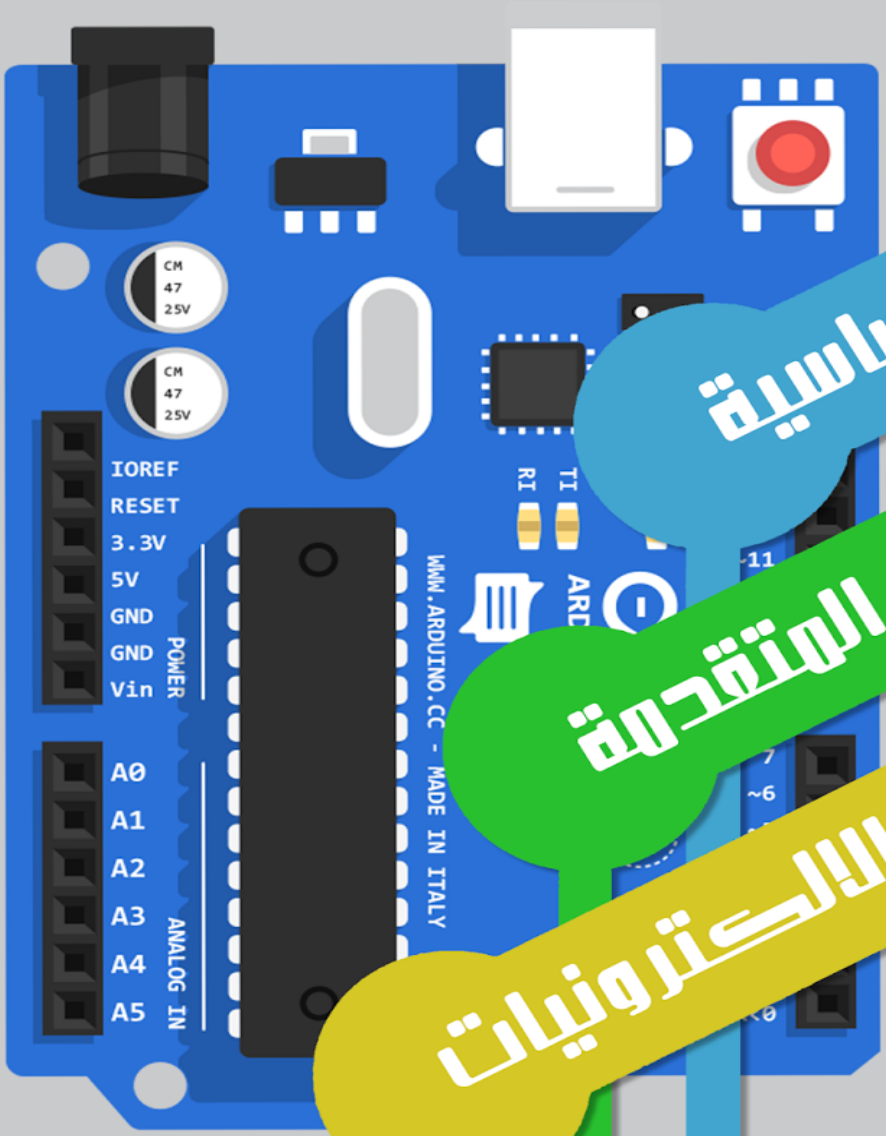
ARDUINO

برمجة الأردوينو



الأوامر الأساسية

المكتبات المتقدمة

وبعض الالكترونيات

jeem2.com
م.سامي قرامي

سوف نبدأ بوضع الأوامر الأساسية والشائعة في برمجة الأردوينو مع شرح مختصر في جداول لتكون مرجعا لك أثناء البرمجة. كتاب برمجة الأردوينو الفعلي يبدأ عند الصفحة (9)

أساسيات كود الأردوينو Sketch

`#include <EEPROM.h>` `int x =0;`	المنطقة بالأعلى (فوق void setup) يتم فيها تعريف المتغيرات و ادراج المكتبات عادة لاحظ أن مكان تعريف المتغير تؤثر على طريقة عمله
كل الأوامر داخل الأقواس {...} سيتم تنفيذها مرة واحدة في بداية تشغيل الكود	`void setup() { ... }`
الأوامر الموجودة بين القوسين { ... } سيتم تنفيذها بشكل مكرر مادام الأردوينو يعمل.	`void loop () { ... }`
`void fun1(){ ... }` `int fun2(int x) { ... }`	المنطقة أسفل دالة void loop تستخدم لكتابة دوال جديدة عادة
لكتابة ملاحظات من سطر واحد // لكتابة ملاحظات من عدة أسطر */ ..... /*	`// ....` `/* ..... */`

تحديد عمل المنافذ الرقمية 0-13(دخل / خرج)

و تكتب عادة في جزء الـ سيتاب void setup( ){ ... }

`pinMode(13,OUTPUT);`	تهيئة الطرف رقم 13 ليكون مخرج لاحظ أن الأطراف 3,5,6,9,10,11 تقبل الخرج التماثلي
`pinMode(12,INPUT);`	تهيئة الطرف 12 ليكون مدخل (لقراءة الجهد)
`pinMode(11,INPUT_PULLUP);`	تهيئة الطرف ليكون دخل مع ربطه بمقاومة رفع داخلية إلى 5v _ عادة لايستخدم مع الطرف 13

أوامر الإدخال (القراءة) أو الإخراج (الكتابة) input & outputs

`digitalWrite(13,HIGH);`	يخرج 5v على الطرف 13 ، لإخراج 0v اكتب LOW يمكن كتابة 1 بدل HIGH أو 0 بدل LOW
`digitalRead(13);`	يقرأ الحالة الرقمية على الطرف 13 ، و تكون 0 أو 1
`analogRead(A0);`	يقرأ قيمة الجهد على مدخل تماثلي و يكون الناتج 0-1023
`analogWrite(3,255);`	يخرج جهد تماثلي على المخرج 3 قيمته 5v لاحظ أن الأطراف 3,5,6,9,10,11 تقبل الخرج التماثلي
`tone(4,300,1000);`	4 رقم المخرج ، 300 التردد ، 1000 زمن النغمة المزيد

`noTone(4);`	إطفاء النغمة على المخرج 4

أوامر للتأخير وحساب الزمن Delay & Time commands

`delay(300);`	تأخير زمني لمدة 300 ميللي ثانية
`unsigned long x = millis();`	يجب أن يكون نوع المتغير unsigned long ويوضع فيه عدد ms منذ بداية تشغيل البرنامج.
`unsigned long y=micros();`	مثل الأمر السابق لكن بالمايكرو ثانية µs
`delayMicroseconds(12500);`	تأخير زمني بالمايكرو ثانية
`pulseIn(10,HIGH)`	يقيس زمن النبضة بالمايكروثانية المزيد

المتغيرات Variables هي أسماء ، قد تكون حرف مثل (X) أو كلمة مثل (input)

وتكون لها قيمة عددية عادة مثلا : x=10 أو sensorPin = 0

ملاحظة : تختلف الأحرف الكبيرة عن الأحرف الصغيرة ، لذا قد تجد أن x=10 و X=15

`int x = 0;`	متغير يحمل رقم صحيح -32K إلى 32K تقريباً
`const int m=7;`	إذا كانت قيمة المتغير لن تتغير أثناء عمل الكود (غير ضروري)
`static int n=10;`	تعريف متغير في loop بدون أن يعيد وضع قيمة لنفسه كل دورة
`float y =16.25 ;`	متغير يحمل رقم كبير و يقبل الفاصلة العشرية 15.78 ملاحظة يجب كتابة 5.0 وليس 5 فقط . (4B_4B-)
`byte a = 10;`	0-255 مثل القيمة التي يطلبها الأمر analogWrite
`long var = 123456789;`	متغير يحمل رقم كبير (يستخدم 4 بايت) -2B إلى 2B
`unsigned long X=120000000;`	متغير يمكنه حمل رقم كبير 0 إلى 4.3 بليون
`bool z=0;`	z=1; z=0; z=HIGH; z=LOW ; متغير يحمل قيمة منطقين (نعم أو لا) 1 أو 0
`char v = 'W';`	متغير يحمل قيمة حرف ASCII
`String s= "hey everyone " ;`	مجموعة من الحروف (ASCII) تسمى String
`char z[] = "Hello world!";`	مصفوفة من الحروف و تسمى string
`int Pins[]= {2, 4, 8, 3, 6};`	مصفوفة أرقام array، و للحصول على أحد العناصر اكتب x = Pins[2] >>> 8

يمكنك تعريف عدة متغيرات من نفس النوع في أمر واحد	`int x=2 , y , z ;`

العمليات الحسابية arithmetics

إعطاء قيمة لمتغير assigning a value	`int x = 35 ;`
إجراء العمليات الحسابية المعروفة operators	`/ * - +`
يعيد لك باقي القسمة فقط modulo مفيد في تطبيقات قليلة ولا يعمل مع الـ floats	`y = 9 % 3 >> y=0` `x = 10 % 3 >> x=1`
أمر مختصر يعمل على زيادة أوطرح 1 من قيمة المتغيرi	`i++; i-- ;`
أمر مختصر يعمل على زيادة 3 أو أي رقم آخر لقيمة المتغير .. كأنها x = x+3 مثلاً	`x+=3; x-=3;`
يحسب 5 أس (للقوة) 2 = 25	`pow(5,2);`
يحسب الجذر التربيعي لـ 16 و يساوي 4	`sqrt(16);`
يعيد القيمة المطلقة (بدون سالب) لـ x	`abs(x);`
حساب الدوال المعروفة sin , cos , tan	`sin( );cos( ); tan( );`
حساب اللوغاريتم logarithm	`log( );`
يعود بقيمة عشوائية من صفر إلى 9	`random(10);`
يعود بقيمة عشوائية من 5 إلى 14	`random( 5 , 15 );`
يعود بأعلى قيمة من بين القيمتين	`max(x,y);`
يعود بأقل قيمة من القيمتين	`min(x,y);`
تحويل قيمة من نطاق رقمي إلى نطاق رقمي آخر مثال: تحويل قيمة من مدخل تماثلي (1023) إلى قيمة لمخرج تماثلي (255) لا يعمل مع أعداد كسرية	`map(x,0,255,0,5000);`
إذا تجاوزت قيمة x الحدود 0 و 100 فإن الأمر يعيد القيمة لتبقى داخل النطاق المطلوب	`constrain(x,0,100 );`

الشروط conditions والحلقات loops : مقارنة تحدد تنفيذ الأوامر في الكود.

`if (x == 10) { … }`	سوف يتم تنفيذ ما بين الأقواس { … } فقط إذا تحقق الشرط بين الأقواس ( )
`else if ( x > 20 ) { … }`	يمكن إضافة هذا الأمر بعد الـ if
`else { … }`	يمكن أضافة الأمر بعد الـ if
`while ( x <= 5) { … }`	أبسط نوع من الـ loop سيتم تنفيذ ما بين القوسين بشكل مستمر { .. } فقط مادام الشرط محقق (…)
`for(int i=0;i<100;i++){…}`	دائرة for تستخدم عند الرغبة في تكرار أمر لعدد محدد من المرات
`break;`	يعمل على الخروج من الـ دائرة loop الحالية
`goto label` `label: ...`	ينتقل مباشرة إلى مكان آخر في الكود
`return;`	ينهي استدعاء دالة أو يعيد قيمة (متقدم، عند كتابة دوال)

جميع الشروط conditions :

`(x>10)`	يتحقق الشرط إذا كانت قيمة x أكبر من 10 ويكون الجواب نعم (1) أو لا (0)
`>  <  >=  <=  ==  !=`	باقي الشروط (انتبه لعلامة ==)
`(x>10 && y!=0)`	يجب أن يتحقق الشرطين معاً (AND)
`(x<10 || x>30)`	يجب أن يتحقق أحد الشرطين (OR)

التعامل مع شاشة المتسلسلة Serial monitor

`Serial.begin(9600);`	تشغيل الشاشة المتسلسلة مرة واحدة في البداية لاحظ: عند تشغيل السيريال لا يمكن استخدام الأطراف 0 و 1
`Serial.println( x );`	إظهار قيمة (x) على الشاشة ثم سطر جديد
`Serial.print( "\t \n hello world");`	كتابة عبارة (string) \t : tab \n : new line
`while(!Serial.available()){}`	طريقة لإيقاف الكود في انتظار قيمة من المستخدم

`while(Serial.available()==0){}`	
`if (Serial.available()>0) {...}`	طريقة لاستقبال أي قيمة من المستخدم عبر الشاشة المتسلسلة serial monitor
`char x=Serial.read(); if (x=='y'){...}`	قراءة بايت من الشاشة المتسلسلة ككود ASCII لمعرفة الرقم المدخل اطرح 48 من القراءة
`Serial.parseInt();`	لقراءة قيمة ووضعها في متغير int
`Serial.parseFloat();`	لقراءة قيمة و وضعها في متغير float
`Serial.readString();`	لقراءة متغير (عبارة) و وضعها في String

أوامر متقدمة نسبياً ...

`int x = 0x8C ;`	كتابة قيمة بالنظام السداسي عشر .
`int y=0b10001100 ;`	كتابة قيمة بالنظام الرقمي الثنائي
`int z= bitRead(y,2);`	لاستخلاص بيت واحد من قيمة
`bitWrite(y,0,1);`	للكتابة على بت واحد -المثال يكتب 1 في الـ LSB
`shiftOut(13,12, LSBFIRST,0b11001000);`	مصمم للتعامل مع مسجلات الإزاحة shift registers 13 يستخدم لإرسال الإشارة (البيانات) 12 سيستخدم كساعة تزامن LSBFIRST أو MSBFIRST القيمة يستحسن أن تكون بالنظام الثنائي 0b11101
`attachInturrupt(0,goFast, FALLING);`	لتنفيذ مقاطعة عند تطبيق إشارة على الأطراف D2,D3 المعطى الأول 0 : D2 _ أو 1: D3 _ سوف يستدعي الدالة اسمها goFast المعطى الثالث يمكن أن يكون FALLING , RISING , CHANGE
`noInterrupts();`	يوقف تنفيذ أي مقاطعة (الافتراضي استشعار المقاطعات)
`interrupts();`	يعود لاستشعار المقاطعات (بعد الأمر السابق)
https://www.arduino.cc/en/Reference/HomePage	ولمشاهدة قائمة الأوامر من الموقع الرسمي :

تـــــمـــــهـــــيـــــد 😐

لايـــوجـــد عدد محدد من الأفكار أو المشاكل التي يمكنك حلها بالأردوينو ،،، فقط تأمل حولك وستجد عشرات أو مئات المشاكل التي يمكن حلها أو تحسينها باستخدام المتحكمات الإلكترونية.

مشاكل في المنزل ، في السيارة ، في المصانع ، في الحدائق وغيرها الكثير ...

ما هو الحساس المناسب؟ ماهو الفعل المناسب؟

وعادة ستجد الأردوينو يربط المكونات المختلفة و يقوم بالتحكم الإلكتروني اللازم.

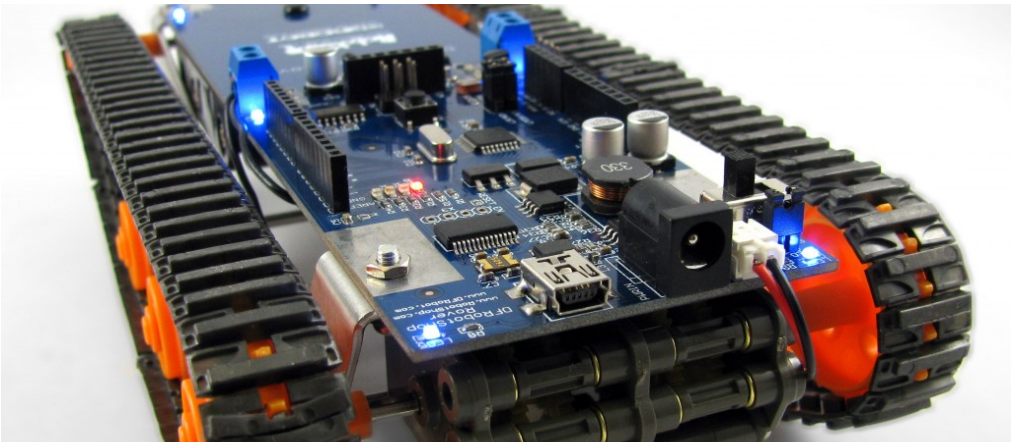
سأحاول في هذا الكتاب أن أسير معك خطوة بخطوة حتى تفهم و تستفيد من قدرات (الأردوينو) الرائع

سنجيب عن الأسئلة : ماهو الأردوينو ؟ أنواعه ؟

كيف يمكنني التحكم بالأجهزة الكهربائية به؟

وكيف أدخل إشارات الحساسات له؟

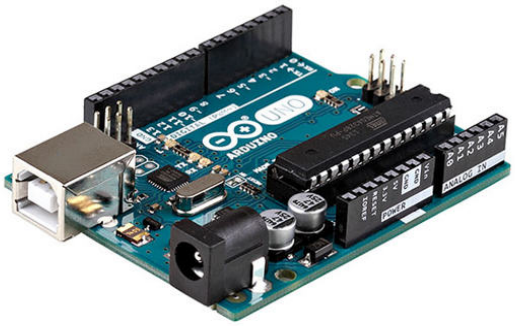
و تركيزنا الأول هو البرمجة ... **كيف أبرمج الأردوينو ليقوم بعمل محدد؟**



أنا الآن في الأسطر الأولى من الكتاب ، لكن يظهر لي أن الكتاب سيكون طويل ...

عفواً ... كنت أجرب الإيموجي في هذا البرنامج ... أتمنى أن تستمتع معي في هذا الكتاب.

م.سامي قرامي (كتبه في : 2017-3-29 الساعة 11:46 بنص الليل)



ماهو الأردوينو ؟ what is the Arduino

الأردوينو هو لوحة إلكترونية صغيرة الحجم، رخيصة السعر وسهلة الاستخدام .
تعمل كمتحكم ورابط بين المكونات الكهربائية المختلفة .
يتم التحكم بطريقة عملها ببرمجتها.
مثال على المداخل : لوحة أزرار ، مقياس ضوء، حساس حرارة.
مثال على المخارج: لمبات ، محركات ، صوت ، شاشة.

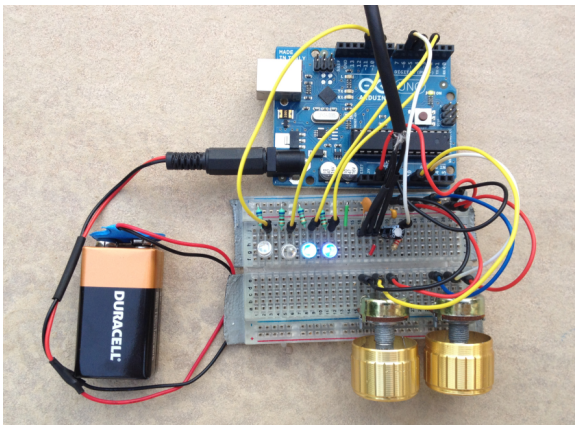
أهم مميزات الأردوينو : advantages

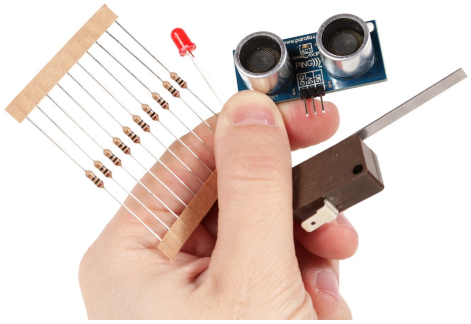
انخفاض السعر (أردوينو أونو سعره حوالي 10$ = 40 ريال)
سهولة الإستخدام (مقارنة بغيره من الدوائر المبرمجة)
كثرة الإضافات المتوافقة مع الأردوينو و التي تقوم بأعمال متنوعة وتسمى shields
موقع الإنترنت الخاص بالأردوينو منظم ومفيد جداً arduino.cc
الشهرة الواسعة وآلاف المستخدمين و الدروس و المشاريع عبر العالم

العيوب : Disadvantages

تعتبر القدرة البرمجية للأردوينو أقل بكثير من الكمبيوتر ، ولا يتمكن من تشغيل برامج مثل الويندوز أو الأندرويد أو تشغيل شاشات ذات وضوح عالي و هكذا . هذه الأعمال تناسب الكمبيوتر أو الراسبري باي والذي قد نتحدث عنه في كتاب آخر.
التيار الذي يتمكن الأردوينو من إخراجه من المنافذ = 20mA وهذا لا يكفي لتشغيل محرك أو مرحل (ريلاي) عادة . لذا يجب استخدام عناصر إلكرتونية لتكبير الطاقة الكهربائية في بعض التطبيقات.

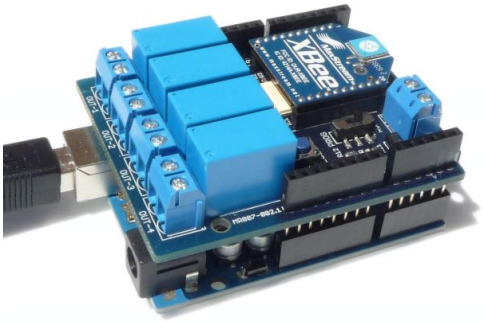
مع الأردوينو ومع القليل من الإلكترونيات ستتمكن من تنفيذ مشاريع إلكترونية أو كهربائية تقوم بأعمال رائعة كثيرة مثل: التحكم بالإضاءة أو تحريك الأشياء بالمحركات المناسبة أو تشغيل شاشة بسيطة أو إصدار صوت أو الوصول للانترنت...





يوجد أنواع عديدة من الأردوينو (standard boards) مثل **UNO** و **MEGA** و **micro** وغيرها ، و الرائع أنه يمكن إضافة خصائص إضافية على الأردوينو بتركيب دوائر ملحقة بالبورد الأساسي و تسمى: **shield boards**

هذه الإضافات كثيرة و لها مهام متعددة مثل:

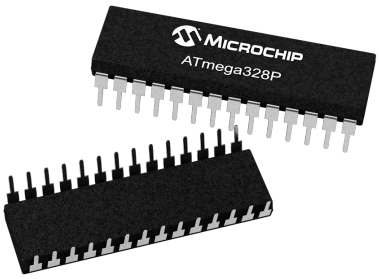


Ethernet Shield ($45)
4 Relay Shield ($20)
Protoshield (~$15)
LCD Screen ($20)
Motor drivers
USB Host to control USB devices

where did it all start ؟ كيف بدأ الأردوينو

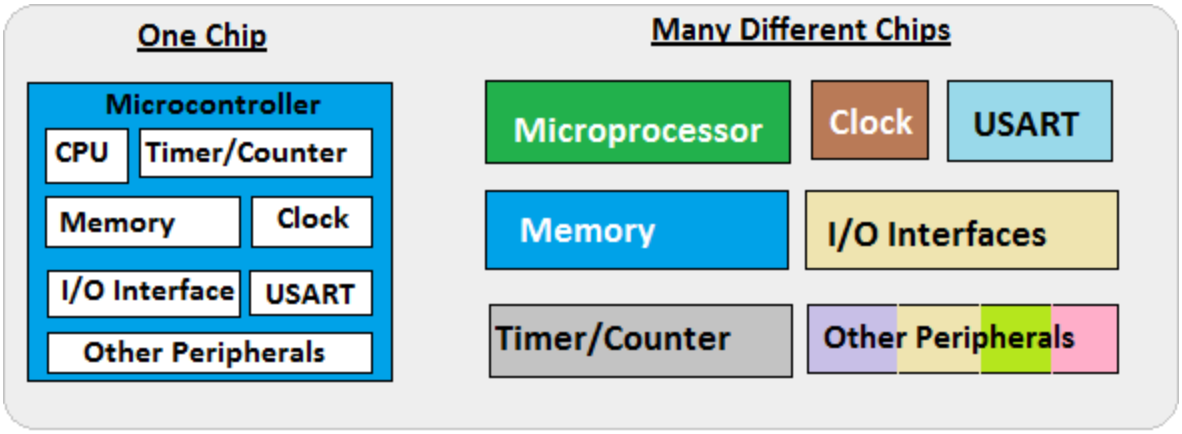
تم تطوير الأردوينو عام 2005 ليساعد في تعليم الطلاب الإلكترونيات و البرمجة. كان ذلك في إيطاليا .

سهولة استخدامه و انخفاض تكلفته و موثوقيته جعلته ينتشر بسرعة حول العالم. السبب الآخر لانتشار الأردوينو أنه (مفتـوح المصدر) وهذا يعني أن أي شركة بإمكانها تصنيع الأردوينو و تطوير الملحقات الخاصة به بدون أي تعقيدات وحقوق ملكية.



