميل مكتبة لوحات الأردوينو من الموقع التالي:

2-بعد التحميل قم بفك ضغط الملف وانقل مكتبات لوحات الأردوينو إلى داخل مكتبة برنامج Proteus الموجودة في الموقع (أو في موقع تنصيب البرنامج في جهازك)

C:\Program Files\Labcenter Electronics\Proteus 8 Professional/ LIBRARY

3-بعد فتح برنامج البروتوث، وفتح نافذة إضافة العناصر، نكتب Arduino لتظهر لوحات الأردوينو المضافة.

الآن سنتكلم عن خطوات محاكاة الدارة

1-بعد كتابة كود البرمجة في برنامج Arduino cc، نختار من قائمة الأدوات Sketch ومن ثم نضغط على Export Compiled Binary ليتولد ملف توسعه hex ضمن المجلد الذي تم فيه حفظ السكتش.
(ملف hex هو الملف الذي يتم نقله من الحاسب إلى المتحكم في اللوحة، وهو عبارة عن كود بلغة الآلة مكتوب بشكل ست عشري hexadecimal).

2-بعد إعداد دارة الأردوينو على البروتوث، اضغط مرتين على لوحة الأردوينو، واختر program file واختر الملف hex الذي تم توليده من الخطوة السابقة.

3-اضغط على زر التشغيل وتأكد من صحة كود البرمجة.

2-برنامج VirtualBreadboard

من أشهر برامج المحاكاة التي تستخدم ألواح التجارب BreadBorads في نمذجة الدارت بشكل يحاكي الواقع. يتوفر البرنامج على صورة اصدارين الأولى هي VBB express، والثانية VBB full version. الاختلاف بينهما أن الأولى مجانية لكن مع قدرات محاكاة محدودة أما الثانية نحصل على كامل الإمكانيات البرمجية و المحاكاة.

يمكن تحميل الأكواد البرمجية والمحاكاة للأمثلة مع مكتبات المستوى الأول من الموقع:

https://www.mediafire.com/file/s7cr0ynegwjm8da/examples.rar/file

يتناول كتاب الأردوينو من البداية وحتى الاحتراف – مستوى المتوسط – ما يلي:

- قراءة إشارة حساسات مختلفة (حساس حرارة ورطوبة DHT11، حساس مسافة HC-SR04، حساس غاز MQ5، حساس أول أكسيد الكربون MQ7، حساس حركة، حساس حريق، حساس التسارع وتحديد الاتجاه Accelerometer and Gyroscope).
- التحكم بشاشة ملونة ولمس.
- إرسال واستقبال البيانات (أوامر تحكم ومراقبة) من خلال تقنية البلوتوث.
- إرسال واستقبال البيانات (أوامر تحكم ومراقبة) من خلال شبكة Wi-Fi وشبكة الانترنت.
- إرسال واستقبال البيانات من خلال شبكة محلية LAN وشبكة الانترنت.
- ربط الأردوينو مع وحدة تحديد الموقع العالمية GPS.
- إرسال واستقبال البيانات من خلال الشبكة الخلوية.
- التقاط الصور ومعالجتها باستخدام الكاميرا OV7670 640 x 480 VGA CMOS.
- إرسال واستقبال البيانات من خلال بروتوكول I2C وتطبيق ساعة زمن حقيقي.
- تصميم واجهات ربط تفاعلية على الحاسب باستخدام برنامج فيجول بيسك.
- تصميم واجهات ربط تفاعلية على الأجهزة المحمولة التي تعمل وفق نظام الأندرويد.