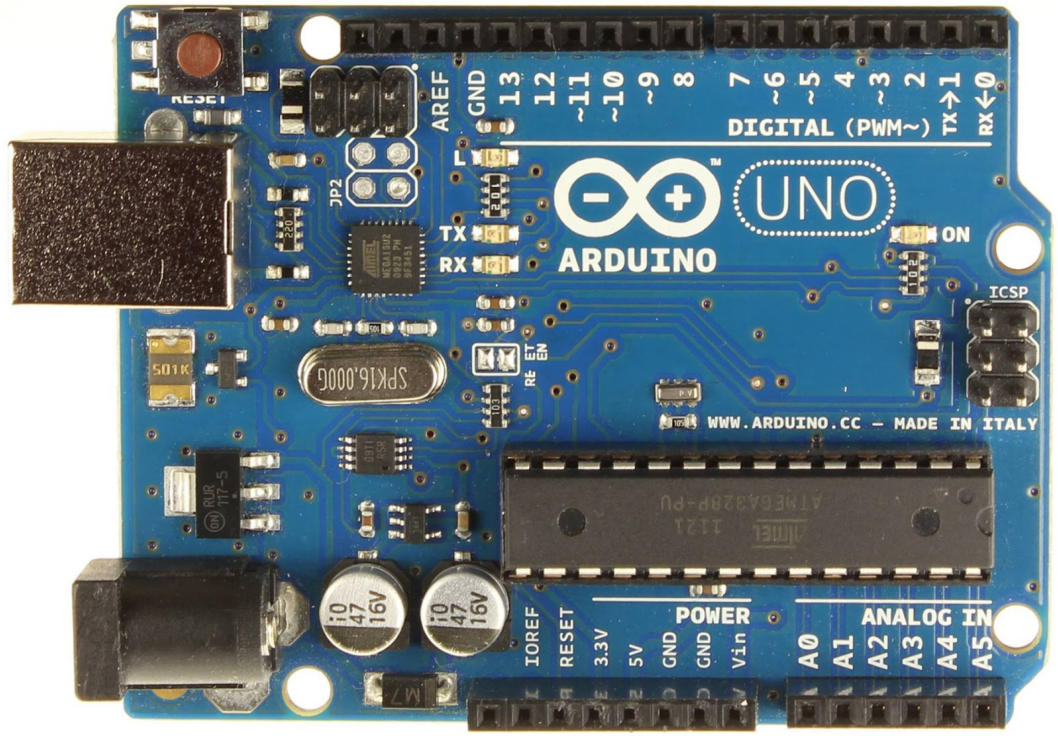
المايكروكنترولر : Microcontroller هو شريحة إلكترونية تجمع معظم مكونات الكمبيوتر في شريحة بسيطة واحدة (معالج، ذاكرة ، مواجهة دخل و خرج)

المايكروكنترولر لا يتمكن من تشغيل التطبيقات المتقدمة مثل الكمبيوتر _ فوتوشوب أو ايتيونز مثلاً



يمكن إدخال أو اخراج إشارة تماثلية أو رقمية إلى الشريحة (المايكروكنترولر)

تعرف على بورد الأردوينو أونو: Arduino UNO



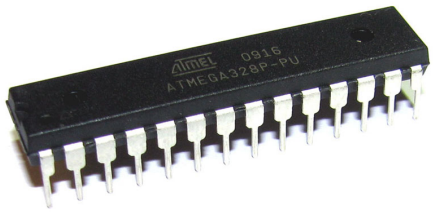
أسفل اليمين شريحة المايكروكنترولر **ATmega328p** قابلة للاستبدال وبها 28 طرف pin

لاحظ أعلى اليسار **مدخل سلك الـ USB** لونه فضي- يستخدم لتشغيل الأردوينو وربطه بالحاسب للبرمجة

مدخل الطاقة external power يمكن استخدامه لتوصيل الطاقة للأردوينو من شاحن أو بطارية

أعلى اليسار يوجد زر **Reset** أحمر . عند الضغط عليه يعيد الأردوينو تشغيله (إعادة تنفيذ البرنامج)

بجانب زر **Reset** يوجد 6 دبابيس معدنية . يمكن استخدامها بدل منفذ الـ USB للاتصال مع الكمبيوتر، لكننا لن نحتاجها في هذه الدورة.



داخلياً تحتوي شريحة الـ ATmega328p على التالي:

معالج CPU يعمل بتردد 16MHz

ذاكرة RAM سعتها 2KB

ذاكرة فلاش Flash Memory وسعتها 32KB وتستخدم لتخزين الكود sketches

ذاكرة EEPROM سعتها 1KB وتستخدم لتخزين بيانات إضافية (غير الكود)

دائرة للربط مع المنفذ التسلسلي المستخدم في البرمجة وتسمى UART

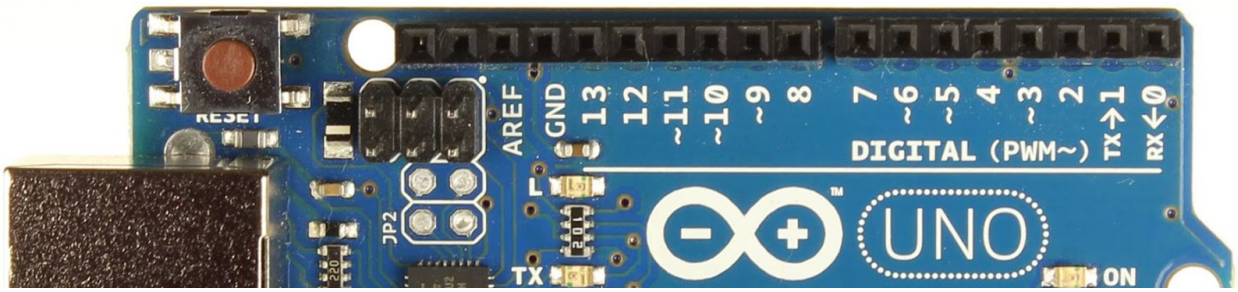
دوائر لتشغيل منافذ الدخل و الخرج I/O ports

ويمكنك شراء شريحة ATmega328P بحوالي : $5 = 20 ريال سعودي تقريباً

المنافــذ pins i/o ports

لاحظ أن اللوحة الزرقاء (البورد) تحتوي كتابات توضيحية كثيرة أمام كل منـفـذ . و هذا يساعدك على معرفة عمل كل طرف. و سنتحدث عن هذه مجموعات المنافذ الآن .

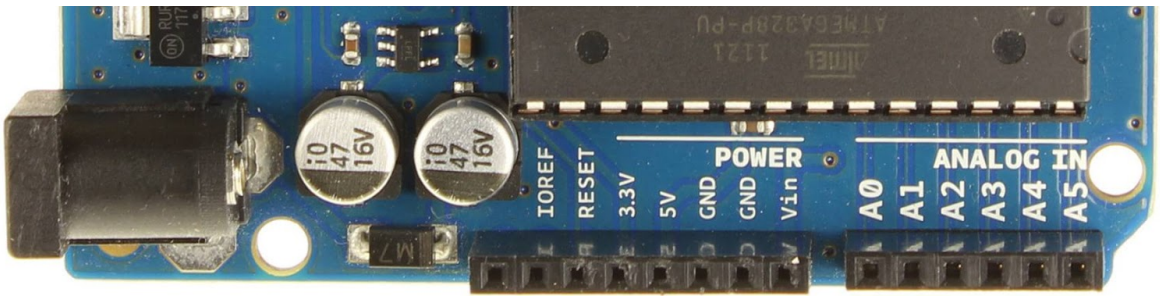
المنافذ الرقمية Digital Pins



المنافذ الرقمية عددها **14 منفذ** ... وهي مرقمة (0 - 13) ويمكنك في الكود تحديد عمل كل منفذ
عندما تعمل المنافذ كمخارج ؛ يمكنك حسب كتابة الكود إخراج 5v أو 0v
كما يمكنك جعل هذه المنافذ تعمل كمداخل رقمية (لاستشعار حالة زر مثلاً)
المنـفـذ الرقمي يمكن أن يمدّ الحمل (الشيء المتصل بالمنـفـذ) بـ 5v و أمبير 20mA . هذا التيار مناسب لتشغيل مبين ضوئي LED لكنه بالتأكيد لا يكفي لتشغيل محرك.

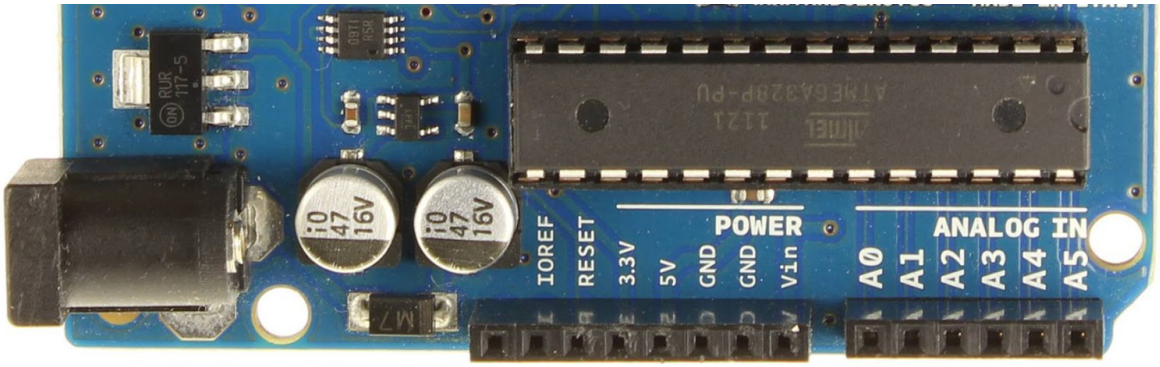
يوجد مبين ضوئي LED صغير بجانب المنفذ 13 وهو يعمل عندما يكون المنفذ 13 Hi
استخدام المنفذ 13 أصعب كدخل ، و ذلك بسبب الـ LED المتصل معه ، حاول استخدام منفذ آخر.
المنفذ GND يعمل كأرضي للدائرة الإلكترونية 0v
العلامة ~ تعني أن هذا الطرف يصلح لإخراج قيمة جهد تماثلية. ويسمى أيضاً PWM
المنفذين(0,1) يسميان RX , TX ويستخدمان للتواصل مع الكمبيوتر (ملاحظة: إذا استخدمت الأمر Serial.begin في الكود فلا يمكنك استخدام المنفذين 0,1 كمنافذ رقمية)
المنفذ AREF نادر الاستخدام ويستخدم لضبط أعلى قيمة في نطاق الجهود للمداخل التماثلية (0-5v)

منافذ الطاقة : في الجهة السفلية يسار (Power Pins)



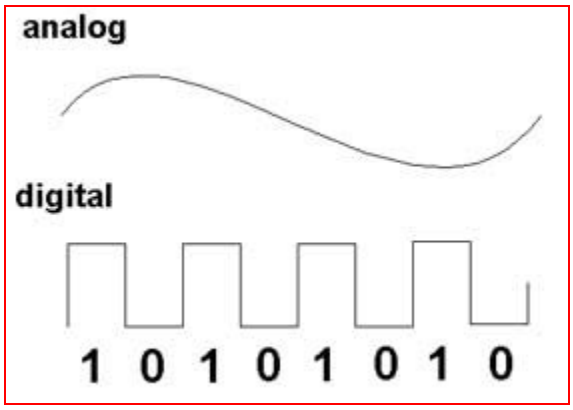
بعد تشغيل الأردوينو و توصيل الطاقة المناسبة له ، يمكنك أن تستخدم هذه المجموعة لتمد دائرتك الإلكترونية بالطاقة المناسبة (5v أو 3.3v) الطرفين GND تسمى الأرضي و جهدها 0v
لاحظ أيضاً يمكنك أن تمدّ الأردوينو بالطاقة عبر توصيل جهد مناسب (7v - 12v) إلى الطرف Vin
كما يمكنك أن تعمل Reset إعادة تشغيل للأردوينو عبر استخدام المنفذ Reset
(لعمل هذا : وصل المنفذ reset بـ GND)

المداخل التماثلية Analog inputs



عددها 6 (A0-A5) و يمكنها قياس الجهد (تماثلياً) _ ويكون التعامل معها بتوصيلها مع السلك المطلوب قياس الجهد عنده ، ثم التحكم بها في البرنامج .

ملاحظة : يمكن استخدام هذه الأطراف كمداخل رقمية أو مخارج رقمية . سنشرح لاحقاً.



الفرق بين الإشارات التماثلية و الإشارات الرقمية:-

باختصار الإشارة الرقمية ON/OFF مثال: الضوء شغال او طافي فقط

بينما الإشارة التماثلية يمكن أن نتحكم بقيمة الجهد فيها فيكون مثلاً : 0v , 1v , 3.3v 4.9v

بهذا يمكن تشغيل اللمبة بتدريج (ضوء عالي ، منخفض، منخفض جداً ، و هكذا ...)

مثال للتفرقة بين التماثلي و الرقمي ::

رقمي_ اللمبات العادية (ON-OFF) فقط

تماثلي _ اللمبات التي تقبل التحكم بشدة الإضاءة _ dimmer

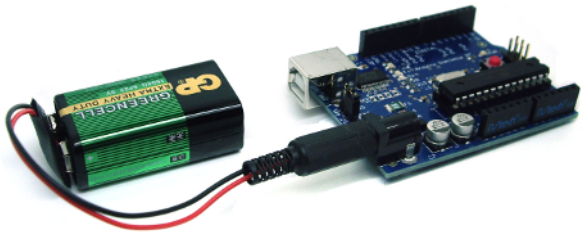
```
digitalWrite(13,1);
analogWrite(3,127);
```

يمكن تشغيل الأردوينو (توصيله بالطاقة) بثلاثة طرق

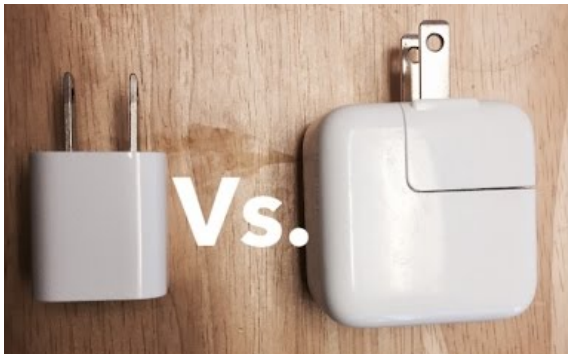
1- من الكمبيوتر أو شاحن إلى مدخل الـ USB

2- مدخل الطاقة 5.5mm-2.1mm jack ويجب البحث عن المحول المناسب 12v - 7v

3- تطبيق جهد الدخل 12v - 7v مباشرة على المنفذين Vin و GND



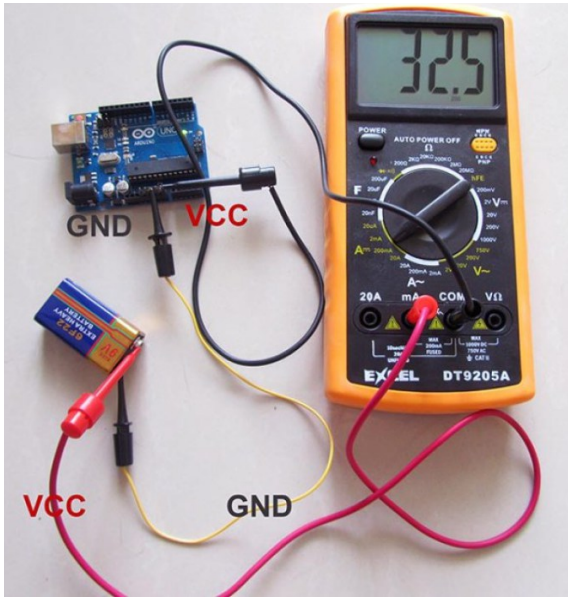
حساب التيار الذي يسحبه الأردوينو و باقي الدائرة



كما تعلم فـليست جميع الشواحن متساوية في الجهد والتيار الكهربائيين (شاهد مثلا شاحن الايفون و شاحن الايباد)

الأردوينو أونو يسحب 45mA عادة . لكن كلما شغلت ملحقات أكثر فإن سحب التيار سيزيد (مثلاً إضاءة ، صوت ، مرحلات...)

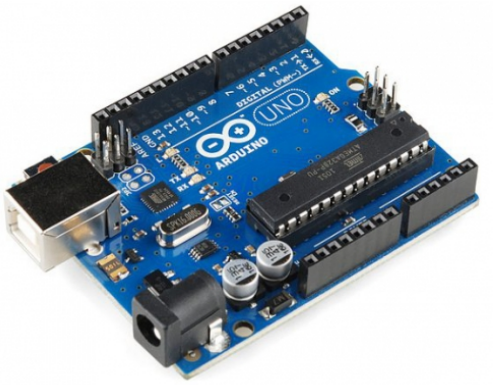
ويمكن قياس التيار المستهلك من الدائرة باستخدام أجهزة مناسبة.



ملاحظة : التيار الذي يستطيع الأردوينو إخراجه من المنفذ الرقمي الواحد حوالي 20mA
والتيار الذي يمكن سحبه من الطرفين Vcc و GND يكون حوالي 200mA

عـائلة أردوينو Arduino Family

توجد إصدارات متعددة من بوردات الأردوينو و بينها اختلافات في الحجم والخصائص التقنية (مثل الذاكرة وعدد المنافذ) جميع البوردات تتبرمج بنفس الطريقة فلا تقلق



Arduino UNO R3

أشهر نوع من بوردات الأردوينو وقد شرحناه وعرضنا صوره بالتفصيل سابقاً . معظم هذا الكتاب سنركز عليه.
أهم ميزة رائعة في هذا البورد هو إمكانية استبدال شريحة الـ مايكروكنترولر. هذا يتيح لك تصنيع دائرتك الخاصة بعد برمجة الشريحة.سوف نشرح هذه الميزة الرائعة في مراحل متقدمة في الدورة وفي الكتاب.

Arduino Leonardo

لا يختلف في شيء عن الـ arduino UNO إلا أن شريحة المايكروكنترولر مثبتة ولا يمكن استبدالها . والمنفذ الـ USB المستخدم هو micro USB وليس USB typeB مثل السابق. سعره أرخص من الـ UNO أيضاً

أردوينو يون Arduino YUN

يتميز اللـ YUN بأنه مدمج مع واي فاي WiFi . كما أن الـ "يون" يتيح خيارات برمجية متقدمة بنظام لينيكس.لكن برمجة الـ " يون " تكون في العادة مثل أي أردوينو بواسطة برنامج برمجة الأردوينو Arduino IDE . كما يمكن برمجته لاسلكياً إذا تم تعريف الوايفاي على نفس الشبكة التي يتصل بها الكمبيوتر.
عند بداية تشغيل الـ يوون سينشئ شبكة لاسلكية اسمها

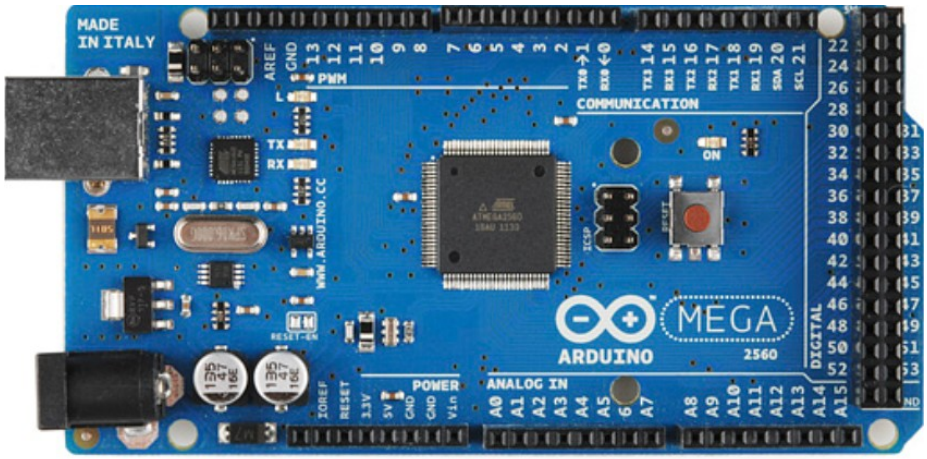
arduinoYun-xxxx

اتصل بها ، ادخل المستعرض ثم أدخل العنوان

192.168.240.1

وأدخل الرقم السري: arduino ستواجهك صفحة فيها اعدادات الـ يوون .

Arduino Mega الأردوينو ميغا

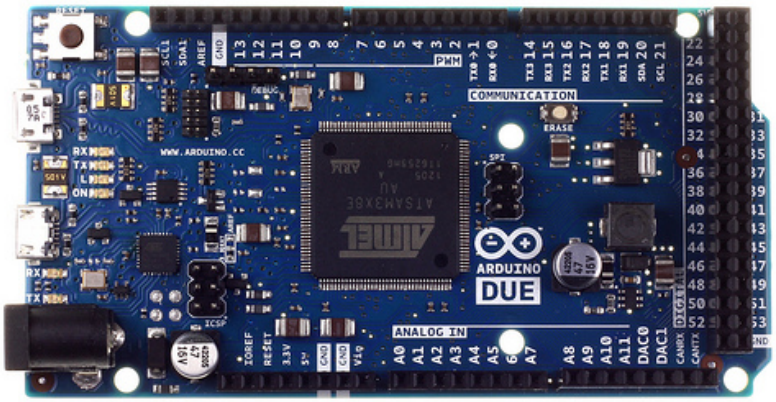


يختلف الأردوينو ميغا عن الأونو بأن له مداخل و مخارج أكثر بكثير والجميل أن الجهة اليسرى من اللوحة تتشابه مع الأونو، و هذا يجعل من السهل استخدام نفس البرنامج ، وتوصيل الأسلاك بنفس الطريقة . أيضا هذا يتيح استخدام نفس الدوائر الإضافية التي تعمل على الأنواع الشائعة من الأردوينو .

*مقارنة بين الأردوينو أونو و الأردوينو ميغا UNO vs MEGA

موديل الأردوينو	المنافذ الرقمية	المداخل التماثلية	ذاكرة الكود	ذاكرة RAM	EEPROM	الأبعاد cm
UNO	14	8	32KB	2KB	1KB	6.8*5.3
Mega	54	16	128KB	8KB	4KB	10*5.3

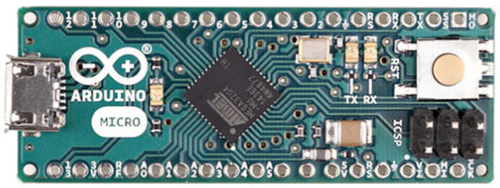
Arduino Due الأردوينو دوو



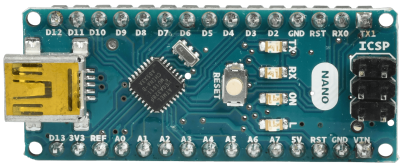
الأردوينو دوو يتشابه مع الميغا في الحجم و عدد المنافذ. لكنه يحتوي معالج مختلف يعمل بسرعة أعلى (84MHz) كما أنه يعمل على جهد 3.3v وليس 5v مثل الأنواع الباقية من الأردوينو . هذا يجعل بعض الدوائر الإضافية لا تعمل على الأردوينو دوو.

الإصدارات الصغيرة من الأردوينو الأردوينو مايكرو ، نانو ، ميني

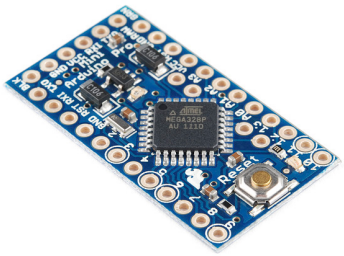
في بعض التطبيقات ستحتاج لتصغير الدائرة قدر الإمكان . توجد موديلات عديدة صغيرة الحجم من الأردوينو . و عيبها هو صعوبة استخدام الدوائر الإضافية Shields المعتادة معها.
(ملاحظة هنا سنضع منتجات شركة أردوينو الأم . لن نضع الأنواع التي تنتجها شركات أخرى)



أردوينو مايكرو Micro
الأبعاد: 48 * 18 mm
المنافذ الرقمية : 20 _ منها 7 pwm
المداخل التماثلية: 12



أردوينو نانو Nano
الأبعاد: 45 * 18 mm
المنافذ الرقمية : 22 _ منها 6 pwm
المداخل التماثلية: 6



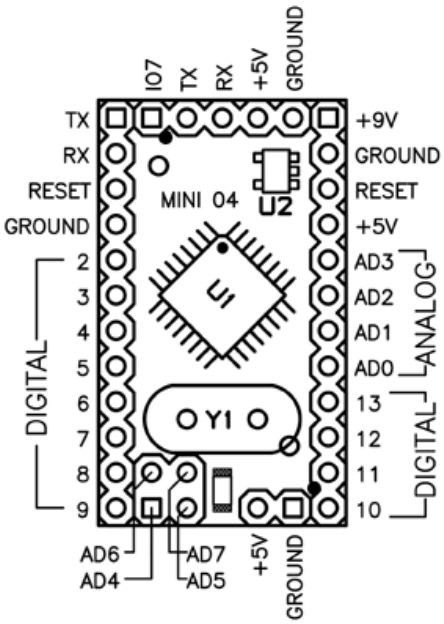
أردوينو ميني Mini
الأبعاد: 30* 18 mm
المنافذ الرقمية: 14 _ منها 6 pwm
المداخل التماثلية: 8
ملاحظة: لا يمكن توصيل الـ ميني إلى الكمبيوتر مباشرة، يجب استخدام دائرة وسيطة.

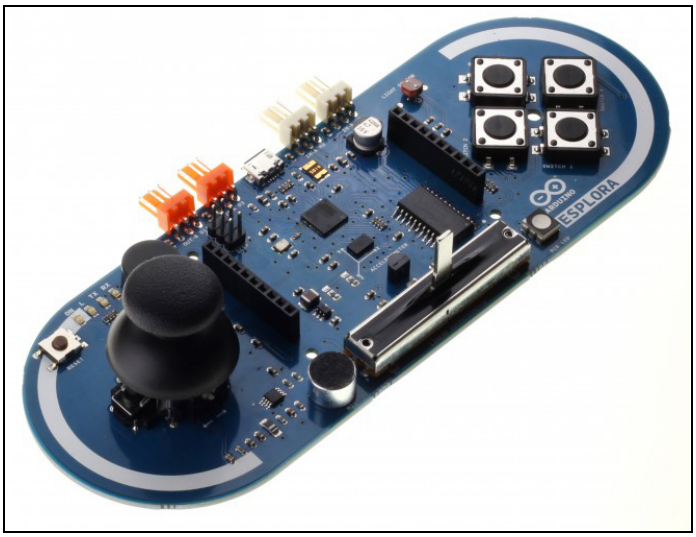
ملاحظة: لكل نوع يجب أن تعرف توزيع الأطراف : موقع المنافذ التماثلية و موقع أطراف التغذية الكهربائية و هكذا . في هذا الكتاب لن نضع صور لكل هذا ، بل سنعلمك الطريقة 😊
فقط استخدم موقع أردوينو

Arduino.cc >> products

ثم اضغط على اسم الأردوينو المطلوب.

أو بحث الصور في قوقل و اكتب اسم الموديل ثم : pinout _
ستجد كل شيء 😘





أردوينو اسبلورا arduino esplora

فكرة الأردوينو إسبلورا أنه لوحة أردوينو مدمج معها العديد من الإضافات المتنوعة ولهذا يمكن اعتبار الإسبلورا مختبر لدمج الحساسات مع البرمجة.

وهو مفيد للتطبيقات التعليمية.

يحتوي الكثير من الاضافات :

حساسات ضوء، حرارة ، صوت ، حركة وميلان

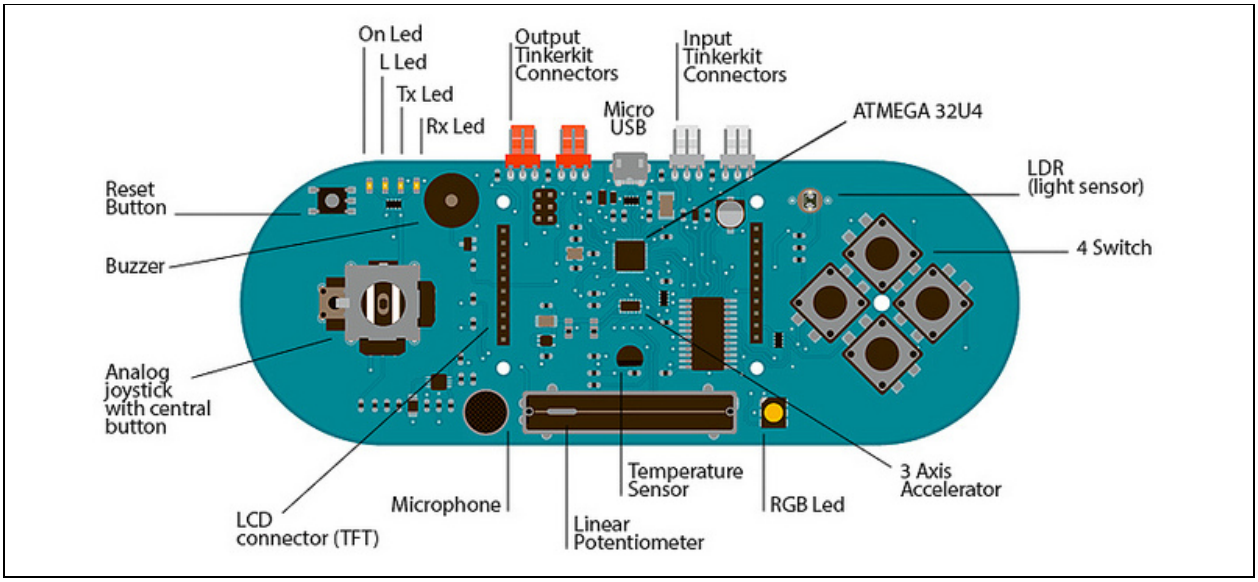
عصا تحكم تماثلية Analog

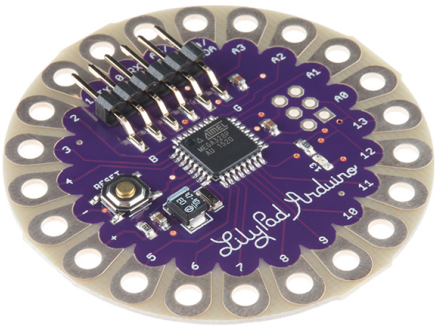
4 أزرار ضاغطة

مقاومة متغيرة خطية slider

جرس (buzzer)

مدخل لذاكرة SD card





أردوينو ليليباد Arduino Lilypad

هذا الموديل مميز وغريب الشكل من الأردوينو . يتميز بأنه رفيع و مصمم لوضعه على الملابس. من عيوبه أنه لا يمكن برمجته بتوصيله إلى الكمبيوتر ، بل يحتاج إلى دائرة إلكترونية إضافية حتى تتم برمجته.

لمشاهدة جميع أنواع الأردوينو الرسمية (من الشركة الأم) اذهب إلى موقع Arduino.cc

Arduino.cc >> products

أو اضغط على الرابط هنا

في الصورة أسماء أشهر أنواع الأردوينو _ فقط اضغط على اسم الموديل وستفتح صفحة فيها كل المعلومات التي تحتاجها عن هذا الموديل (باللغة الإنجليزية طبعاً ☺)

ENTRY LEVEL	UNO	LEONARDO	101	ROBOT	ESPLORA	MICRO	NANO	MINI
ENHANCED FEATURES	MEGA	ZERO	DUE	MEGA ADK	PRO	M0	M0 PRO	MKRZERO	PRO MINI
INTERNET OF THINGS	YÚN	ETHERNET	TIAN	INDUSTRIAL 101	LEONARDO ETH	MKRFOX 1200	MKR1000		
WEARABLE	GEMMA	LILYPAD ARDUINO USB	LILYPAD ARDUINO MAIN BOARD	LILYPAD ARDUINO SIMPLE				

بوردات الأردوينو من تطوير شركات أخرى

بعد النجاح والانتشار الكبير للأردوينو، إلى جانب سماح الشركة الأصلية للمطورين بتطوير بوردات جديدة بدون تعقيدات قانونية و حقوق ملكية وغيره . ظهرت شركات كثيرة تسعى لتطوير الأردوينو بما تراه مناسباً . بعضها صنعت بوردات أردوينو أصغر . أو مدمجة مع وايفاي أو بلوتوث، أو مترابطة مع تطبيق هاتف ذكي أو مربوطة مع موقع انترنت.

لكن جميعها يتم برمجتها بالطريقة البسيطة المعتادة مثل الأردوينو العادي.

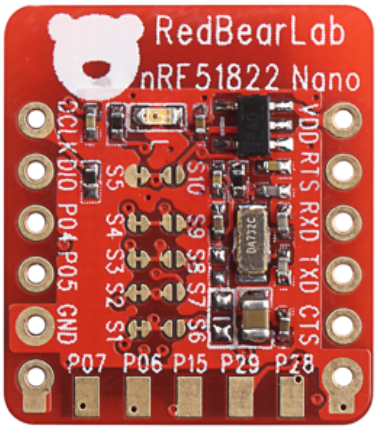
سوف نعرض لك هنا بعض الشركات و الموديلات المشهورة و المتميزة.

redbearlab.com

شركة تطور بوردات مدمج معها بلوتوث ومتوافقة مع تطبيق للهواتف الذكية (تشكيلة رائعة تستحق الاهتمام)

RedBearLab

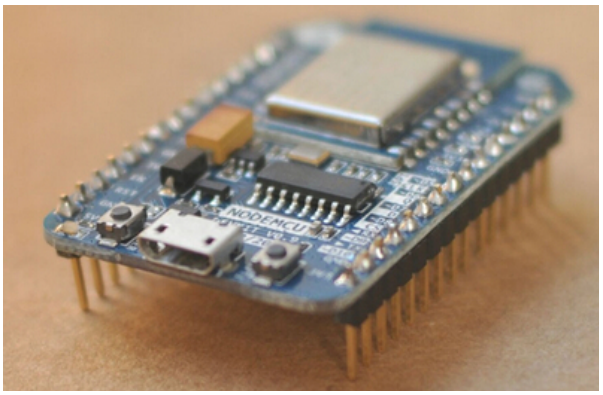
شاهد فيديو



nodemcu.com

شركة تصنع بوردات أردوينو يكون مدمج معها وايفاي

شاهد فيديو

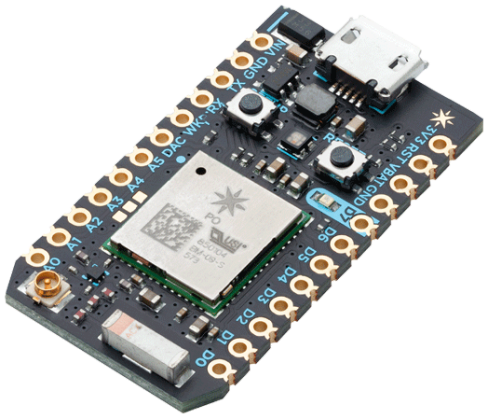


etherTen

بورد أردوينو مدمج معه مدخل شبكة ethernet ومدخل ذاكرة SD card

freetronics
www.freetronics.com

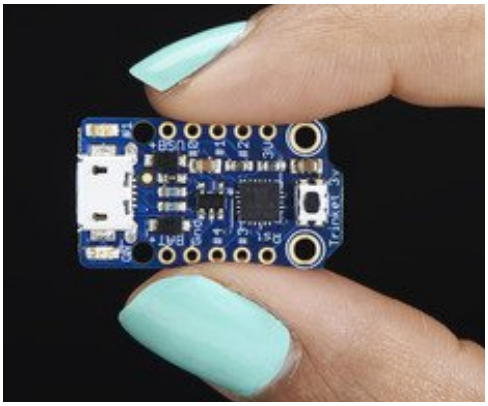
شاهد فيديو



Particle Photon

بورد أردوينو تم تطويره بحجم صغير مدمج معه وايفاي

شاهد فيديو



Adafruit Trinket

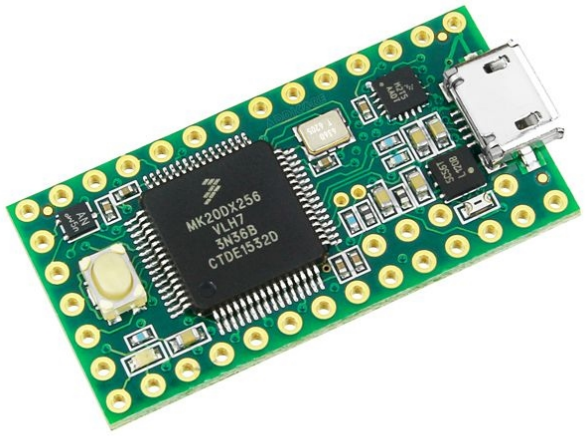
تشكيلة من بوردات الأردوينو الصغيرة جداً و تستحق الاهتمام

adafruit

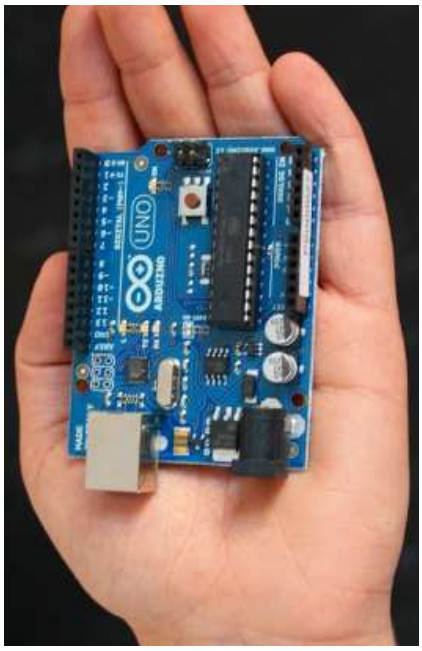
شاهد فيديو

teensy

تعمل على تصنيع عدد من البوردات المتنوعة
عادة بسرعات عالية وإضافات مدمجة عديدة
معظمها تعمل على 3.3v وليس 5v

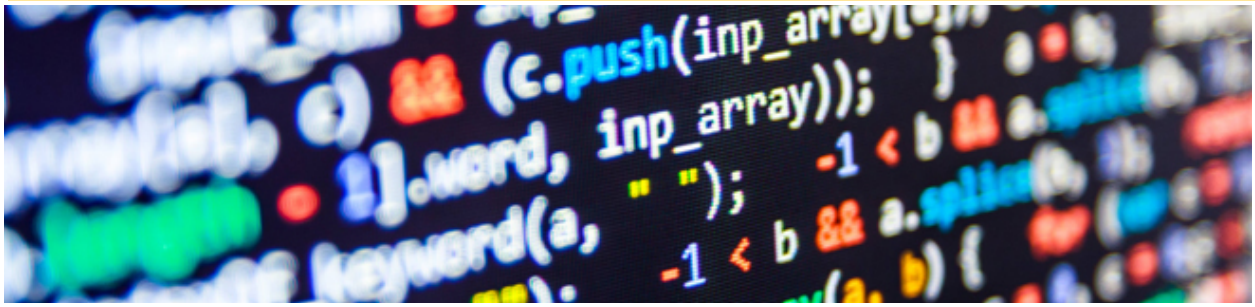


شاهد فيديو



*في هذا الكتاب لن نتمكن من تغطية الأنواع المختلفة الكثيرة جدا من أنواع الأردوينو من الشركات المطورة. سنكتفي بالتركيز على بوردات الأردوينو المصممة من الشركة الأم الشركة الأولى ARDUINO و بالتحديد **Arduino UNO** لأن له ميزة رائعة جداً سنتحدث عنها لاحقاً

الباب الثاني: البـرمـجـة Coding



تمهيد لفهم البرمجة: جميع الكمبيوترات و الهواتف الذكية تعمل ببرامج (كود)

البرامج (الأكواد) هي نصوص مكتوبة تم كتابتها من قبل مبرمجين لتـقـوم بعمل معين.

البرمجة تختلف عن استخدام التطبيقات بالفأرة أو بلمس شاشة جوالك. البرمجة هي كتابة أسطر من الأوامر بلغات وسيطة بين الإنسان و الكمبيوتر .و هذه العملية ليست بسهولة استخدام شاشة اللمس أبداً

البرمجة: هي كتابة سلسلة من الأوامر لينفذها الكمبيوتر

الطفل يستخدم التطبيقات _____ التي برمجها المبرمجون

اذا كانت لديك خلفية بسيطة في البرمجة ستعرف أنه توجد لغات برمجة كثيرة مثل:

Java , Python , HTML , Objective-C

بالنسبة للأردوينو فتتم برمجته بلغة : C ولغة ++C

وهي من اللغات الأساسية للبرمجة ومشهورة للغاية.

لذا إذا أحببت تعلم البرمجة فمن أفضل الطرق أن تبدأ مع الأردوينو و لغة C.

كيف أبدأ بكتابة كود للأردوينو

إذا كنت قد اشتريت الأردوينو وتريد تشغيله فأمامك طريقتين للبرمجة:

1-تحميل برنامج الأردوينو Arduino IDE على جهاز الكمبيوتر الخاص بك.

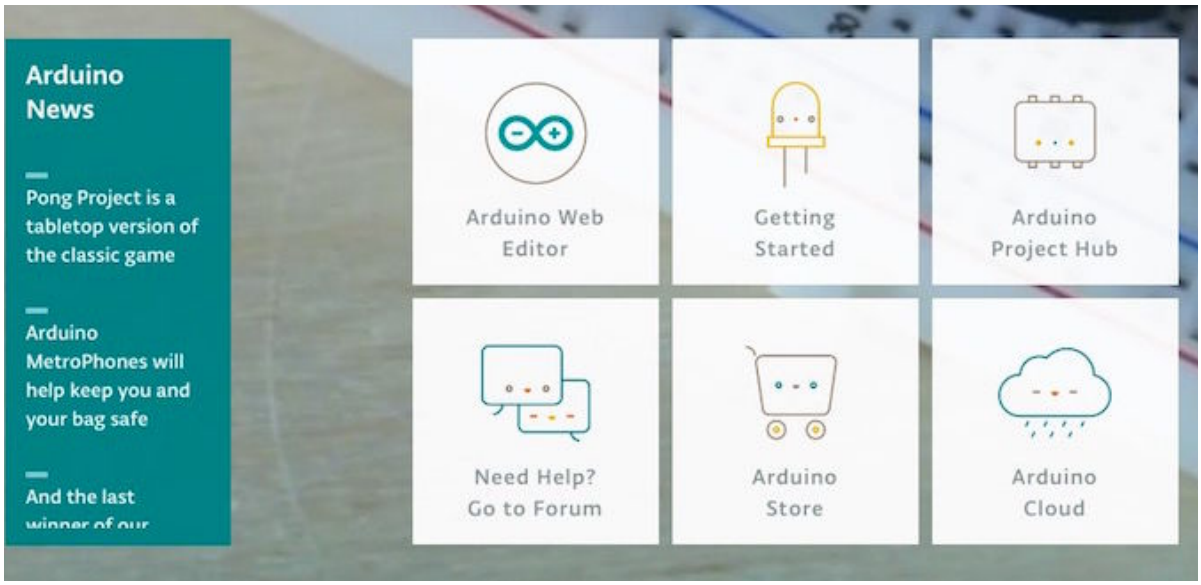
في موقع **google** اكتب **arduino** وادخل الموقع الرسمي **arduino.cc**
في أعلى الصفحة ستجد زر في القائمة اسمه **software** اضغط عليه
ستجد خيارات عديدة للبرنامج : تحميل لويندوز أو ماك أو لينيكس.

2- استخدام مبرمج الأردوينو الحديث على الانترنت Arduino Create

يعتبر تطور كبير على الطريقة السابقة من ناحية التحديثات و التنظيم و سهولة إضافة المكتبات. يعيب هذه الطريقة تعطل السيرفر أو مستعرض الانترنت أحياناً

(أنا شخصياً أفضل هذه الطريقة على الطريقة السابقة)

ملاحظة : سيطلب منك انشاء حساب في الموقع ، كما سيطلب منك تثبيت ملف صغير على الكمبيوتر قبل استخدام **Arduino Create _ web editor**



محاكاة عمل الأردوينو Arduino Simulation

توجد طريقة أخرى رائعة و أحبها كثيراً وهي استخدام الأردوينو افتراضياً 🤩

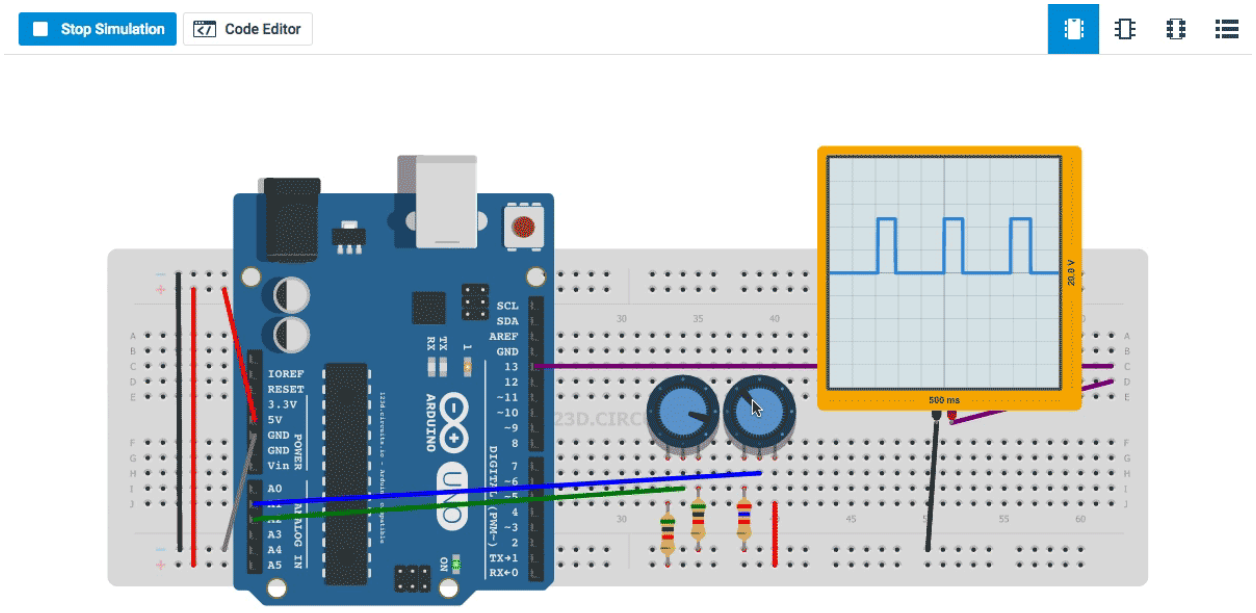
(على موقع انترنت وليس وجوده فعلياً معك)

هذه الطريقة سهلة ، سريعة ، ولا تكلف شيئاً فقط أنشيء حساباً وابدأ في استخدام معمل الإلكترونيات الافتراضي مجاناً. الموقع من تطوير شركة :

AUTODESK.

في موقع **Google** اكتب **tinkercad** وسوف يقودك للموقع :- **tinkercad.com**

اختر (**LEARN**) ثم اضغط على (**CIRCUITS**) وهنا تتبع الخطوات و أنشئ حساباً



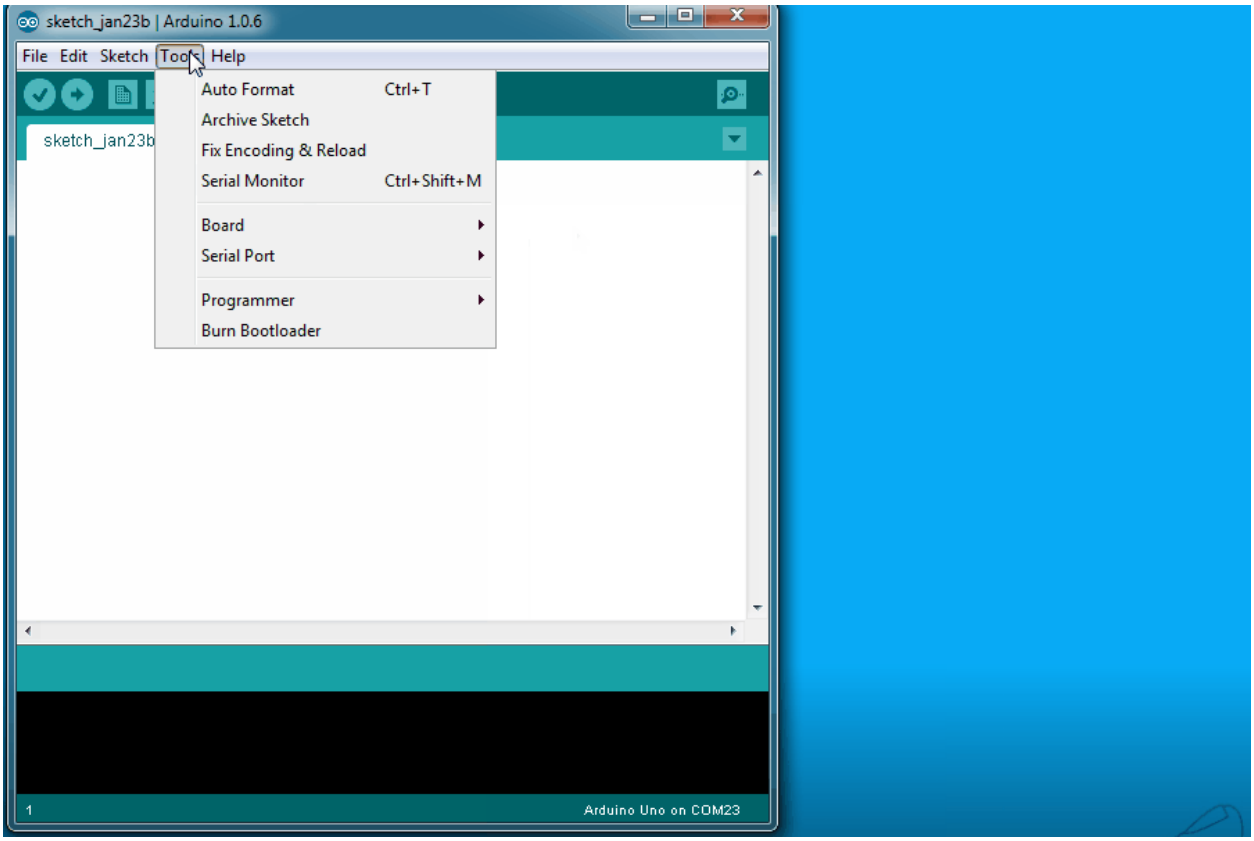
في الحقيقة أنا استفدت كثيراً من هذا الموقع كمعمل إلكترونيات رائع ، و كمبرمج للأردوينو. أتمنى أن تستمتع باستخدامه. شكراً **autodesk**

ملاحظة: لا يمكننا شرح استخدام البرنامج في هذا الكتاب ، لكننا سنشرحه في دورة فيديو على jeem2

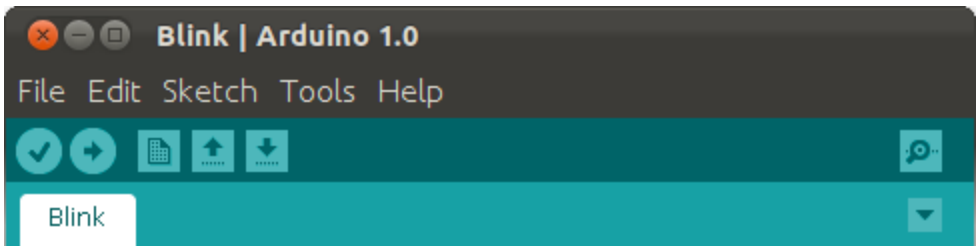
تعرف على برنامج Arduino IDE

سنشرح الطريقة الأولى لبرمجة الأردوينو (تثبيت برنامج Arduino IDE) على الكمبيوتر.
بعد تثبيت برنامج ArduinoIDE من موقع Arduino.cc افتح البرنامج …

تعتبر الواجهة سهلة الاستخدام و قليلة الخيارات. تأمل في الصورة وستلاحظ ذلك.

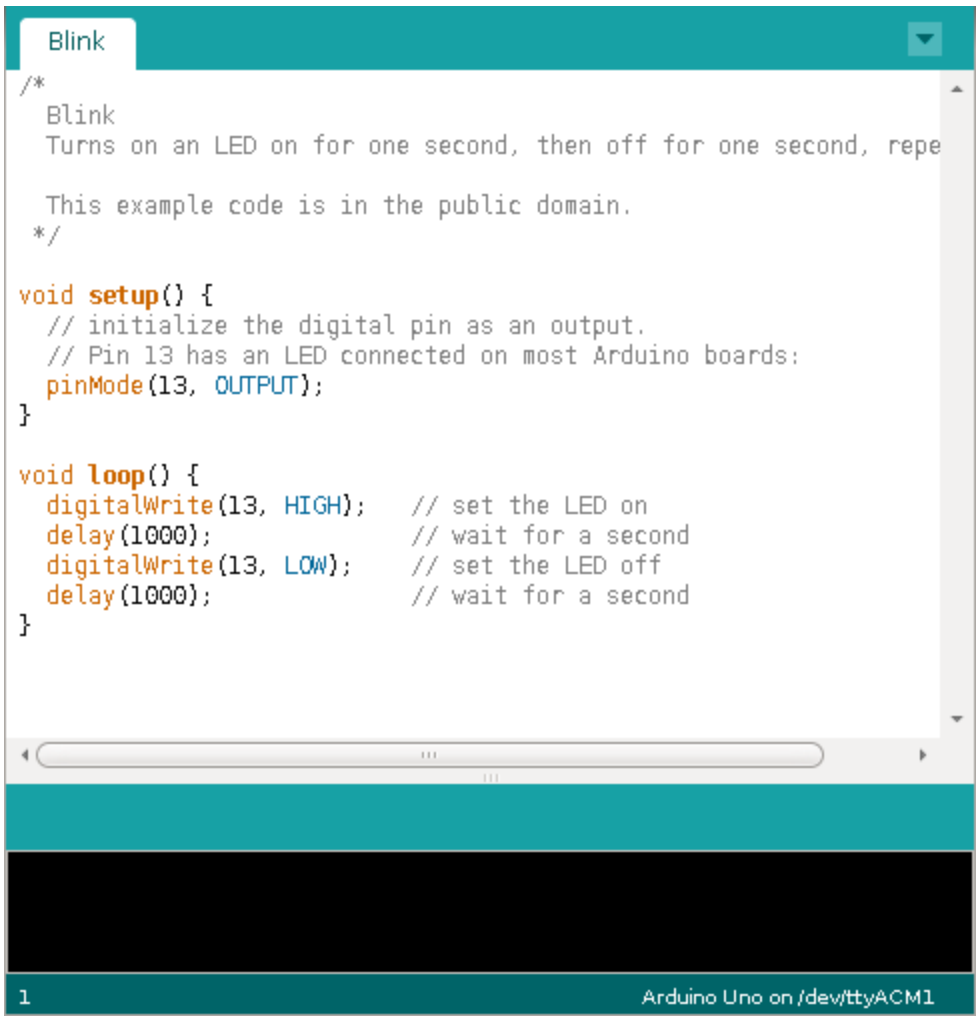


لن أشرح كل قائمة ، كل خيار كل زر هنا . استكشفها بنفسك . سأذكر أهم الأزرار هنا فقط.



File	فتح أمثلة كثيرة من الأكواد Examples (مفيد جداً) ضبط خيارات الخط
Edit	قص نسخ لصق ، البحث عن كلمة في الكود Find
Sketch	إضافة مكتبات الأوامر Library (سنشرح المكتبات قريباً)
Tools	اختيار موديل الأردوينو المطلوب برمجته Board اختيار المنفذ شيء يسمى Port . في حال وجود أكثر من أردوينو متصل للكمبيوتر
Help	مجموعة روابط مساعدة.
	فحص الكود في حال وجود خطأ في القواعد الكتابية سينبهك أسفل الصفحة (المربع الأسود)
	فحص الكود ثم رفعه إلى الأردوينو
	فتح صفحة جديدة فارغة
	فتح كود مخزن سابقاً
	حفظ الكود الحالي
	فتح شاشة السيريال (مهم جداً) Serial monitor

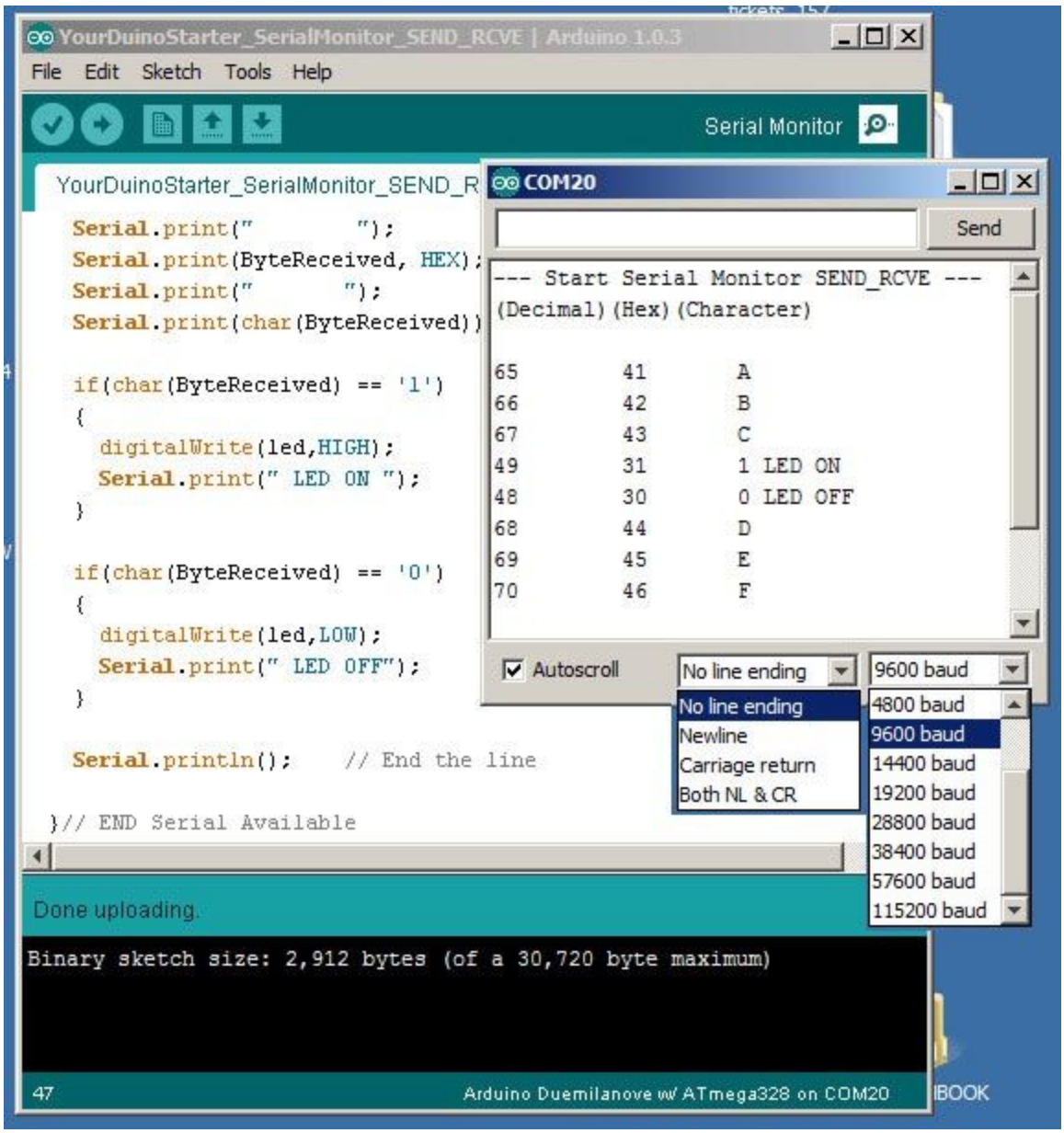
نصيحة: قبل البدء بكتابة الكود اذهب لـ **Tools** واختر الموديل (مثلا UNO)
و تأكد من ظهور رقم واختياره عند الـ port



في المساحة البيضاء ستكتب الكود

في الفراغ الأسود بالأسفل ستظهر لك بعض الملاحظات في حالة وجود خطأ في كتابة الكود.

تعرف على شاشة السيريال: Serial monitor (قبل البرمجة)



كما ذكرنا سابقاً ... الرمز  على اليمين يفتح شاشة السيريال. هذه طريقة تواصل بين الكمبيوتر والأردوينو. يمكنك استقبال أرقام و عبارات من الأردوينو . و يمكنك أيضاً إرسال أرقام وعبارات في المربع بجانب الزر : **Send**

لاحظ الخيار في الأسفل: (baud 9600) يحدد سرعة التراسل مع الأردوينو . معظم أنواع الأردوينو تتراسل بسرعة 9600

سوف نحل الكثير من الأمثلة في دورة الفيديو المرافقة للكتاب ونستخدم الشاشة كثيراً .

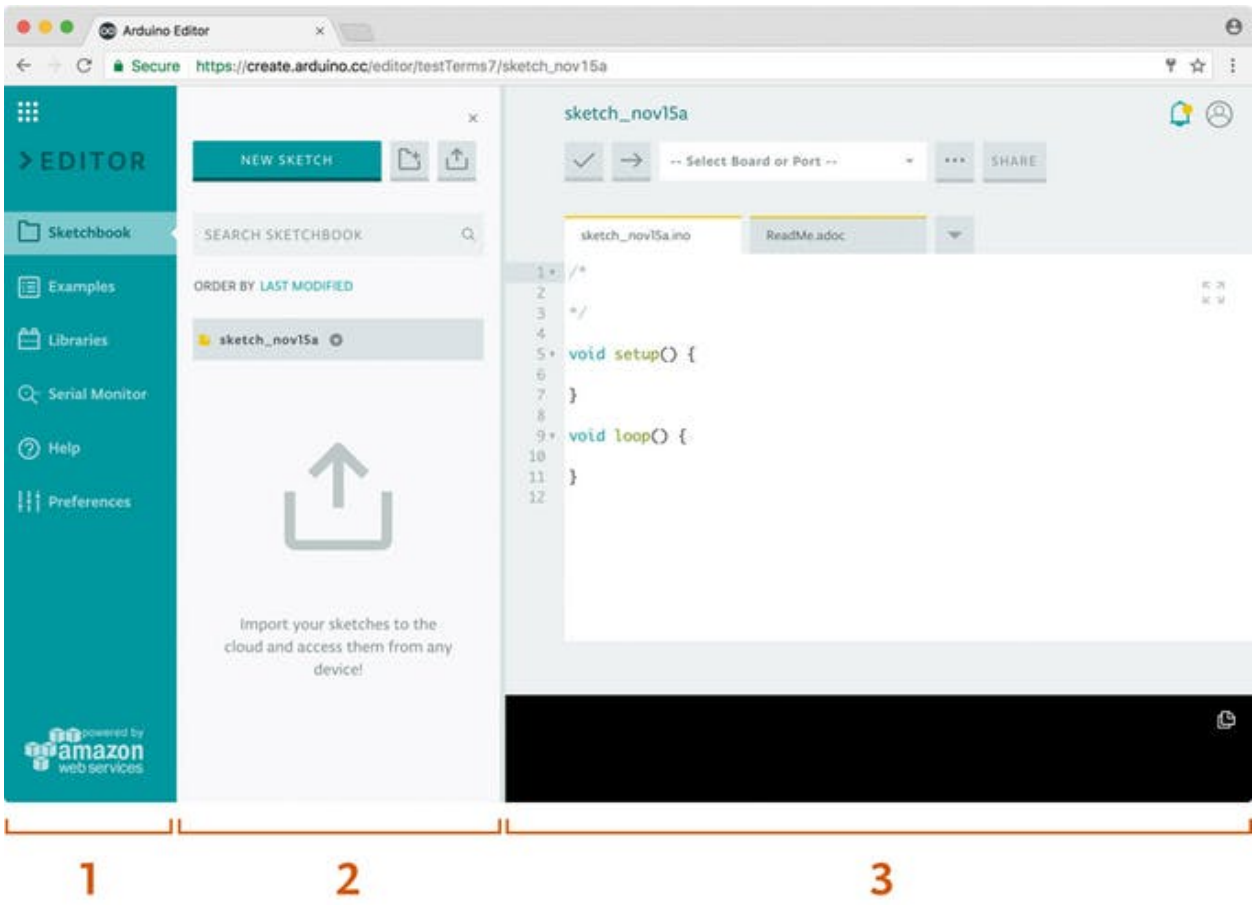
برمجة الأردوينو عبر الانترنت Arduino Create

اذا طلبت مني برمجة أردوينو اليوم فإني أفضل برمجتها باستخدام أداة البرمجة المتوفرة أونلاين **arduino-Create** الواجهة أكثر تنظيماً ، الوصول للمكتبات أسهل . وجميع الأكواد مخزنة مع حسابك و يمكنك الوصول لها من أي كمبيوتر متصل بالانترنت. أيضاً يمكنك تغيير الألوان لتكون غامقة و مريحة للعين أكثر.

فقط اذهب لموقع الأردوينو Arduino.cc وستجد رابط

Arduino >> software >> web editor

سيطلب منك انشاء حساب ، و تثبيت ملف صغير على الكمبيوتر. الواجهة سهلة . إذا كنت تعرف العمل على البرنامج العادي **Arduino IDE** فستحتاج لدقائق للاعتياد على الواجهة الجديدة.



القسم الأول : ستتنقل بين الأكواد التي كتبتها ، و الأكواد (الأمثلة) و إضافة المكتبات وفتح شاشة السيريال، و خيارات العرض و المساعدة.

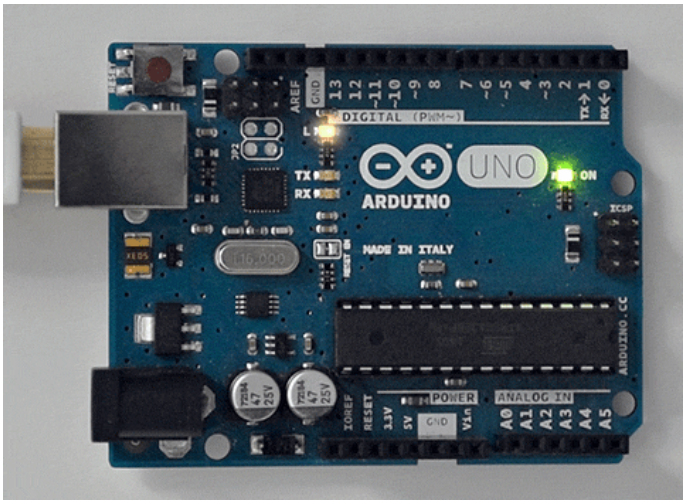
القسم الثاني : سترى فيه تفاصيل القسم الأول : جميع الأكواد ، جميع الأمثلة ، البحث عن المكتبات وإضافتها ، وهكذا.

القسم الثالث: هنا ستكتب الكود و بإمكانك إدراج ملف كتابي أو صورة خاصة بمشروعك 😀 رائع

Easiest code ever أسهل كود أردوينو

انظر مثلاً إلى هذه الأسطر : 👀 ربما تكون هذه الأسطر أسهل و أشهر برنامج (كود) للأردوينو إنه كود الوميض (**BLINK**) وإذا رفعته إلى الأردوينو ستجد أنه يقوم بالوميض مثل الصورة.

```
void setup(){
    pinMode(13,OUTPUT); }
void loop(){
    digitalWrite(13,HIGH);
    delay(500);
    digitalWrite(13,LOW);
    delay(1000);    }
```



دعنا نفكك الكود الماضي لنفهم كل أجزائه 👧

أولاً : لا يمكن أن يقبل الأردوينو أي كود بدون أن يحتوي على الجزئين التاليين:-

```
void  setup( ) {     }
void  loop  ( )   {     }
```

والعبارتين السابقتين `void setup` و `void loop` ضرورية و مفيدة بحيث يتم تقسيم الكود منطقياً إلى قسمين.

الجزء الأول يكون بين الأقواس { } بعد عبارة `setup`
وهذه الأوامر سوف يتم تنفيذها مرة واحدة فقط عند بداية التشغيل.

الجزء الثاني يكتب بين القوسين { } بعد العبارة `loop`
هذه الأوامر يتم تنفيذها بشكل متكرر طوال فترة تشغيل الأردوينو.

الآن لندخل قليلاً ونفهم الأوامر في قسم الـ `setup`

```
void setup(){
    pinMode(13,OUTPUT); }
```

كما قلنا سابقاً يحتوي الأردوينو أونو على 14 منفذ رقمي . هذه المنافذ يمكن أن تعمل كمداخل أو مخارج.
لذا يجب عليك تحديد عمل المنفذ قبل استخدامه، لاحظ الأمر `pinMode` يعمل على تهيئة المنفذ 13 ليصبح مخرج .

لاحظ وجود العلامة (;) بعد الأمر . يجب أن ينتهي كل أمر بهذه العلامة حتى يقبله الأردوينو

والآن سنفهم الأوامر في قسم الـ `loop`

```
void loop(){
    digitalWrite(13,HIGH);
    delay(500);
    digitalWrite(13,LOW);
    delay(1000);    }
```

الأمر `digitalWrite` يعمل على إخراج إشارة كهربائية على الطرف 13
`0v=LOW` أو `5v=HIGH`
وبما أن المخرج 13 مرتبط بـ ضوء في الأردوينو فسوف ترى هذا الضوء يعمل.
الأمر `delay` يعمل على تأخير زمني لمدة نصف ثانية
(500) ميللي ثانية = نصف ثانية
وذلك أننا لو شغلنا الضوء و أطفأناه بدون تأخير فلن نلاحظ الوميض بسبب سرعة الأردوينو
أعتقد أن باقي الكود واضح ولا يحتاج لشرح . سوف ينطفئ الضوء لمدة ثانية واحدة ثم يتكرر الوميض.

لمشاهدة الكود في المختبر https://circuits.io/circuits/5246043-jeem2_book_ex1

مبروك !! لقد فهمت الكود الأول وبإمكانك تجربته على الأردوينو و التلاعب بزمن التأخير للتحكم بالسرعة جرب مثلا (300) بدل (1000) ولاحظ الفرق. افرح فأنت الآن تبرمج ...

رحلة البرمجة ... التخاطب مع الكمبيوتر

كم تحتاج من الوقت لتصير مبرمجاً جيداً ؟ ...

ليس أياماً ولا أسابيع. البرمجة عالم واسع وعلم متجدد. البرمجة مهارة العصر الحديث؛ عصر المعلومات و الاتصالات و الانترنت. البرمجة هي التخاطب المباشر مع الآلة. مع الإلكترونيات و الكهرباء. البرمجة تحتاج أشهر أو سنوات لتتعلمها وتتقنها.

لا غرابة أن أعلى الرواتب في كثير من الشركات العظمى تكون للمبرمجين 😐 ... هل أخفتك؟

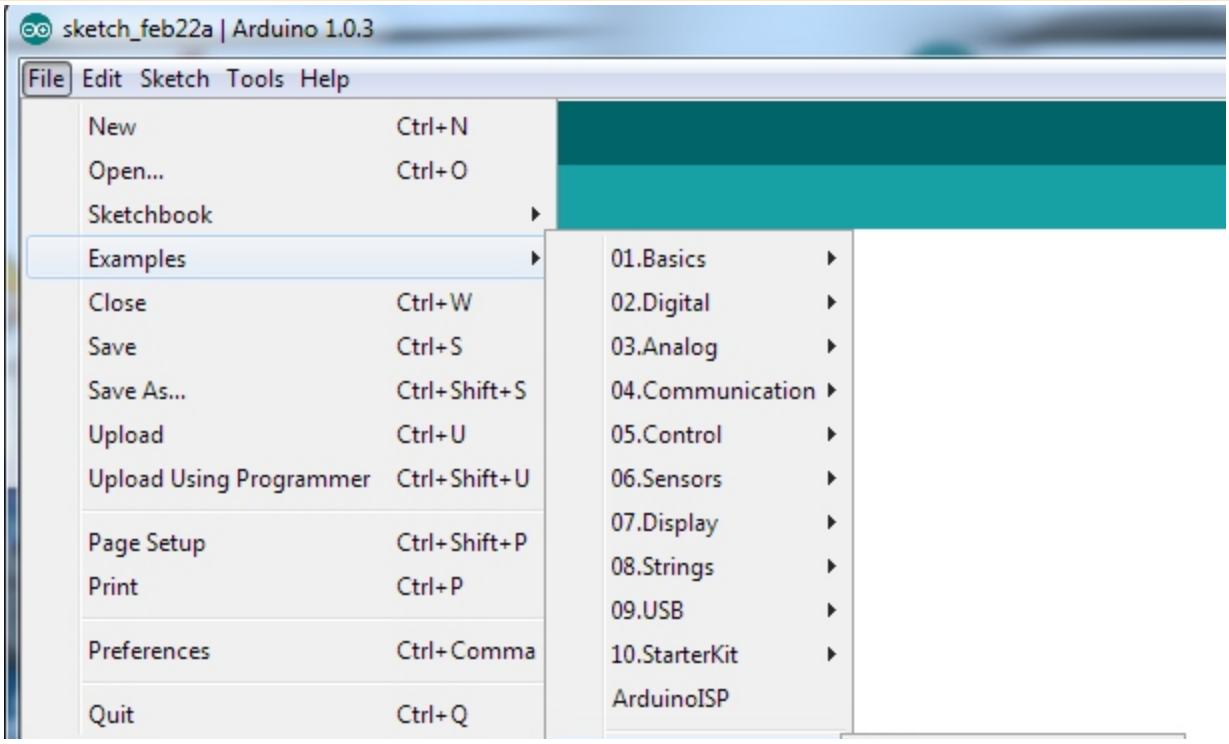
لا يمكن في هذا الكتاب ولا في أي كتاب شرح جميع الأكواد. الأكواد لا حصر لها 😐

الكتاب مصمم ليواكب دورة مرئية (فيديو) على موقع **jeem2** وعلى اليوتيوب.

في دورة الفيديو سوف نبرمج العديد من الحالات المختلفة . سنبرمج أكثر مما سنكتب هنا.

أنصحك بمتابعتنا هناك. بالفيديو يمكننا شرح الأكواد بشكل أفضل من الكتابة هنا.

الاستفادة من الأمثلة (الأكواد) المخزنة



الآن لن نبني برنامج من جديد ، لكننا سنتعلم من مثال موجود و مخزن مع البرنامج. اتبع الخطوات لفتح الكود:

File >> Examples >> basics >> Blink

الأخطاء في كتابة الكود Errors , bugs

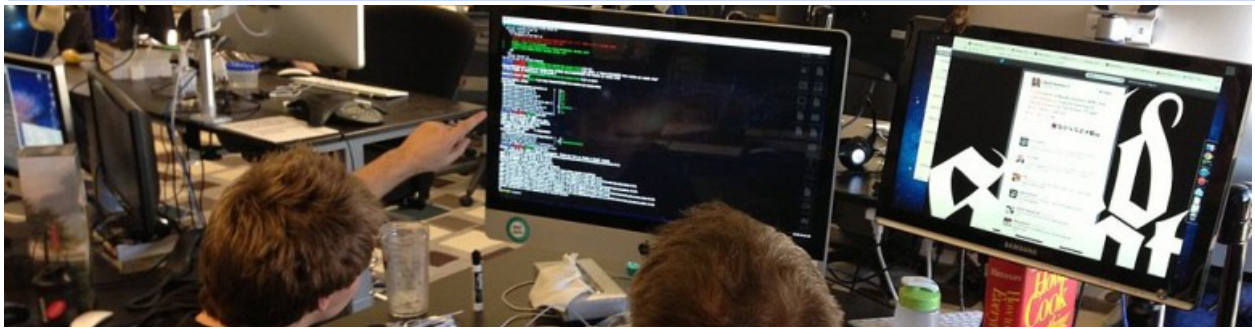
يتكون الكود من مجموعة من الأوامر تسير عبر منطق معين لتنفيذ هدف معين. (مثل عمل إشارة المرور)
ويكون الكود مكتوب بلغة من لغات البرمجة (في حالة الأردوينو اللغة هي C و C++)

أنواع الأخطاء عند كتابة الكود :

1- أخطاء في قواعد كتابة الكود Syntax bugs وهذه أخطاء يسهل إيجادها بواسطة البرنامج IDE
مثل: نسيان كتابة (;) بعد أحد الأوامر ، أو استخدام متغير لم يتم تعريفه مسبقاً .

2- أخطاء منطقية : Logic bugs وهي أخطاء تجعل الكود لا ينفذ العمل المطلوب ، مع أن قواعد الكتابة صحيحة.

أخلاقيات البرمجة Coding ethics



يتفق معظم المبرمجين حول العالم على بعض العادات التي تجعل الكود أسهل للقراءة و الفهم . فالهدف ليس أن يعمل الكود وحسب ، بل أن يكون واضحاً لأي مبرمج آخر أن يراه و يفهمه ثم يعدل عليه. ومن هذه العادات الأخلاقية عند البرمجة:

```
void setup(){
    pinMode(13,OUTPUT); }
void loop(){
    digitalWrite(13,HIGH);
    delay(1000);
    digitalWrite(13,LOW);
    delay(1000);    }
```

1- كتابة الملاحظات Comments :-

يعمل المبرمجين على جعل الأكواد مفهومة للمبرمجين الآخرين ، لذا فهم يكتبون ملاحظات لتشرح الكود كاملاً . أو لشرح بعض الأوامر. هذه الملاحظات لا تؤثر على طريقة عمل الأردوينو أبداً . توجد طريقتين لكتابة الملاحظات

- **لعمل ملاحظات من عدة أسطر اكتب :** /* هنا اكتب أي ملاحظات تحب */
- **لعمل ملاحظات من سطر واحد اكتب :** // هنا اكتب أي ملاحظات

2- تعريف المتغيرات الهامة أعلى الكود :-

الكود الماضي قصير جداً لكن الحال لن يكون هكذا دائماً . ينصح المبرمجين بتعريف متغير في أعلى الكود يحتوي القيم التي يمكن تغييرها للتحكم بطريقة التشغيل. هذا يسهـّـل التعامل مع الكود في المستقبل من أي شخص.
في الكود الماضي ، الطرف الذي يومض هو 13 ، و التأخير هو ثانية واحدة .

3-ترك مسافات بداية الأسطر indenting :-

من المهم أن يكون الكود سهل القراءة والتتبع. لذا فإن المبرمجين يتركون مسافات بدايات الأسطر تجعل من السهل معرفة أقسام الكود. ستظهر فائدة استخدام المسافات أكثر عندما نتعلم الأوامر if و for

---

4-تسمية المتغيرات بطريقة camelCase :-

في كثير من الأحيان سنحتاج لتعريف متغيرات في الكود. وبإمكانك أن تسمي المتغير أي إسم (مثلا x أو z3) لكن إذا كان الكود كبيرا ينصح بتسمية المتغيرات أسماء معبرة عن عملها. و إذا كانت أكثر من كلمة استخدم حرف كبير بداية الكلمة الثانية و الثالثة لأنه لا يمكنك ترك مسافة في اسم المتغير. مثلا

```
int    ledPin=10;     int    delayTime=750;
```

هل لاحظت كتابة الأوامر بهذه الطريقة ؟

```
digitalRead( )   ,   pinMode( )
```

لكن في بعض العبارات لا تنطبق هذه القاعدة و تكون جميع الحروف كبيرة !

الطريقة التي ينصح بها هي:-

```
/*
هذا الكود يعمل على تشغيل وميض على منفذ رقمي
 */
int  LED=13;  //اختر المنفذ الذي سيرتبط به الضوء
int  D=1000;  //اكتب التأخير الزمني بالمللي ثانية
void setup(){
    pinMode(LED,OUTPUT); }
void loop(){
    digitalWrite(LED,HIGH);
    delay(D);
    digitalWrite(LED,LOW);
    delay(D);   }
```

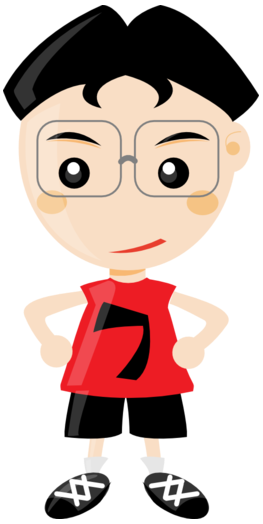
لاحظ كتابة الملاحظات أعلى الكود و أمام بعض الأوامر (عادة يتغير لونها آلياً في البرنامج)

لاحظ أيضا تعريف المتغيرين **LED** و **D** أعلى الكود لجعل التحكم بالكود أسهل و أوضح.

لاحظ أيضاً في الكود تم استبدال رقم **13** باسم المتغير **LED** في باقي الكود

و استبدال القيمة **1000** باسم المتغير **D**

لاحظ أن بعض الأسطر تبدأ من البداية ، و بعضها تترك مسافة (فارغة) قبل كتابة الأمر ، هذا يسمى indenting وهي تساعد في تسهيل قراءة الكود و تقسيمه إلى أجزاء عند النظر إليه.



هــل تشعر أنك فهمت الكود السابق بشكل جيد (كود الوميض) ؟

سنتعلم الكثير من الأوامر و المهارات لبرمجة الأردوينو في الصفحات التالية. كن مستعداً ... 💪

Variables Types أنواع المتغيرات

في الأكواد السابقة ... لاحظ وجود (int) قبل تعريف كل متغير . للمتغيرات أنواع كثيرة أشهرها int وهو لتعريف متغير (variable) يحمل رقم صحيح (بدون فاصلة عشرية) ويمكن أن تكون قيمته 32767 بالموجب أو بالسالب. في معظم الأكواد ستستخدم متغيرات من نوع int وسنشرح الأنواع الأخرى مع الأمثلة القادمة. في الجدول التالي أشهر أنواع المتغيرات التي قد تحتاج لها

نوع المتغير	الاستخدام	عدد البايتات
int	متغير يكون فيه قيمة (رقم) تكون -32700 إلى +32700 (تقريباً)	2
float	متغير يقبل قيمة عددية كبيرة ويقبل الفاصلة العشرية (+/-) `float z=10.12;`	4
Byte	متغير يقبل قيمة عددية صغيرة (0-255) فقط `Byte x=255;`	1
bool	متغير يحمل حالة من حالتين فقط (0 أو 1) (LOW , HIGH) `bool x=LOW;`	1
long	متغير يحمل رقم كبير (-2B إلى 2B) تقريباً `long x=1000000;`	4
char	متغير يحمل رقم ويعبر عن حرف بترميز ASCII `char x='m';`	1
String	مصفوفة من الحروف (كلمة أو جملة) `String x="hello World ... " ;`	any

ملاحظة : إذا كنت قد درست البرمجة في الحاسب سابقاً و ربما لغة C يوجد بعض الاختلافات في أحجام المتغيرات بين الأردوينو و الكمبيوتر العادي. مثلا int في الكمبيوتر تستهلك Bytes-4 وليس 2 فقط

لمشاهدة جميع أنواع المتغيرات اذهب للصفحة : هـــــنا قسم Data types

const int في بعض الأكواد ستلاحظ كتابة عبارة (const) قبل نوع المتغير ... لا تقلق كل المقصود أن قيمة المتغير هذا ستبقى ثابتة طوال عمل الكود و لن تتغير و في حالة أنك كتبت الكود بدونها فلن تؤثر على طريقة عمل البرنامج.

unsigned long معظم المتغيرات السابقة (مثل int أو long) تقبل قيم موجبة أو سالبة (مثلاً 500 أو -2050 يمكن أن تكون قيم لمتغير int) و لكن في بعض التطبيقات نكون متأكدين أن القيمة لن تكون سالبة بأي حال. كما نريد الإستفادة القصوى من مساحة الذاكرة . فنستخدم عبارة unsigned قبل نوع المتغير

مثال: المتغير long يمكنه تخزين رقم يصل إلى 2 بليون تقريباً

بينما المتغير unsigned long يمكنه تخزين رقم يصل إلى 4 بليون تقريباً

في بعض التطبيقات (مثل حساب الزمن بالمايكرو ثانية) نحتاج إلى تخزين قيم كبيرة جداً و يستحسن استخدام unsigned long في هذه الحالة.

static int ليس من الشائع مشاهدة هذا النوع (static) لكنه مفيد في حالات نادرة . مثل تعريف المتغير داخل حلقة بدون أن يعيد تعيين قيمته في كل دورة .

ترميز ASCII

نحتاج كثيرا جداً لاستخدام الحروف والكلمات (وليس الأرقام فقط) تم الإتفاق عالميا على نظام ترميز يستخدم بايت واحد (8 بت) للتعبير عن جميع الرموز المستخدمة في الكتابة (باللغة الانجليزية)
فعندما تخزن المتغير بالطريقة التالية

```
char  x='w';
```

في الحقيقة أنت تضع فيه القيمة 119 فقط !!

في بعض التطبيقات تتسبب بمشكلة فالرقم 5 بنظام ASCII يتم تخزينه : 53 !!!

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

لا تحاول حفظ الجدول !! الجدول لتوضيح الفكرة فقط 😂

المتغيرات العامة والمتغيرات المحليّة Global vs Local variables

في الأكواد السابقة قمنا بتعريف (Declare & assign) المتغيرات أعلى الكود -قبل void setup هذه المتغيرات تسمى متغيرات عامة (Global Variables) و يمكن استخدامها في أي مكان من الكود و ستحتفظ بقيمتها. بينما يوجد خيار ثاني وهو تعريف المتغيرات داخل الدوال . وهذا يجعلها قابلة للاستخدام في الدالة فقط. توجد فوائد كبيرة لهذه الطريقة ولكن يجب عليك أن تفهمها بشكل جيد.

في بعض الحالات (عادة داخل القسم void loop) نقوم بتعريف متغير محلـّـي local variable ليعمل كعداد مثلاً :

```
void loop(){
  int x=0;
  Serial.print( x );
   x++; }
```

قد تتوقع أن الكود سيطبع على الشاشة العد 0,1,2,3,..... لكن في الواقع سيظهر 0,0,0,0,.....
السبب أنه في كل مرة يتكرر تنفيذ الحلقة سيعاد إنشاء المتغير وإعطائه القيمة 0 !
أقصر حل لهذه المشكلة هو جعل المتغير من نوع **static int** هذا سيجعل قيمة المتغير تتحدد في أول مرة بـ 0 و في باقي الدورات يحافظ المتغير على قيمته ولا يعود للقيمة 0 .

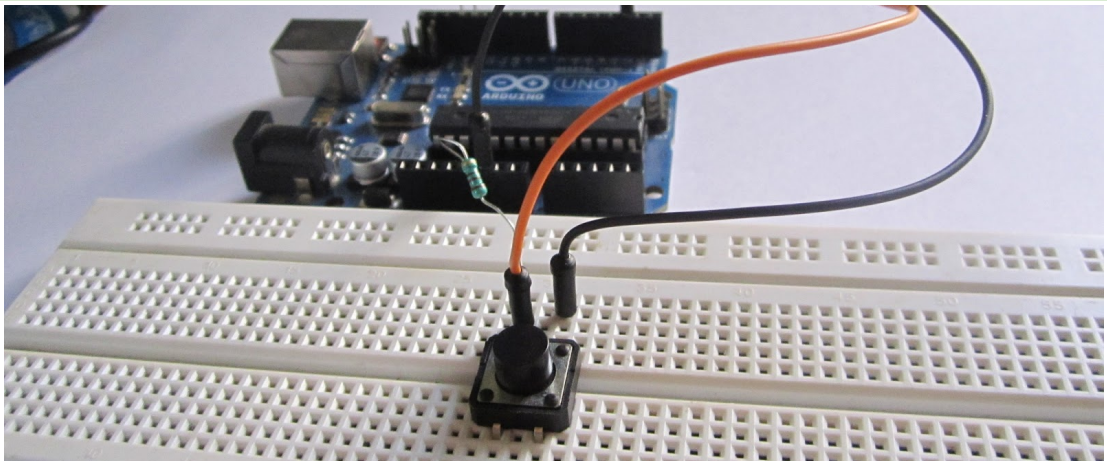
في حالة وجود متغير عام بإسم x و متغير محلي بإسم x فإنه داخل الدالة سيتم استخدام المتغير المحلي ، و خارجها سيستخدم المتغير العام. هذه الطريقة مفيدة كثيرا عند استخدام دوال functions.

```
int  x=10;
void setup(){ … }
void loop(){ … }

void  fun() {
  int x=5; }
```

داخل الدالة : fun تكون قيمة **x=5** بينما خارجها ، تكون قيمة **x=10**
أتمنى أن الفكرة اتضحت

المداخل الرقمية digital inputs والأمر الشرطي if



الأزرار buttons هي أبسط طريقة لإدخال إشارات رقمية إلى الأردوينو.

تكون الإشارة الكهربائية LOW أو HIGH (صفر أو 5v) وعادة نعبر عنها بـ 0,1

من المهم جدا استخدامها للتحكم بعمل الأردوينو أثناء تشغيله . لذا نستخدم الأمر **digitalRead**

الأمر if مهم جدا وهو يعمل على تنفيذ مجموعة أوامر فقط إذا تحقق الشرط بين الأقواس( )

الشرط	>	<	==	=!	>=	<=
المعنى	أكبر من	أصغر من	يساوي	لا يساوي	أكبر من أو يساوي	أصغر من أو يساوي

لاحظ: يمكن دمج شرطين أو أكثر في شرط واحد بعلاقة (آند AND أو أور OR) أتمنى أنك تعرف هذين الأمرين. مثلا أنت تريد تشغيل الضوء بشرط أن يكون x=10 و y>100

```
if(x==10   &&   y>100) {digitalWrite(13,1);}
```

وبإمكانك تنفيذ أمر إذا تحقق أحد شرطين الشرط الأول أو الشرط الثاني

مثلا : تريد تشغيل الضوء إذا كان x<10 أو y<10

```
if(x>10   ||   y>10) {digitalWrite(13,1);}
```

أيضا بإمكانك إتباع الأمر if بالأمر else if أو else حتى يتم تنفيذ أحد هذه الشروط وليس جميعها . شاهد المثال:

```
if (x==0) { … }              //التنفيذ فقط إذا المتغير يساوي 0
if (x>5 && y<x) { … }        //يجب أن يتحقق الشرطين للتنفيذ
else if(x>=25||x<= -25){ … } //التنفيذ حال تحقق أي من الشرطين
else { … }
```

مثال ::- قد يكون الشرح بالمثال أفضل و أسرع لذا شاهد المثال التالي ثم سنشرحه:

```
/* Blink , but when switch pressed = fast blink */
int  LED=13;       //حدد رقم المنفذ الذي سيتصل به الضوء
int  SW=0;         //المنفذ الذي سيتصل بالمفتاح_الزر
int  D1=700;       //التأخير الطويل
int  D2=250;       //التأخير القصير

void setup(){
  pinMode(LED,OUTPUT);
  pinMode(SW,INPUT_PULLUP);}//تهيئة الطرف كمدخل مع استخدام مقاومة رفع
void loop(){
  int x=digitalRead(SW);
  if(x==0){ int D =D2; }    //الزر سيرسل -صفر- عند الضغط عليه
  else{int D=D1; }          //أنشأنا متغير جديد لتسهيل الكود
  digitalWrite(LED,HIGH);
  delay(D);
  digitalWrite(LED,LOW);
  delay(D); }
```

يشبه برنامج الوميض السابق، سوى وجود مدخل رقمي . في حال الضغط عليه تزيد سرعة الوميض.

سنتحدث عن بعض الملاحظات في الكود الماضي:-

-- المتغيرات التي يمكن تغييرها أعلى الكود ، هذا يجعل من السهل عليك التعديل على الكود مستقبلاً.

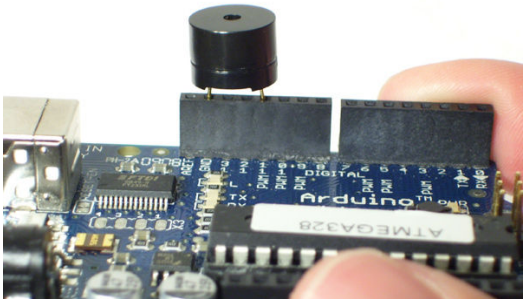
--داخل (void setup) تحديد عمل المنفذ 13 (تم تسميته LED) ليعمل كمخرج . و المنفذ 0 (تم تسميته SW) ليكون مدخل. الكلمة (PULLUP) تربط الدخل بمقاومة داخلية وهذا مفيد ومشروح في الكتاب (مقاومة الرفع والخفض ص100)

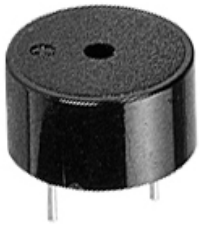
--داخل (void loop) لاحظ يمكن تعريف المتغيرات هنا . لا يلزم تعريفها في أعلى الكود. **int x**
الفكرة باختصار هي وضع قيمة D1 أو D2 في المتغير الجديد D حسب حالة المفتاح.
ثم تنفيذ الوميض بقيمة التأخير D .... أتمنى أنك فهمت الفكرة ☹

من وقت لآخر سأضع لك روابط للاستزادة (ليس ضرورياً مشاهدتها)
شرح للأمر digitalRead اضغط هنا شرح لأمر (if) اضغط هنا

لمشاهدة هذا الكود في المختبر الافتراضي:: https://circuits.io/circuits/5246020-jeem_book_ex2

إصدار صوت باستخدام الأمر tone





إصدار صوت من الدائرة الإلكترونية من الطرق السهلة و الفعالة جداً للتنبيه. لاحظ أن الأردوينو غير مصمم لإصدار أصوات معقدة مثل الموسيقى أو صوت كلام بشري. لكن بإمكانه إصدار نغمات مختلفة باستخدام سماعة بسيطة مثل الصور السابقة.

تنقسم السماعات البسيطة إلى نوعين عموماً :

- سماعات خاملة تحتاج لإشارة كهربائية مترددة حتى تصدر صوت
- سماعات نشطة : تحتاج لجهد مستمر DC فقط لتصدر صوت

معظم السماعات المستخدمة (والأفضل) هي السماعات الخاملة والتي يمكن التحكم بنغمة الصوت بتغيير التردد.

- لاحظ أن السماعة لها قطبية (+ و -) وصل السالب إلى الأرضي و الموجب إلى أي منفذ رقمي.
- معظم السماعات البسيطة (buzzer) تتصل مباشرة بمنفذ الأردوينو سوى أن بعضها تتطلب وجود مقاومة (يستحسن قراءة التعليمات للسماعة إذا وجدت)

```
tone(2,300);   //يعمل على إصدار صوت من المنفذ2 بتردد 300 هيرتز
noTone(2);     //يعمل على إيقاف النغمة على المنفذ 2
```

```
tone(2,300,500);      // يمكنك إضافة رقم ثالث يكون مدة النغمة بالمللي ثانية
```

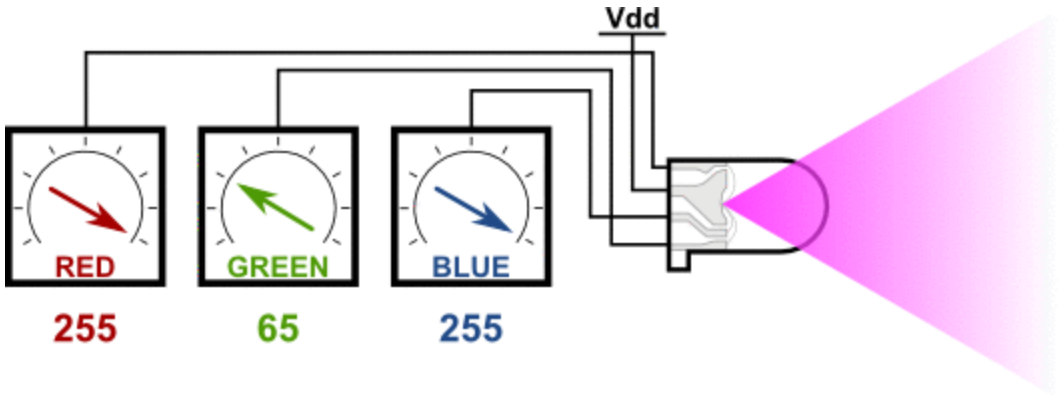
جرب تشغيل أوامر الصوت في هذا الرابط: هـــنـــا

للاستزادة : شرح للأمر tone

تمرين: صمم أورغ بسيط جداً بـ 3 أزرار أو 6 _ بحيث كلما ضغطنا على زر يصدر نغمة مختلفة.

C=261 , D=293 , E=329, F=349 , G=392

المداخل والمخارج التماثلية Analog inputs/outputs

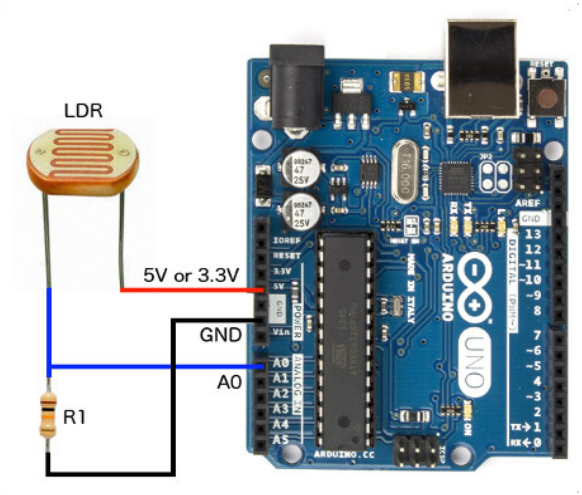


هل تعرف الفرق بين الإشارة الكهربائية الرقمية و الإشارة الكهربائية التماثلية؟أتمنى أنك تعرف
إذا لم تكن تعرف ، ابحث في الفهرس عن القسم الذي يشرح الفكرة باختصار في هذا الكتاب (ص16)
معظم الحساسات في الواقع ترسل إشارات تماثلية (قياس الضوء ، الحرارة ، الصوت) في هذا المثال سوف نقيس قيمة جهد تماثلي باستخدام مقاومة متغيرة . و يكون الخرج سماعة بسيطة . سوف يتغير الصوت (التردد) حسب تغيير الجهد الداخل

```
/*  (تردد الصوت) عند تحريك ذراع المقاومة سيتغير الصوت
int  spkr=3;       //المنفذ الذي سيتصل بالسماعة

void setup(){
  pinMode(spkr,OUTPUT);  }
void loop(){
  int IN=analogRead(A0);   //هنا سنقرأ قيمة الجهد التماثلي عند المدخل
  tone(spkr,IN+31); //هذا الأمر سيصدر إشارة صوت على المنفذ3
  delay(100);
  noTone(spkr);  }
```

لمشاهدة تشغيل الكود في المختبر الافتراضي : https://circuits.io/circuits/5246595-jeem2_book_ex3

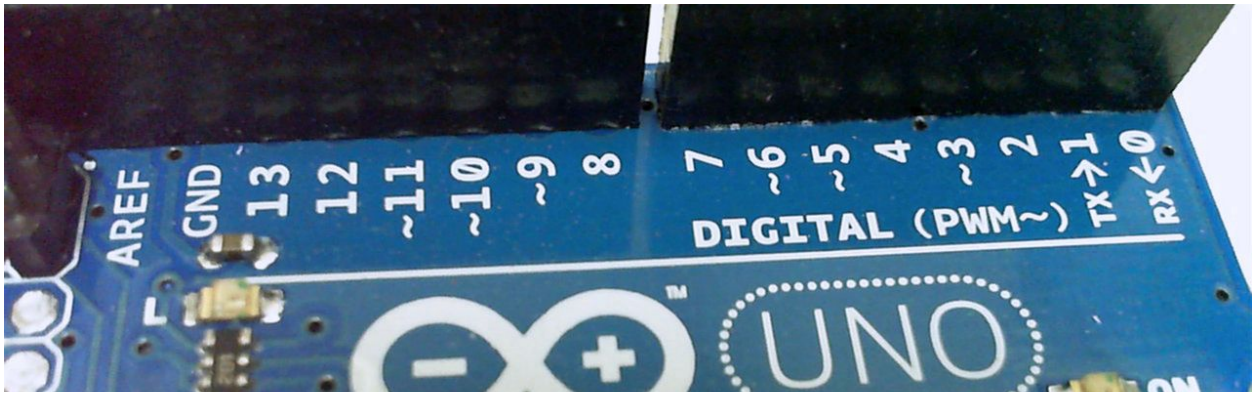


لاحظ الأمر **analogRead** يدخل قيمة من المقاومة المتغيرة **(0-1023)** . وضعنا القيمة في متغير اسمه (**IN**)
ثم استخدمنا الأمر **tone** لإخراج الإشارة إلى السماعة بحيث تكون قيمة (**IN**) تردد الصوت.
يستمر الصوت لمدة 0.1 ثانية ثم تنطفئ النغمة وتتكرر العملية في الحلقة **loop**

مثال لتوصيل مقاومتين ضوئية و ثابتة إلى مدخل تماثلي Analog بحيث يمكن معرفة (قراءة) شدة الضوء في المكان .

إخراج قيمة جهد تماثلية PWM باستخدام الأمر analogWrite

إذا كان الجهد الرقمي يكون 0v أو 5v فقط ، فإن الجهد التماثلي يمكن أن يكون أي قيمة بينهما. مثلا 3.5v أو 1.25v
إذا دققت النظر بجانب المنافذ الرقمية ستجد بعضها يظهر بجانبه العلامة (~) وهي تدل أن هذا المنفذ يمكن استخدامه لإصدار جهد تماثلي (بطريقةpwm) وهي في الأردوينو أونو (3,5,6,9,10,11)



استخدم الأمر بالطريقة التالية:

```
analogWrite(10,125);   //(pin, value 0-255)
```

لاحظ أن القيمة التي يستقبلها الأمر تكون 0-255 لذا ففي المثال السابق سنجد أن الجهد الصادر من المنفذ حوالي 2.5v

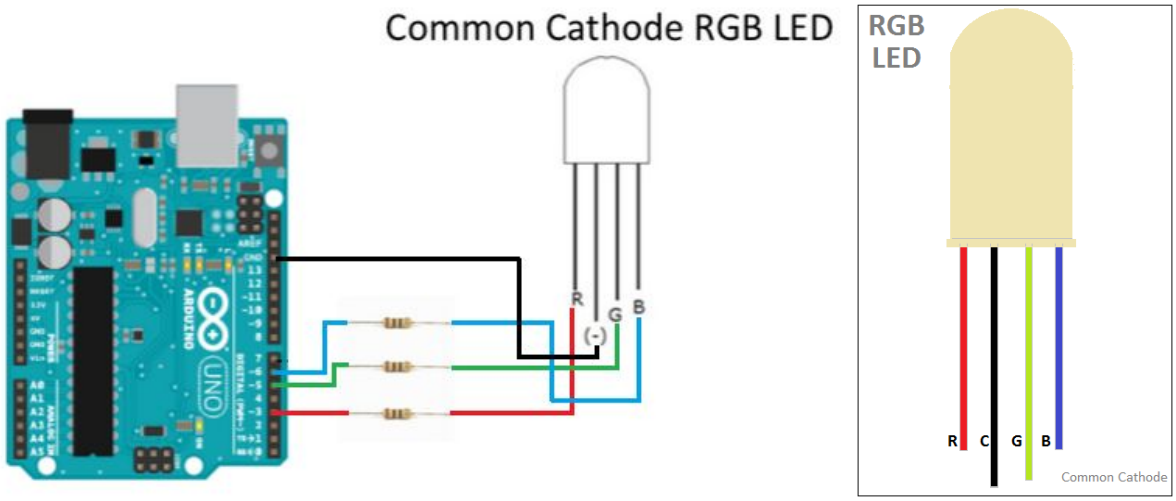
مثال لكود بسيط يجعل ضوء (LED) يضيء بشكل تدريجي ثم ينطفئ فجأة.

```
void setup(){
  pinMode(10,OUTPUT);}
void loop(){
  for(int i=0;i<=255;i++){
    analogWrite(10,i);
    delay(30);}}
```

شاهد تشغيل هذا الكود على المعمل الافتراضي (اضغط هنا)

للاستزادة : شرح للأمر analogRead شرح للأمر analogWrite

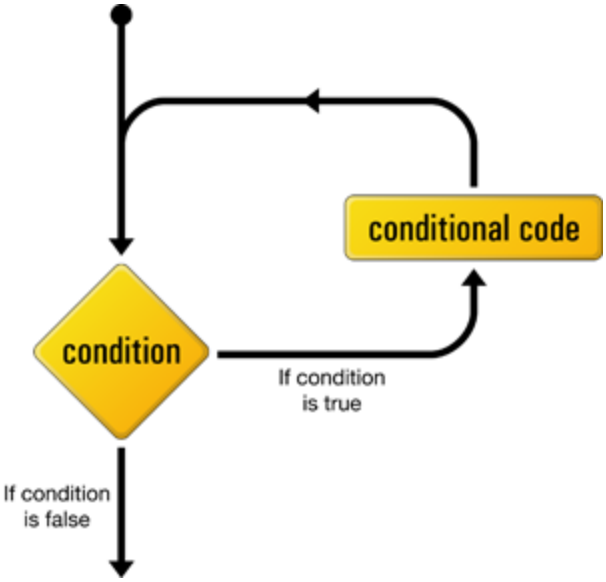
تطبيق خفيف دم 😊 : استخدم الإشارات التماثلية للتحكم بلون LED متعدد الألوان . RGB LED




```
int G=5 , B=6 , R=3;
void setup(){
  pinMode(9,OUTPUT);
  pinMode(10,OUTPUT);
  pinMode(11,OUTPUT);  }
void loop(){
  int GV=random(0,255);
  int BV=random(0,255);
  int RV=random(0,255);
  analogWrite(G,GV);
  analogWrite(B,BV);
  analogWrite(R,RV);
  delay(1000); }
```

شاهد تشغيل هذا التطبيق : اضغط هنا

حلقات while و for



الأمر while يعمل على تكرار جزء من الكود بين الأقواس { } مادام الشرط متحقق.

```
while(x<100) { … }
```

وفي حال تغير قيمة x حتى تصير أكبر من 100 فسيخرج من الحلقة

*نادراً تتم كتابة الأمر بهذه الطريقة:

```
do{ … } while( a<10);
```

الأمر for يشبه الأمر السابق سوى أنه يستخدم عندما تكون تعرف عدد الدورات التي تريد تنفيذها

```
for(int i=10 ; i>0 ; i--)
{   …   }
```

وسوف ينفذ الأوامر بين الأقواس { } عشر مرات و تكون قيمة المتغير (i) هي العداد الذي يعد من 10 إلى 0

مثال :نود جعل جميع المنافذ مخارج ثم نود تشغيل 5 نغمات صوتية.

```
void setup(){
  for(int i=0;i<14;i++){pinMode(i,OUTPUT);}}
  int z=1;
  while(z<6){tone(10,300,200); delay(500);z++;}
```

للاستزادة : شرح while شرح for

تمرين: صمم كود يعمل على إظهار 10 ومضات سريعة ثم 3 بطيئة باستخدام الأمر for
تمرين: قراءة مقاومة متغيرة إذا كانت القراءة أعلى من 200 يصدر صوت، إذا كانت أقل يعمل وميض عادي _ استخدم الأمر while

الذهاب إلى وَســم goto label

هذه الطريقة نادرة الاستخدام لكنها تستحق الذكر في الكتاب.
يسير الأردوينو على تنفيذ الأوامر بالتوالي حسب كتابة الكود. لكن الأمر goto ينقل تنفيذ الأمر مباشرة إلى أي مكان آخر في الكود ؛ شرط أن يتم تسمية ذلك المكان ثم وضع (:) بعد الإسم.
مثال:

```
void setup(){
  pinMode(3,OUTPUT);
  goto label1;
label2: tone(3,1000);
  delay(500);  }
void loop(){
label1:  int  x=analogRead(A0);
  if(x>800) { goto label2;}
  tone(3,300);
  delay(500);
  tone(3,400);
  delay(500);  }
```

عمل الكود هو : إصدار صوت متتابع من ترددين (300،400) مع مراقبة دخل تماثلي ، في حالة زيادة القراءة عن (800) فإن التنفيذ سوف يعود إلى الوسم label1 ويصدر صوت بتردد عالي (1000Hz)

للاستزادة من أمر goto اتبع الرابط

تمرين:: يشبه فكرة سندرسها مستقبلاً هي الضبط المبدئي (calibration) لكن أبسط.
المطلوب : كود يعمل وميض و كلما ضغطنا على زر رقمي يتوقف الوميض ويصدر صوت لمدة 5 ثواني. ثم يعود للوميض _ استخدم الأمر goto

شاشة السيريال Serial Monitor

تحدثنا سابقاً عن شاشة السيريال كفكرة لكننا لم نتحدث عن الأوامر التي ستحتاجها للتعامل مع شاشة السيريال. باختصار هي أداة على الكمبيوتر (توجد في برنامج Arduino IDE أو Arduino Create) تعمل على التواصل مع الأردوينو أثناء تشغيله و يمكنك استخدامها لعرض معلومات من الأردوينو إلى شاشة الكمبيوتر (مثلا قيم المتغيرات) أو إرسال أرقام من الكمبيوتر إلى الأردوينو (أثناء تشغيله) هي مفيدة جدا جدا بالذات في مرحلة التشغيل التجريبي للكود و تساعدك كثيرا على تتبع قيم المتغيرات أثناء عمل البرنامج.

ملاحظة: عند استخدام شاشة السيريال فإن الأردوينو يستخدم المنفذين الرقميين 0،1 للتواصل مع الحاسب، لذا لا يمكنك استخدامهما كمنافذ في هذه الحالة.

الأوامر الأساسية لإظهار كتابة أو قيمة على شاشة السيريال هي:

`Serial.begin(9600);`	**قبل استخدام شاشة السيريال يجب إعطاءها أمر البدء العدد 9600 هي سرعة التراسل ولا تقلق نفسك بالأمر كل بوردات الأردوينو تعمل بهذه السرعة _ماعدا قليل تجد تفصيلها في الموقع**
`Serial.print("hey");` `Serial.print("\n\t");`	أمر print لإظهار عبارة من الأردوينو إلى شاشة الكمبيوتر n\ لبدء سطر جديد و t\ لترك مسافة tab
`Serial.println(x);`	**عند اضافة الحرفين (ln) بعد الأمر السابق فإنه يظهر العبارة ثم ينتقل لسطر جديد .**

مثال : الكود التالي يظهر عبارة ترحيبية ، ثم يقوم بالعد التصاعدي كل ثانية.

```
int i;
void setup(){
  Serial.begin(9600);
  Serial.println("welcome Sami ");}
void loop(){
  Serial.println(i);
  delay(1000);
  i++; }
```

لتجربة الكود في المعمل الافتراضي **اضغط هنا**

للاستزادة : شرح Serial.print

خيارات إضافية لإظهار القيم بالأمر print

```
Serial.print(78, BIN) gives "1001110"
Serial.print(78, OCT) gives "116"
Serial.print(78, DEC) gives "78"
Serial.print(78, 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"
```

إرسال القيم من شاشة السيريال إلى الأردوينو أثناء تشغيله

لاستقبال أي قيمة أو عبارة من المستخدم عبر شاشة السيريال إلى الأردوينو فإننا عادة ننفذ ثلاث خطوات:

1- إرسال رسالة تظهر على الشاشة و تشرح المطلوب من المستخدم

2- نوقف تنفيذ الكود بانتظار المستخدم ليدخل قيمة

3- نستخدم الأمر المناسب لقراءة القيمة المرسلة من المستخدم (int, float,char,String)

`Serial.available();`
`while(Serial.available()==0){}`
`if(Serial.available()>0){...}`
	يستخدم هذا الأمر للكشف إذا قام المستخدم بإرسال أي قيمة أثناء تشغيل الأردوينو عبر شاشة السيريال.
	الطريقة **while** توقف تنفيذ الكود بانتظار دخل (الأسهل)
	if طريقة أخرى للكشف عن الدخل لكنها لاتوقف تنفيذ الكود

`char x=`
`Serial.read();`
`if (x=='y'){...}`
	هذا الأمر يعمل على قراءة بايت واحد تم ارساله من المستخدم ، يقرأ البايت كترميز ASCII فإذا أرسل المستخدم 1 فإن القيمة التي ستتخزن هي '1' وهي نفسها 49

`int x=Serial.parseInt();`	قراءة عدد ووضعه في متغير من نوع int
`float y=` `Serial.parseFloat();`	قراءة عدد و وضعه في متغير من نوع float
`String z=` `Serial.readString();`	قراءة عبارة من المستخدم و وضعها في متغير من نوع String

مثال: الكود التالي يطلب رقم من المستخدم يكون عدد ثواني التأخير لتنفيذ وميض.

```
float d;
void setup(){
```

```
  serial.begin(9600);
  pinMode(13,OUTPUT);
  Serial.println("please enter the delay time ex:0.5 ");
  while(Serial.available()==0){}
  float d=Serial.parseFloat(); }
void loop(){
  digitalWrite(13,1);
  delay(d*1000.0);
  digitalWrite(13,0);
  delay(d*1000.0); }
```

في التمرين التالي سنحاول استخدام حلقة while و حلقة for و شاشة السيريال :-

عمل الكود هو سؤال المستخدم عن عدد الومضات و سرعتها . و بعد ادخال المستخدم لهذه القيم، سيتم تنفيذها ثم سؤاله مرة ثانية و هكذا.

```
int led = 12;
void setup() {
  pinMode(led, OUTPUT);
  Serial.begin(9600);   } //تشغيل التراسل مع شاشة السيريال
void loop() {
  Serial.println("how many blinks? ");//اظهار رسالة على الشاشة
  while(Serial.available()==0){}//التوقف وانتظار دخل من المستخدم
  int n=Serial.parseInt();  //ينقل الرقم من المستخدم إلى متغير
  Serial.println("what is the delay in (ms)? ");
  while(Serial.available()==0){}
  int d=Serial.parseInt();
  for(int i=n ; i >0 ; i--){
    digitalWrite(led,HIGH);
    delay(d);
    digitalWrite(led,LOW);
    delay(d);  }   }
```

لمشاهدة الكود في المختبر الافتراضي: https://circuits.io/circuits/5247139-jeem2_book_ex4

شرح الكود السابق:

الـ LED سيتصل مع المنفذ 12 لذا نحتاج مقاومة 220 لحماية الـ LED

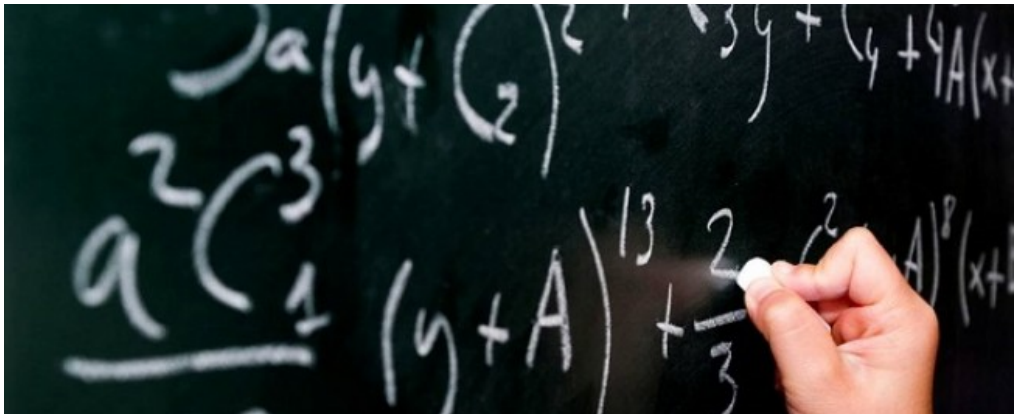
لإدخال أي قيمة من شاشة السيريال ننفذ ثلاث خطوات عادة:

1- اظهار رسالة للمستخدم تطلب معلومة معينة

2- انتظار المستخدم حتى يدخل القيمة المطلوبة
3- وضع القيمة في المتغير المناسب (رقم صحيح ، رقم كسري ، نص كتابي)

الأمر `Serial.println` يعمل على إظهار عبارة أو قيمة متغير على شاشة السيريال
الأمر `while` **مع** `Serial.available` هنا يوقف الكود بانتظار ادخال قيمة من جهة المستخدم.
الأمر `Serial.parseInt` يعمل على قراءة رقم صحيح مرسلة من الكمبيوتر.
لاحظ طريقة عمل الـ for بحيث يكرر الحلقة لعدد n من المرات.

الحسابات والعمليات المنطقية calculations and logic



بالتأكيد أن الحسابات هامة جدا جدا في معظم التطبيقات . الحسابات البسيطة (الجمع و الطرح و الضرب و القسمة) و حسابات أكثر تعقيدا أحياناً (الأس ، الجذر ، الزوايا sin , cos , tan) وعمليات منطقية (AND , OR , NOT) و عمليات دقيقة على مستوى البت الواحد في الرقم (مثل تحريك البت خطوات لليمين أو اليسار) كما يمكنك التعبير عن الرقم بأنظمة عددية مختلفة غير العشري مثل النظام الثنائي و السداسي عشر . هذه التحويلات ستظهر فائدتها في مشاريع قادمة.

في هذا الكتاب لن نتعمق في الحسابات باستخدام الأردوينو لكننا سنناقشها لبعض الوقت ، فبالتأكيد ستحتاج لبعض الحسابات في برامجك القادمة.

إجراء العمليات الحسابية المعروفة operators	+ - * / ( )
أمر مختصر يعمل على زيادة أوطرح 1 من قيمة المتغير i	i-- ; i++;
أمر مختصر يعمل على زيادة 3 أو أي رقم آخر لقيمة المتغير .. كأنها x = x+3 مثلاً	x-=3; x+=3;

مثال: صمم كود يحول درجة الحرارة من درجة مئوية C إلى فهرنهايت F

$$F = C \cdot \frac{9}{5} + 32 \qquad C = (F - 32) \cdot \frac{5}{9}$$

```
float  C=23.48;
void setup(){
  Serial.begin(9600);
  float F=C*9/5+32;// وضع الأقواس أحيانا يسبب مشاكل
  Serial.println(F); }
```

وللتحويل بالعكس شاهد الكود

```
float  F=205.00;
void setup(){
  Serial.begin(9600);
  float C=(F-32.0)*5.0/9.0;
  Serial.println(C); }
```

ملاحظة: برمجة الأردوينو غير مخصصة لإجراء الحسابات وقد واجهت نتائج خاطئة كثيراً بسبب ترتيب الأقواس أو كتابة فاصلة عشرية. ترتيب إجراء العمليات و نظام الأقواس يختلف عن للغات المتقدمة و المناسبة جدا لإجراء حسابات (مثل ماتلاب أو بايثون) لذلك أنصح دائما بتجربة الكود عدة مرات والتأكد من النتائج قبل اعتماده.

بعض المشاكل الشائعة:-

اجراء العمليات بين متغيرات int مع float

عند كتابة رقم صحيح في معادلة تحتوي float اكتبه بالطريقة (5.0) ضع النقطة حتى لو لم تحتجها.

الأقواس لا تفهم حسب الطريقة الشائعة ، جرب تغييرها إذا وجدت نتائج غريبة.

الأمر (%) modulo هذا الأمر الغريب (غير موجود في الرياضيات التي درسناها) ولكنه مفيد في عالم البرمجة و الحسابات، وسنستخدمه في عدة مشاريع قادمة. فكرته باختصار أنه يعيد لك باقي القسمة.

مثال : 20 \ 6 = 3 والباقي 2 لذا:

```
x=20%6;     >>   x=2
```

يعيد لك باقي القسمة فقط modulo مفيد في تطبيقات قليلة ولا يعمل مع الـ floats	y = 9 % 3 >> y=0 x = 10 % 3 >> x=1

مثال : لدينا 9583 ثانية . نود معرفة كم ساعة و كم دقيقة و كم ثانية

علماً أن الدقيقة 60 ثانية و الساعة 3600 ثانية.

```
int x=9583;
int sec = x%60;
int min=x%3600/60;
int hr=x/3600;
```

المصفوفات Arrays

المصفوفة هي وضع مجموعة من القيم بشكل منظم ضمن إطار واحد (مصفوفة واحدة). تخيل معي طبق البيض مثلاً 😀... في البرمجة هذا يساعدنا كثيرا لإجراء عمليات متتابعة بكود سهل .

مثال: لو كان عندنا درجات طلاب فبدل أن تكتبها هكذا:

```
int x=85 , y=89 , z=76;
```

فالأفضل أن تكتب في مصفوفة كما بالشكل:

```
int  a[ ]={85,89,76};
```

ملاحظة : للوصول لأحد عناصر المصفوفة استخدم طريقة التالية:-

```
a[0] >> 85
a[1] >> 89
a[2] >> 76
```

*لاحظ أن العنصر الأول برقم 0 وليس 1

في الاستخدامات المتقدمة (مثل بعض المكتبات) يجب عليك إرسال بعض المعلومات على شكل مصفوفات . و سنشرحها في وقتها.

مثال: نود جعل المنافذ 0،1،3،4،6،7،9،10،12،13 مخارج OUTPUTS

بينما المنافذ 2،5،8،11 نريدها أن تكون منافذ دخل مع ربطها بمقاومة رفع الجهد الداخلية PULLUP

بدل أن نكتب الأوامر في 14 سطر ، بإمكاننا الاستفادة من المصفوفات كما يلي:

```
int OT[]={0,1,3,4,6,7,9,10,12,13};
int IN[]={2,5,8,11};
for(int i=0; i<10; i++){ pinMode(OT[i] ,OUTPUT);}
for( i=0; i<4;i++){ pinMode(IN[i],INPUT);}
```

تمرين : ضع درجات 10 طلاب في مصفوفة، ثم احسب متوسط الدرجات

هذه الأوامر حسابية معروفة ، و لن نشرحها في هذا الكتاب !!

يحسب 5 أس (للقوة) 2 = 25	`pow(5,2);`
يحسب الجذر التربيعي لـ 16 و يساوي 4	`sqrt(16);`
يعيد القيمة المطلقة (بدون سالب) لـ x	`abs(x);`
حساب الدوال المعروفة sin , cos , tan	`sin( );cos( ); tan( );`
حساب اللوغاريتم logarithm	`log( );`

كتابة القيم بالأنظمة العددية : الثنائي، السداسي عشر Binary , HEX

توجد عدة أنظمة عددية و يستخدمها الكمبيوتر و المبرمجين وقت الحاجة. النظام العددي المعروف يسمى النظام العشري decimal بينما الكمبيوترات وجميع الإلكترونيات الرقمية تعمل بنظام عد يسمى الثنائي Binary كما يوجد أنظمة أخرى مثل السداسي عشر HEX وهو نظام وسيط بين البشر و الكمبيوتر وله استخدامات عديدة.

عندما تضع قيمة في متغير (مثلاً x) فبإمكانك أن تضعها بأي نظام عددي حسب الحاجة.

الـ x يحتوي قيمة تساوي 100 النظام العشري (العادي)	`int x=100;`
الـ y يحتوي القيمة 4 لكن تم كتابتها بالنظام الثنائي	`int y=0b100;`
الـ z يحتوي قيمة 256 لكن تمت كتابتها بالنظام السداسي عشر	`int z=0x100;`

بالمناسبة : عندما تظهر قيمة على شاشة السيريال يمكنك إظهارها بأي نظام تود.

```
Serial.print(78)      gives "78"
Serial.print(78, BIN) gives "1001110"
Serial.print(78, OCT) gives "116"
Serial.print(78, HEX) gives "4E"
```

العمليات المنطقية NOT , OR , AND

إذا درست بعض الإلكترونيات فستكون قد درست هذه العلاقات المنطقية . ستعرف أنها علاقة عادة بين مدخلين ولها خرج. بدون التعمق كثيراً في الموضوع نستخدم هذه العمليات عادة مع الشروط و التي يكون ناتجها نعم أو لا

AND	يجب أن يتحقق الشرطين معاً	**&&**
OR	إذا تحقق أي واحد من الشرطين	**||**
NOT	يستخدم لعكس الحالة 0 إلى 1 والعكس	**!**

مثلاً : أنت تريد أن يصدر صوت تنبيهي في حال أن المدخل التماثلي يقرأ قيمة أقل من 200 وقيمة المتغير x = 0 (يجب أن يتحقق الشرطان حتى يعمل التنبيه)

```
int  a=analogRead(A0);
if(a<100  &&  x==0) {tone(2,500,1000);}
```

مثال : تريد الضوء أن يعمل إذا كان الدخل التماثلي أقل من 400 أو أكثر من 600

```
int a=analogRead(A0);
if(a<400  ||  a>600){digitalWrite(13,1);
```

مثال : نريد أن يعد x من 1 إلى 10 ثم تتغير حالة الضوء (13) و يعيد العد من جديد

```
int x=1;
bool  state=0;
void loop(){
  if(x==11){state=!state; x=1;}
  x++; }
```

العمليات على مستوى البت Bit operators

في بعض التطبيقات المتقدمة (مثل التراسل مع شرائح إلكترونية ببروتوكول SPI) ستحتاج أن ترسل أكواد رقمية دقيقة . ويجب التحكم الدقيق بكل بت في الرقم قبل إرساله.
هذه المجموعة من الأوامر نادرة الاستخدام . لذا سنذكرها بسرعة وبدون أمثلة

الكتابة و القراءة من بت واحد:

كل متغير من أي نوع سواء كان int أو غيره يتكون في النهاية من عدد من البت bits و يمكنك تغيير واحد منها مثل المثال:

```
int x=0b10110111011110;
bitWrite(x,0,1); // 1 سوف يغير آخر خانة في اليمين إلى
bool y=bitRead(x,5); //يقرأ البت السادس من اليمين في قيمة إكس
```

*لاحظ ليس من الضروري تخزين x بالصيغة الثنائية ، لكنها أسهل للتبع هنا.

الإزاحة bit shift : مثال : لدينا متغيرين (الأوضح أن نكتبهما بالنظام الثنائي ، لكن يمكن كتابتهما بأي نظام عددي)

```
byte  x=0b11001011;
byte  y=0b00010011;
```

والمطلوب هو جمع المتغيرين كـ 16 بت في متغير واحد قبل إرساله . هنا سنحتاج للإزاحة:

```
int  z= y<<8 + x;
result:  z=0b1001111001011
```

العمليات المنطقية على مستوى البت Bitwise logic operators :

بإمكانك أيضاً إجراء العمليات المنطقية المعروفة على متغيرات من نفس النوع، مثل : AND , OR , NOT , XOR لن نتعمق في هذا الموضوع لأن الحاجة له نادرة في رأيي.

&	|	^	~
(bitwise and)	(bitwise or)	(bitwise xor)	(bitwise not)

مثال: قم بإجراء عملية XOR على المتغيرين x و y
ثم عملية النفي NOT على المتغير x

```
byte  x=0b11001011;
byte  y=0b00010011;
byte  z=x^y;
byte  a=~x;
void setup(){
  Serial.begin(9600);
  Serial.println(z,BIN);
  Serial.println(a,BIN);}
```

تكتيكات برمجية programing tactics

توجد العديد من الوسائل (الطرق البرمجية) لحل المشاكل و تحسين طرق عمل الكود بشكل عام . بعضها سهل ولا يحتاج لشرح و بعضها قد يحتاج لبعض الشرح و هذا ما سنحاول أن نشرحه في هذا الجزء.

تكتيك التحويل بين النطاقات . باستخدام الأمرين map , constrain

كثيرا ما نحتاج للتحويل المتناسب بين نطاقين رقميين مختلفين. (لا تقلق إذا لم تفهم العبارة السابقة ، لأني سأحاول شرحها الآن) أبسط مثال يحضرني الآن أنك لو حصلت في الجامعة على معدل 4.33 من 5 و أردت أن تفهمه بشكل أفضل و تحوله للنظام القديم (من مئة) فأنت تأخذ القيمة 4.33 من النطاق الأول 0-5 و تريد معرفة القيمة الموازية في النطاق الجديد 0-100.

وفي الأردوينو نحتاج فكرة التحويل بين النطاقات بشكل أوسع. أحد الأمثلة البسيطة. نود التحكم بـ شدة إضاءة LED كمخرج ، بحيث يكون الدخل : مقاومة متغيرة كمقسم جهد.

تكون قراءة الدخل : 0-1023 بينما الخرج التماثلي يجب أن لا يتجاوز 0-255

هنا نحتاج للتحويل من النطاق الأول للنطاق الثاني

تخيل أيضاً أن الضوء لا يبدأ في العمل إلا بعد القيمة 50 و لا أود تحريك المقاومة بدون ملاحظة تغير فمن الممكن تعديل النطاق الثاني إلى 50-255 . أتمنى أنك فهمت فائدة الأمر map

لاحظ أن الأمر map يعمل مع الأعداد الصحيحة فقط ولا يعمل مع الأعداد الكسرية.

```
x=analogRead(A0);              //0<x<1023
y=map(x,0,1023,50,255);        //50>y>255
```

الحاجة للأمر constrain

مثال : افرض أنك تود عمل صوت يتغير حسب حركة يدك فوق مقاومة ضوئية (يتغير التردد حسب شدة الضوء) عند تصنيع الدائرة وجدت أن قراءة الدخل في الضوء = 150 وعندما تغطي الضوء بيدك يصبح الدخل=585

وبخصوص التردد الذي تريد أن يظهر على السماعة لتعطي تباين جيد في الترددات (200-360)

سيكون من الواضح أن تصميم الأمر للتحويل بين النطاقين وتشغيل المشروع هو:

```
x=analogRead(A0);
y=map(x,150,585,200,360);
tone(13,y);
```

هذا الأمر يبدوا جيدا في العمل . لكن !!! ماذا لو تغيرت شدة الإضاءة وخرجت عن المتوقع (أقوى أو أقل) سنحصل على قيم خرج (تردد) خارجة عن التوقعات أيضاً

هنا يستحسن استخدام الأمر constrain والذي يعمل على إبقاء قيمة داخل حدود بدون أن يمكن أن تخرج عنه. لذا سنزيد الأمر التالي على الكود السابق بعد أمر map حتى نحل المشكلة السابقة:-

```
y=constrain(y,200,360);
```

لمشاهدة كود يشرح الفكرة وتجربة التغييرات عليه . شاهد الرابط:

https://circuits.io/circuits/5269652-map-constrain-explained

تمرين: طالب حصل على معدل 3.78 من 4 في الجامعة . أود معرفة معدله بالنسبة المئوية و كم سيكون معدله لو انتقل لجامعة تعتمد نظام المعدل من 5 استخدم map (اضرب *100 أيضاً)

تكتيك استخدام الأعداد العشوائية Random

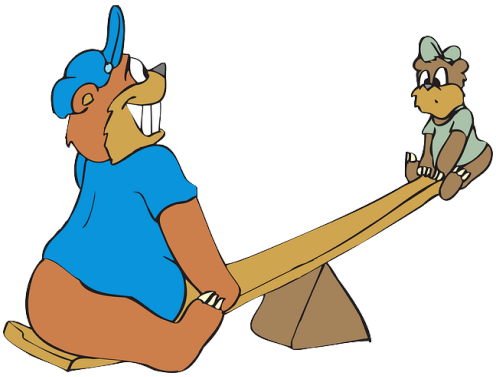
الأمر random يعود لك برقم عشوائي . مثلاً إذا أردت من روبوت (سيارة) أن يسير بشكل عشوائي. فأنت لن تحدد له زاوية التوجيه . فقط استخدم الأمر random . و إذا أردت أن تتوقف سيارة لوقت عشوائي فهنا يمكنك استخدام الأمر random

يعود بقيمة عشوائية من صفر إلى 9	`random(10);`
يعود بقيمة عشوائية من 5 إلى 14	`random( 5 , 15 );`

لاحظ في المثال السابق (القيمة العائدة تتضمن القيمة الأدنى ، ولا تتضمن الحد الأعلى)

تمرين : صمم كود يعمل كأنه نردين المونوبولي ، فيولد قيمتين تتراوح بين 1 و 6 ثم يجمعهما و يعرض النتيجة على شاشة السيريال.

تمرين : صمم كود يولد مصفوفة تحتوي 18 عنصر . و ضع في كل عنصر قيمة عشوائية تتراوح بين 70 و 100 ثم اعرض جميع القيم على شاشة السيريال.



ايجاد القيمة الأكبر أو القيمة الأصغر بين قيمتين

min , max

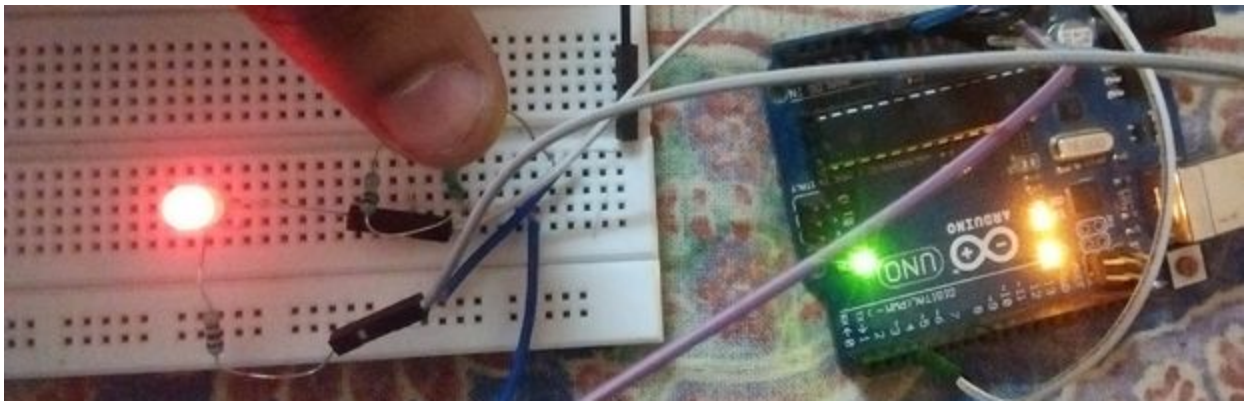
يظهر الأمر هذا بسيط جداً لكنه مفيد جدا في بعض التطبيقات (مثل تطبيق المعايرة في الصفحة التالية) الآن تعلم كيفية استخدام الأمرين min و max

يعود بأعلى قيمة من بين القيمتين	`max(x,y);`
يعود بأقل قيمة من القيمتين	`min(x,y);`

مثلاً لدينا درجات مجموعة من الطلاب في مصفوفة . و نود ايجاد أعلى درجة و أقل درجة في المصفوفة

```
int x[]={5,6,43,9,4,6,8,2,4,55};
int max=0;
int min=100;
void setup(){
  Serial.begin(9600);
  for(int i=0;i<10;i++){
    max=max(max,x[i]);
    min=min(min,x[i]);}
  Serial.print("max=");
  Serial.println(max);
  Serial.print("min=");
  Serial.println(min);}
void loop(){}
```

تكتيك الضبط عند بداية التشغيل Calibration



فكرة الضبط قبل التشغيل هي أحد التكتيكات الجيد استخدامها في بعض التطبيقات التي تحتوي حساس نريد تشغيله في أماكن مختلفة .

مثال : في البيت عند تغطية الحساس الضوئي تكون قراءة الحساس التماثلية 200 و عند كشف الحساس تكون القراءة 700 . لكن في المعرض تتغير القيم بحيث : عند تغطية الحساس تكون القراءة 100 و عند كشفه تكون القراءة 500 كيف نصمم الكود ليتم الضبط الآلي عند بداية التشغيل.

مراحل تنفيذ التكتيك:

1- تعريف متغيرين min , max لنضع فيهما القيم المتوقعة الأعلى و الأقل آليا بعد قليل

2- ضع القيمة 1023 في min والقيمة 0 في max (تظهر العملية معكوسة ، لا تقلق)

3- عند بداية التشغيل سنخصص عدة ثواني (مثلاً 10 ثواني لضبط قيمة min و max حسب التشغيل)

4- أثناء الثواني الـ 10 الأولى سيقوم المستخدم بتغطية الحساس ثم الكشف عنه بشكل متكرر. هذا سيعمل على تعديل قيمة Min و max حسب التشغيل.

5- بعد انتهاء الـ 10 ثواني ، سيعمل الكود اعتمادا على القيم الموجودة في min و max .

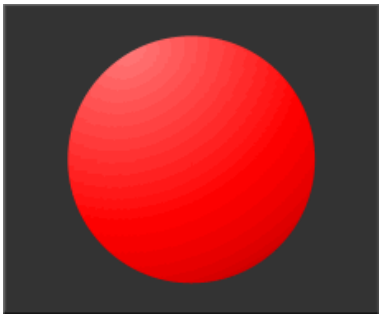
6- سنستخدم الأمرين map و constrain لضبط التشغيل.

مثال لكود يعمل على الضبط الآلي بداية التشغيل .

```
int min = 1023;
int max = 0;
int sensor ;
void setup() {
  while ( millis() < 10000){
    sensor = analogRead(A0);
    if (sensor > max ) { max = sensor;}
    if (sensor < min ) { min = sensor;}}}
void loop(){
  sensor = analogRead( A0 );
  sensor = map ( sensor , min , max , 0 , 255 );
  sensor = constrain ( sensor , 0 , 255 );  }
```

تمرين : اكتب كود بحيث يحدد نطاق التشغيل المتوقع لمقاومة ضوئية analogRead في البداية في متغيرين Min و Max (فترة المعايرة calibration) ثم يحدد النطاق المتوقع (مثلاً 230-350) بعد ذلك نود اصدار صوت بترددات متناسبة من الدخل بحيث يكون نطاق الترددات (260 - 400)
تمرين: مثل أي مثال يحتوي ضبط مبدئي calibration المطلوب هو إضافة زر رقمي (ضاغط) يعمل على إعادة الضبط في أي وقت

تنفيذ برنامج الوميض بدون أمر التأخير delay



أعتقد أن الموضوع يبدو غريباً _ إذا كان بإمكاني تنفيذ الوميض بكل سهولة مثل الكود السابق Blink ، لماذا أنفذها بطريقة أصعب؟ ☹ إذا كان العمل المطلوب من الأردوينو بسيط جداً و المطلوب من الدائرة عمل واحد فقط ، فلا بأس من استخدام الأمر delay لكن في بعض التطبيقات المتقدمة لا ينبغي إيقاف عمل الأردوينو بأمر الانتظار delay . فالأمر delay يوقف تنفيذ كل شيء ، بينما في كثير من التطبيقات المتقدمة يجب على الأردوينو أن يقرأ و يشغل العديد من الأشياء باستمرار.
يوجد طرق تجعل الكود يستمر بالوميض بدون أن تتوقف دورة تنفيذ الأوامر

```
bool x=0;
unsigned long T1=0;
unsigned long T2=0;
void setup(){
  pinMode(13,OUTPUT);}
void loop(){
  T1=millis();
  if(T1>T2+1000){
    x=!x;
    digitalWrite(13,x);
    T2=T1;}
// أي أمر في هذا المكان سيتم تنفيذه بدون تأخير
 }
```

الفكرة باختصار هي استمرار التنفيذ في حلقة loop مع قراءة قيمة millis كل دورة بدون توقف.
الأمر millis يقرأ الزمن منذ بداية تشغيل البرنامج بالميللي ثانية. إذا مر من الزمن ثانية كاملة . سيكون قيمة T1 أكبر من T2 بألف ميللي ثانية ، إذا حصل ذلك سيغير x حالته ، و نساوي بين T2 و T1 لنبدأ حساب الزمن من جديد.
لاحظ أن T1 و T2 متغيرات من نوع unsigned int وهذا ليتمكن من قراءة قيمة الزمن لقيم كبيرة

تمرين1: شغل وميض لضوئين بحيث يعملان بسرعات مختلفة . مثلاً D1=300 D2=1000

تمرين2: المداخل زر ضاغط (رقمي) و مقاومة متغيرة _ المخارج LED و سماعة
المطلوب: الضغط على الزر يزيد سرعة الضوء + تغيير المقاومة يغير تردد النغمة

--- Functions : كتابة الدوال ---

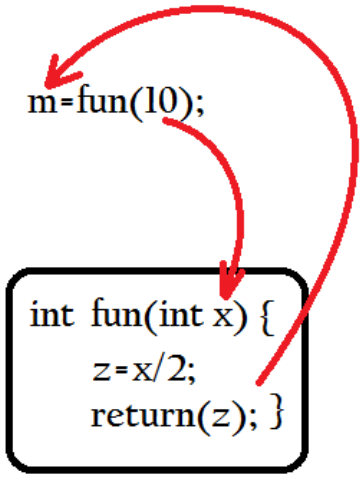
fun( ); fun2(10); fun3(5,15);

الدوال functions : هي طريقة مفيدة في بعض الأحيان لاختصار و تبسيط الكود.

فهي عبارة عن كود جانبي يتم استدعاءه لتنفيذ عمل معين وقت الحاجة ، ثم العودة للكود الرئيسي.

يتم التعرف على الدوال من القوسين اللذين يأتيان بعد اسمها ... مثل:

```
sami();
x= hey(15);
y,z= fun(13,12,600);
```



أمثلة لأكواد يمكن جعلها كدوال واستدعائها من الكود الرئيسي:

تحويل درجة الحرارة من فهرنهايت إلى درجة مئوية.

تحويل الاطوال من قدم الى متر.

تشغيل LED و إطفاؤه.

المداخل و المخارج للدالة: بعض الدوال يتم استدعائها بدون إرسال أي قيمة لها . و تقوم بعمل معين . ثم تنتهي بدون أن تعود بأي قيمة منها .وهذه الدوال تكون من نوع void

مثال: دالة وميض بسرعة ثابتة أو تأخير زمني أو دالة panic لاصدار تحذير صوتي و مرئي.

لذا فمن المهم معرفة هل ستعود قيمة للدالة أم لا ، و إذا كانت ستعود قيمة ، ما هو نوع القيمة؟ int ؟ أو float وربما String ... إذا كانت الدالة لن تعيد أي قيمة فيجب تعريفها بـ void

أمثلة على دوال موجودة و تم تعريفها في برمجة الأردوينو سابقاً: (غير موجودة في لغة C الأصلية)

`delay(1000);`	هذه الدالة لها دخل واحد (1000)وليس لها أي قيمة راجعة void
`digitalWrite(13,1);`	هذه الدالة لها دخلين ولا تعود منها أي قيمة void
`digitalRead(3);`	دالة لها دخل واحد (3) والعائد قيمة 0-1 وممكن تكون int أو boolean
`analogRead(A0);`	دالة لها دخل واحد والعائد يكون عدد 0-255 و تكون من نوع int أو byte

أمثلة لكود و دالة تعمل على تحويل درجة حرارة من فهرنهايت الى مئوية: مع العلم أن :

$$C = (F - 32) * \frac{5}{9}$$

```
int F1=0; //here put the fahrenheit temp
int F2=10;
int F3=25;

void setup(){
int  C1=FtoC(F1);  //FtoC will convert Fahrenheit to degrees
int  C2=FtoC(F2);
int  C3=FtoC(F3); }

void loop(){ }

int FtoC(int x){  //FtoC is a function to convert Fahrenheit to degrees
  int C = x-32*5/9;
  return(C);}
```

مثال لعمل دالة (Blink) ترسل لها قيمتين عدد مرات الوميض و التأخير الزمني.

```
int LED = 13;
void setup(){
  Blink(10,300);
  Blink(5,1000);
  Blink(2,5000);}

void Blink(int T, int D){
  for (   ; T>0 ; T--){
    digitalWrite(LED,HIGH);
    delay(D)
    digitalWrite(LED, LOW);
    delay(D); } }
```

مثال آخر : مثل السابق ، لكن هذه المرة الدالة تستقبل 4 قيم : رقم المخرج الرقمي ، عدد الومضات ، زمن الإضاءة ، زمن الإنطفاء.

```
void loop(){
  blink2(13,3,500,500);
  blink2(12,5,300,200);
  blink2(11,10,200,800); }

void blink2(int P, int T , int DON , int DOF){
  for (    ; T>0 ; T--){
    digitalWrite(P,HIGH);
    delay(DON)
    digitalWrite(P, LOW);
    delay(DOF); } }
```

تمارين: كتابة دوال create a function

1	اكتب دالة بحيث ترسل لها قيمة واحدة فتعمل الدالة على إظهار وميض على المخرج 13 و تكون القيمة هي التأخير الزمني بالمللي ثانية
2	اكتب دالة بحيث نرسل لها قيمتين قيمة قيمة التأخير الزمني وقيمة عدد النبضات
3	اكتب دالة نرسل لها ثلاث قيم التأخير الزمني ، عدد النبضات ، رقم المخرج المطلوب
4	اكتب دالة تحول القيمة من ريال الى دولار ودالة أخرى تحول من دولار الى ريال
5	اكتب دالة واحدة تعمل عمل الدالتين في الفقرة السابقة ، يجب على المستخدم إرسال RTD أو DTR للدالة لتعرف ما المطلوب منها.

استخدام طريقة switch / case في الدوال:-

أحد الطرق المستخدمة في بعض الدوال هو تحديد التصرف حسب قيمة متغير. وهذه الطريقة لا تختلف كثيرا عن استخدام الأمر if ولكن لا تتفاجأ إذا شاهدتها.

طريقة عمل الأمر هو المقارنة بين قيمة المتغير var مع القيمة أمام كل حالة (1 ، 2) فإذا لم تتطابق قيمة var مع أي حالة ؛ يقوم بتنفيذ الأوامر عند عبارة default

شاهد الكود التالي كمثال:

```
switch (var) {
    case 1:
      //do something when var equals 1
      break;
    case 2:
      //do something when var equals 2
      break;
    default:
      // if nothing else matches, do the default
      // default is optional
    break;
  }
```

لقراءة المزيد زر صفحة لشرح الأمر هـــنـــا

-- تمارين برمجية على الصفحات السابقة --

تمرين1: التحكم بمخرج رقمي digital output

أ	وميض Blink ____ اجعل LED يشتغل وينطفئ بشكل متكرر.
ب	جرب تغيير الـ مخرج المستخدم و تغيير السرعة.
ج	عرف متغيرين في البداية d لتحديد التأخير و ledPin لتحديد رقم المخرج المستخدم.
د	أضف ملاحظات (شرح comments) لكل أمر و شرح لعمل الكود في الأعلى استخدم : // */ /*
هـ	اجعل الوميض يبدأ سريعاً ثم يتباطأ مع الوقت.
و	استخدم طرق تعريف المتغيرات const و static في الكود.
ز	صمم كود يظهر 5 نبضات سريعة ثم 3 بطيئة (بدون استخدام حلقة)
ح	مثل السابق لكن استخدم الحلقة for
ط	عرف متغير الزمن d داخل الـ setup loop بطريقة static int
ي	شغل الوميض بدون استخدام الأمر delay استخدم millis بدلا عنه. (متقدم)

تمرين2 : التحكم بعدة مخارج رقمية multi digital outputs

أ	صمم كود بحيث يتناوب ضوئين (2LEDs) على التشغيل بسرعة ثابتة
ب	نفس الفكرة لكن استبدل أحد الـ LEDs بـ صوت (استخدم الأمر tone)
ج	صمم الكود بحيث يغمز الليد الأول لـ 5 مرات و الثاني لـ 7 مرات (لا تستخدم حلقات loops)
د	مثل الفقرة (ج) لكن استخدم الحلقة for
هـ	اجعل كل LED يومض بسرعة مستقلة عن الآخر مثلا d2=500 d1=300 يجب استخدام أمر millis
و	اصنع مصفوفة في البداية، تحدد عدد نبضات كل LED بالتسلسل x[ ] = { 3 , 2 , 4 , 1 , 6 , 3 , 4 , 4 };
ز	مثل الفقرة السابقة ولكن صمم مصفوفتين ، مصفوفة خاصة بكل ليد !

تمرين3: مدخل رقمي Digital input

أ	استخدم مفتاح ضاغط (push button) كمدخل رقمي، و LED كمخرج. المطلوب أن يعمل وميض Blink بطيء متكرر ، وعند الضغط على الزر يعمل الوميض بشكل أسرع. في بداية الكود عرف متغيرين d1=1000 و d2=300 مثلاً
ب	استخدم زر ضاغط لتشغيل تنبيه صوتي. استخدم الأمر tone
ج	لدينا 10 ليدات نريدها تعمل معاً عند الضغط على زر sw المطلوب يكون بينها تأخير زمني بسيط (الأول ثم الثاني ثم الثالث وهكذا) كما أنها تنطفئ بنفس الطريقة عند ترك الزر.
د	صمم ما يشبه الساعة الرملية (بأضواء LEDs) مع زر يعمل كأنه يقلب الساعة.

تمرين 4 : عدة مداخل رقمية

أ	مفتاح ON و مفتاح OFF للتحكم بـ LED استخدم &&
ب	تغيير قيمة متغير x زر يزيد 1 و زر ينقص 1 ، اعرض قيمة x على شاشة السيريال.
ج	اضبط 3 مداخل رقمية وسماعة بسيطة واحدة ، بحيث كلما ضغطت على زر يصدر صوت مختلف.

تمرين 5 : مخرج تماثلي

أ	اضبط ضوء LED بحيث يعمل ويطفئ (مثل الوميض) ولكن بشكل تدريجي.
ب	اضبط مدخل رقمي واحد (زر ضاغط) بحيث تستمر شدة الإضاءة في التغير مادام ضغط الزر، و تثبت الإضاءة إذا تركت الزر.
ج	الإمساك على مفتاح UP يزيد قيمة الجهد في مخرج تماثلي ، و الزر DN يقلله . و يكون بتدريج بطيء.
د	زرين UP و DN لإخراج جهد تماثلي من الأردوينو (من 0 إلى 5 فولت)

تمرين 6 : عدة مخارج تماثلية

أ	لدينا 3 أضواء LEDs مختلفة الألوان ، نود أن تزيد إضاءة كل ضوء و تقل بشكل تدريجي
ب	طبق الفقرة السابقة على ضوء LED ثلاثي الألوان (بأربع أطراف)
ج	طبق الفقرة السابقة بحيث يمكن في أعلى الكود تحديد سرعة تغير كل لون على حدة.
د	استخدم 6 مفاتيح رقمية للتحكم بشدة إضاءة كل لون .

تمرين 7 : مدخل تماثلي :

أ	اضبط مقاومة متغيرة كمقسم جهد إلى مدخل تماثلي، اعرض القراءة على شاشة السيريال.
ب	الدخل مقاومة متغيرة و الخرج 4 ليدات تعمل بطريقة رقمية . مثلا الجهد 0 ، لا يعمل أي ليد ، الدخل 1 فولت يعمل ليد واحد الدخل 2 فولت يعمل 2 و هكذا ...
ج	إضافة للفقرة السابقة ، عندما يزيد الدخل عن 4 فولت ، اجعل جميع الـ ليدات تظهر وميض سريع.
د	مقسم جهد يحتوي مقاومة ضوئية _ يصنع ما يشبه الإضاءة الآلية (تشتغل في الظلام ، و تنطفئ في النهار)
هـ	استخدم الثيرموكوبل (مقياس حرارة بسيط) كدخل وإذا خرجت الحرارة عن القيمة المتوسطة شغل انذار صوتي.

تمرين 8 : عدة مداخل تماثلية :

أ	الدخل مقاومتين متغيرتين والخرج LED واحد بحيث أحد المقاومتين تحدد زمن التشغيل و الأخرى زمن القطع
ب	الدخل عصا تحكم تماثلي و الخرج عدد 9 LEDs الهدف أن نتحكم بالعصى لنجعل LED واحد يعمل يعبر عن اتجاه العصا.
ج	3 مقاومات متغيرة تتحكم بشدة إضاءة 3 ليدات ملونة (أو ليد متعدد الألوان)

تمرين9: (عمل بعض الحسابات بالأردوينو)

أ	حول درجة الحرارة من فهرنهايت إلى درجة مئوية (استخدم float)
ب	حول المعدل من 100 إلى معدل من 5 واعرض الناتج على الشاشة SerialMonitor استخدم float
ج	لدينا مصفوفة تحتوي 5 أعداد مختلفة ، نود جمعها . ثم حساب متوسط الأعداد.
د	(حسابي متقدم) لدينا حساسين ضوئيين بينهما مسافة ثابتة (10cm) نود حساب الوقت بدقة بين الإشارتين على الحساسين . ثم حساب السرعة. ثم عرضه على السيريال مونيتور . استخدم الأمر micros
هـ	(حسابي متقدم) لدينا جهاز يحسب الزمن بالثواني وعند قراءة الزمن ، أعطاك رقم كبير جداً (عدة ملايين) ضع القيمة في متغير unsigned long استخدم الأردوينو لإظهار الزمن بطريقة أيام ، ساعات ، دقائق ، ثواني . (واستخدم الأمر modulo %) تذكر : الدقيقة = 60 ثانية ، الساعة = 3600 ثانية و اليوم = 86400 ثانية
و	المطلوب صنع نرد بسيط إلكتروني يظهر رقم عشوائي عند الضغط على زر ضاغط (1-6)
ز	مثل المثال السابق لكن اجعل كل ضغطة تظهر عددين عشوائيين (كأنه نرد المونوبولي)

ح	حول المعدل من 100 إلى معدل من 5 واعرض الناتج على الشاشة SerialMonitor استخدم map و float
ط	نود قراءة قيمة من مدخل تماثلي ثم إخراج قيمة مساوية لها من المخرج التماثلي. لاحظ الدخل 0-255 والخرج 0-1023 صمم الدائرة و الكود. استخدم الأمر map
ي	مثلث قائم الزاوية اذا كان ارتفاعه 6m و قاعدته 4m احسب طول الـ وتر hypotenuse
ك	قوة مقدارها 77 نيوتن تؤثر على جسم بزاوية 60 احسب مقدار القوة على محور x و على محور y

تمرين 10 : شاشة السيريال Serial monitor

أ	صمم ساعة رقمية تظهر الثواني و الدقائق على الشاشة (توجد أكثر من طريقة)
ب	دخل تماثلي 0-1023 اقرأه و اعرضه على شاشة السيريال ثم احسب ما يوازيه لإخراجه من المخرج التماثلي 0-255 و اعرض المكافئ على الشاشة أيضاً --فكر كيف تجعل جهد الخرج يعاكس جهد الدخل 0 >> 5v و 5v >> 0v
ج	التحكم بـ LED بواسطة شاشة SerialMonitor مثلا أدخل رقم 1 لتشغيل الضوء و 0 لإطفائه. اجعل الشاشة ترسل لك التعليمات قبل كل أمر.
د	اجعل التحكم بـ 2 LEDs و يكون الدخل من شاشة السيريال مثلاً 1 . 2 . 3 . 4 للتحكم بالـ LEDs
هـ	صمم كود بحيث يدخل المستخدم اسم لون red green blue فيضيء اللون المطلوب في RGB LED
و	على شاشة serialMonitor يطلب قيمة ، تحدد سرعة الوميض ، ثم تطلب تردد الصوت .
ز	يقوم المستخدم بإدخال قيم للألوان الأحمر و الأخضر و الأزرق _ ثم يعرض اللون المطلوب على RGB LED
ح	ادخل أسماء 10 طلاب في مصفوفة ، و درجاتهم في مصفوفة مختلفة. زد لكل طالب 5 درجات ، على ألا تزيد الدرجات عن 100 أبداً (استخدم : constrain) ثم على الشاشة أظهر المتوسط . وأعلى طالب و أقل طالب.

تمرين 11:: أوامر متقدمة لم أشرحها أمر المقاطعة !

د	نود تشغيل وميض عادي، وفي حالة وصول مقاطعة على الطرف 2D يعمل صوت تحذيري
هـ	مثل المثال السابق سوى أنه يوجد استشعار لمقاطعتين _ D2 و D3 و كل مقاطعة تصدر صوت مختلف.

اذا استطعت حل المسائل البرمجية السابقة فمبروك ...

أنت مبرمج بلغة C ويحق لك الاحتفال 😀

فقط تذكر ... أنت تعلمت البرمجة بلغة C

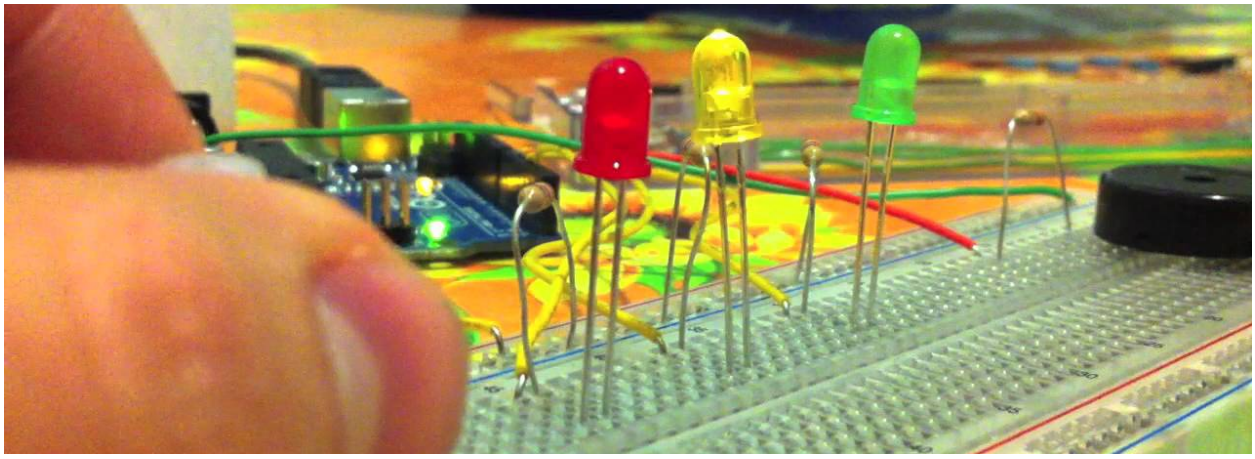
عندما نتقدم في الأردوينو سنضطر لفهم أوامر C++ وهي متقدمة أكثر.

كما أننا سنستخدم الكثير من مكتبات الأوامر والدوال ...

الرحلة طويلة ... لكنها بالتأكيد ممتعة 😊

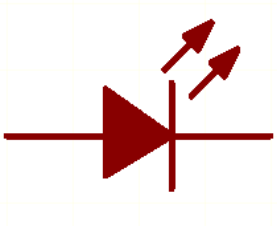
القسم التالي سنشرح بعض مهارات الإلكترونيات المفيدة.

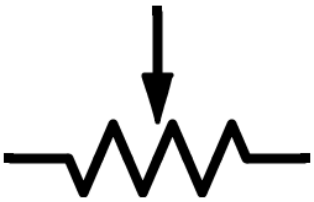
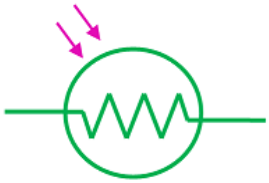
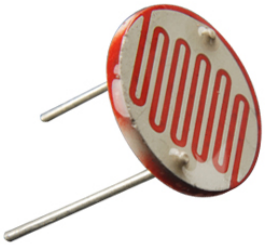
الباب الثالث: إلكترونيات Electronics

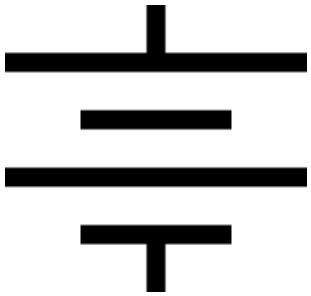
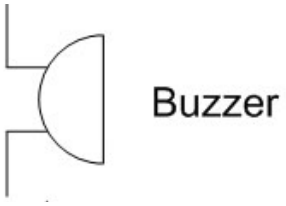
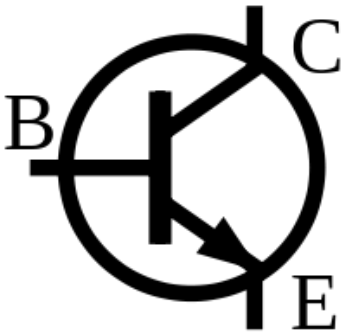
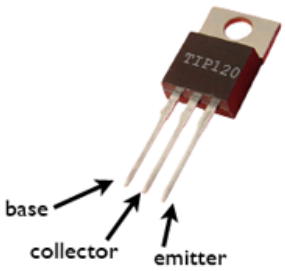
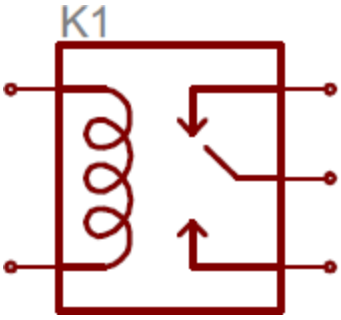
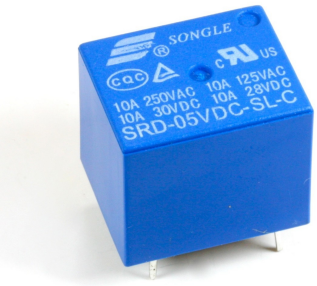


بالتأكيد الإلكترونيات عالم كبير لكننا هنا سنتكلم عما يلزمنا عند استخدام الأردوينو فقط.
الدوائر الإلكترونية هي مجموعة من العناصر و بينها توصيلات (اسلاك) عندما يمر بها التيار الكهربائي فإنها تقوم بعمل معين.

أهم العناصر الإلكترونية التي سنستخدمها :-

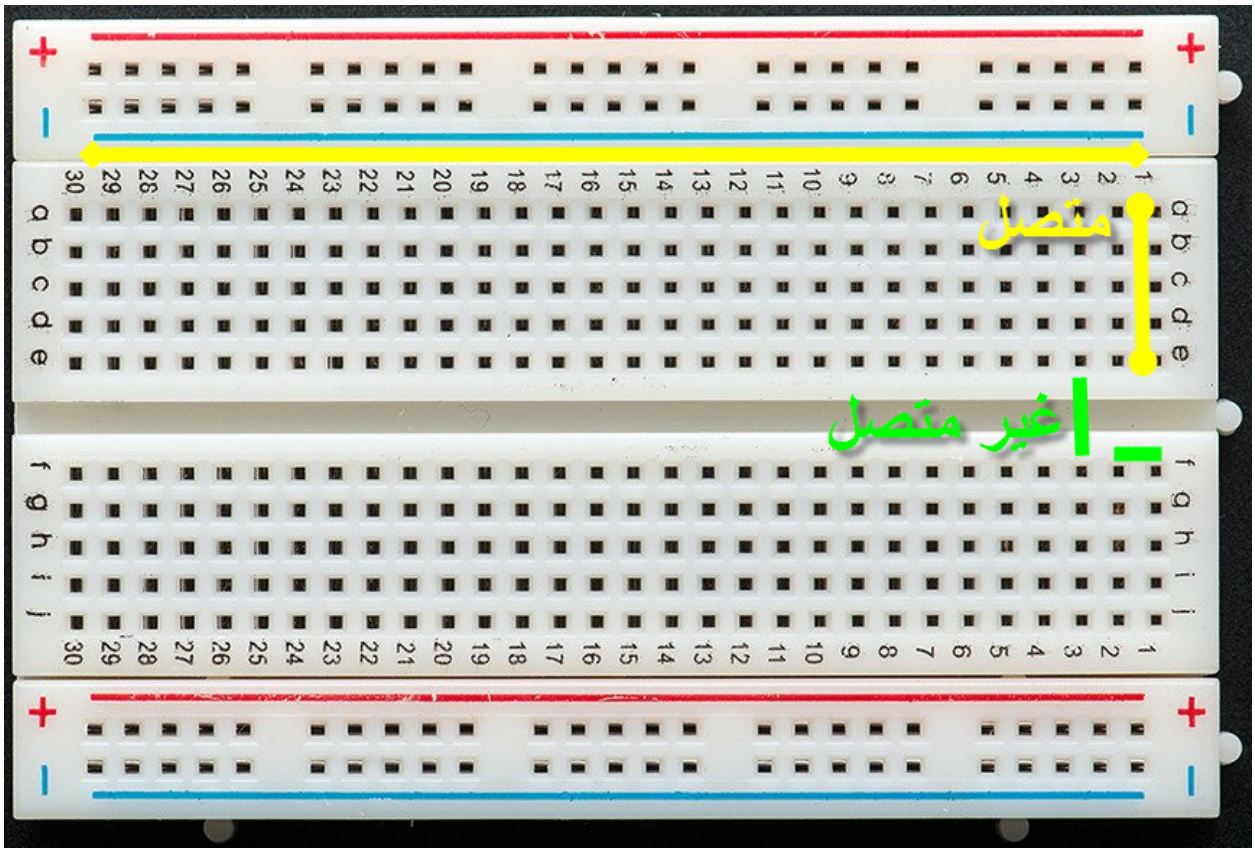
		مبين ضوئي (لـيـد) يعمل على إظهار ضوء عندما نخرج الجهد المناسب من أطراف الأردوينو الرجل الطويلة : آنود و يجب توصيل مقاومة للحماية **LED**
		مقاومة تقلل مرور التيار و تحمي العناصر التي لا تتحمل 5v **Resistor**

		**زر ضاغط** يعمل على توصيل التيار الكهربائي عند الضغط عليه. يستخدم كمدخل رقمي للأردوينو **Push button**
		**مقاومة متغيرة** تعمل عادة كمدخل تماثلي حيث يمكنك يدويا تغيير الجهد الداخل للأردوينو **Potentiometer**
		**مقاومة ضوئية** حساس ضوئي ، يعمل كمقاومة تقل قيمتها كلما تعرضت لضوء أعلى. **LDR**
		**أسلاك توصيل** تعمل على توصيل التيار الكهربائي بين أي نقطتين **Wires-Leads-jumpers**

		**بطارية 9 فولت** تشغيل الأردوينو أو بعض العناصر الأخرى التي تحتاج تيار عالي محرك ، مرحل **Battery**
		**سماعة بسيطة** تعمل على إصدار صوت بسيط حسب التردد الداخل إليها. وليست مصممة لإصدار أصوات دقيقة مثل سماعة الهاتف **Buzzer**
		**ترانزيستور** يعمل على تكبير الطاقة الكهربائية التي تخرج من الأردوينو ، حين أنه في كثير من الأحيان نحتاج لتيار أعلى من 40mA موديل مقترح TIP120 **Transistor**
		**مرحِّـل** يعمل على تكبير الجهد و التيار بدرجة كبير يمكن أن تكون 220v و نحوه **Relay**

لوحة الاختبار Testboard _ Breadboard

إذا حاولت ربط مكونات دائرة إلكترونية مكونة من عدة عناصر فسيكون من الصعب عليك تتبع هذه التوصيلات الحل الأمثل لتوصيل الدائرة الإلكترونية يكون على لوحة الاختبار Test board ما يسمى بـــ Breadboard

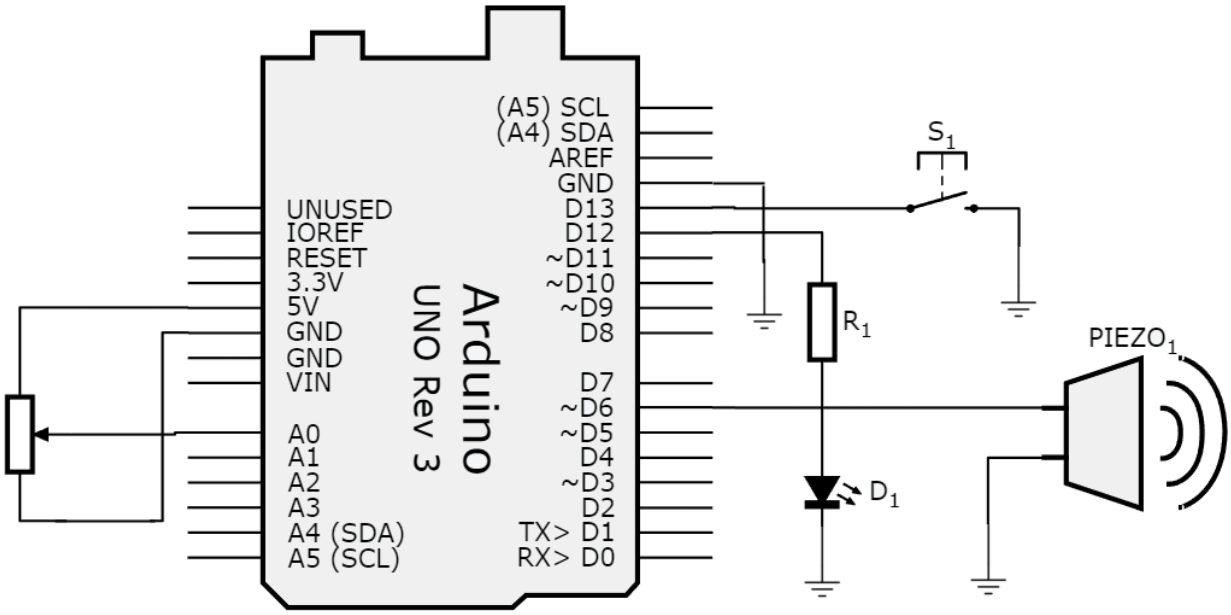


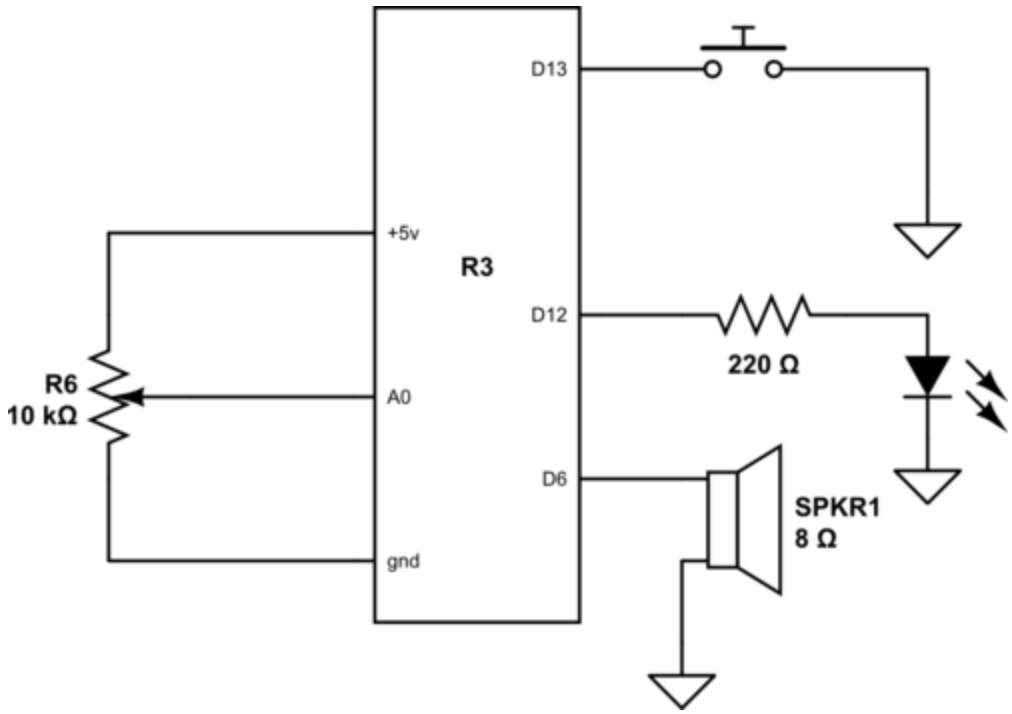
كما ترى توجد نقاط صغيرة . هذه النقاط متصلة من الداخل حسب الخطوط الموضحة.
ففي الصف الأول (1) يتصل a,b,c,d,e معاً و هذه الخمسة غير متصلة مع f,g,h,i,j

لمشاهدة فيديو عن توصيل الدوائر الإلكترونية على لوحة الاختبار : اضغط هنا

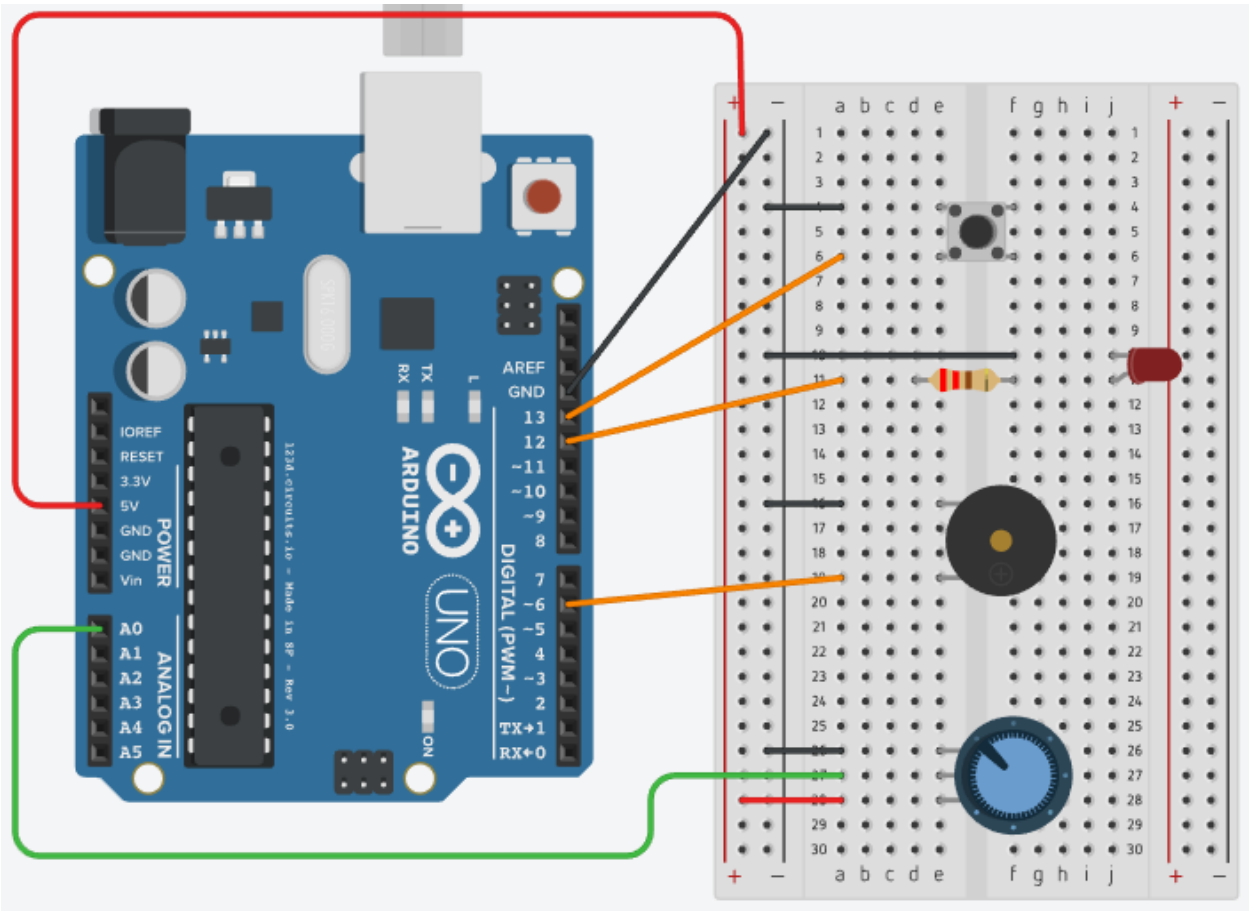
رسم الدوائر النظرية و توصيل الدوائر على لوحة التوصيل Testboard

الهدف من رسم الدائرة النظرية هو معرفة العناصر ومعرفة كيفية توصيلها . لا يوجد طريقة واحدة للرسم . لاحظ أن الرسمين التاليين مختلفين كثيراً لكنهما في النهاية يعبران عن نفس التوصيل.





والآن قارن بين الرسمتين السابقتين ، و تذكر طريقة عمل لوحة التوصيل الـ تيستبورد
ستجد أن هذه الدائرة العملية . هي نفس الدائرة النظرية السابقة...



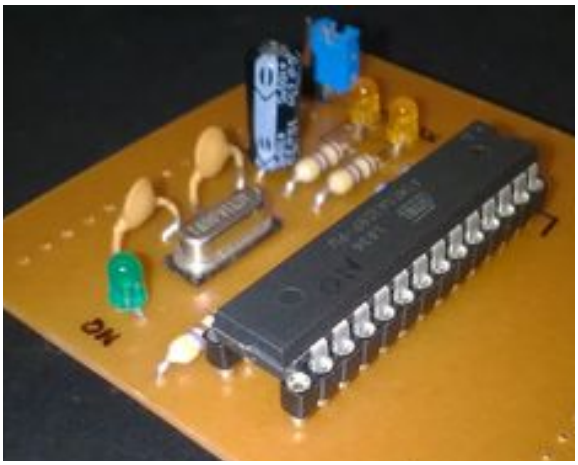
يصعب أن نشرح كتابياً كيفية استخدام لوحة التوصيل الـ تيستبورد . لكن بقليل من التمرين ستتقن هذه المهارة ... صدقني

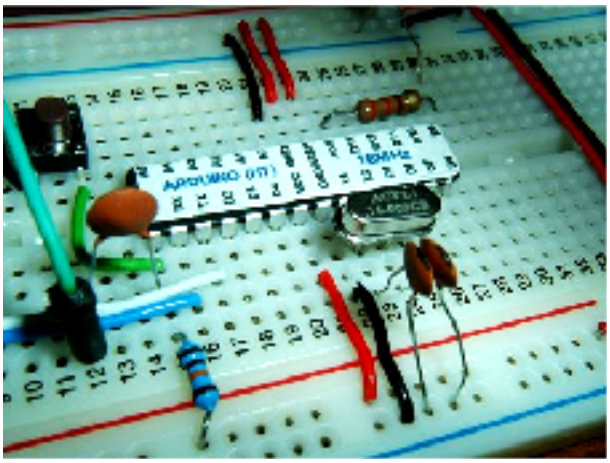
يمكنك في أي وقت التمرن على توصيل أي دائرة على التيستبورد على الموقع الرائع:-
https://www.tinkercad.com/learn/

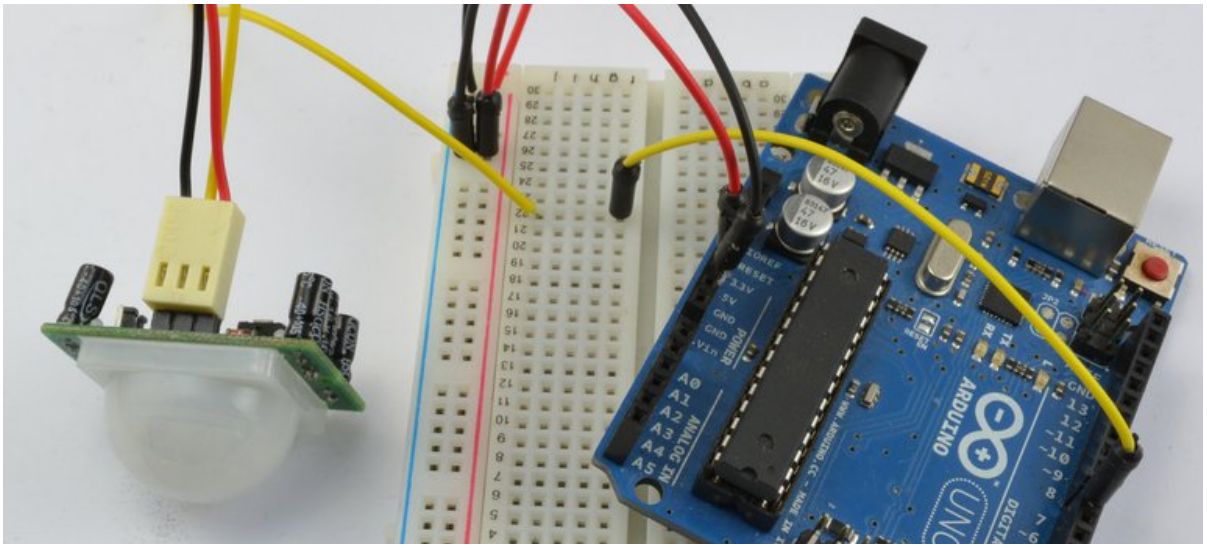
استخدام الشريحة ATmega328P

بدون بورد الأردوينو

في رأيي الشخصي ، هذه أروع مميزات الأردوينو. وينفرد بها **الأردوينو أونو UNO** عن باقي الموديلات. عندما تود أن تصنع دائرتك الإلكترونية فأنت لا تحتاج كل اللوحة (البورد) الكبير للأردوينو . لا تحتاج أن يظهر مشروعك كأنك اشتريت (دائرة جاهزة ووصلت أسلاكها فقط) المطلوب أن تصمم البورد بنفسك وتضع فقط ما تحتاجه من العناصر. انظر للصور التالية.







قارن بين الصورتين العلويتين و الصورة بالأسفل . لن نتجادل أيهما أجمل . لكن المظهر في الدائرة الأسفل يظهر أنك قمت بتوصيل الأسلاك فقط . بينما في الدائرتين العلويتين (اليسرى خصوصا) يظهر أنك

صممت الدائرة وصنعتها بنفسك. كثير من المكونات الموجودة على البورد الـ UNO لن تضعها هنا لأنك لن تحتاجها.

*ملاحظة: في هذا المنهج لن نتحدث عن تصنيع الدوائر المطبوعة pcb حتى لا نشتت تركيزك.

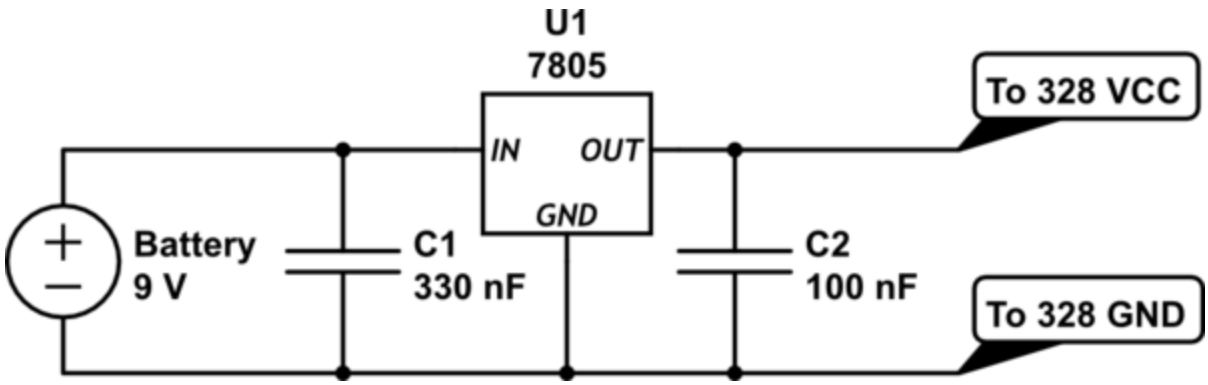
العناصر اللازمة لتشغيل الشريحة ATMega328P خارج الأردوينو قليلة و بسيطة:

1- مثبت جهد : Voltage Regulator

لتعمل الشريحة يجب أن نوصل لها 5v ولكن معظم البطاريات تولد جهد مختلف. فالأفضل هو توصيل جهد أعلى (مثلا 9v) ثم تثبيته باستخدام مثبت جهد بسيط.

المثبت الأمثل لهذا العمل هو : **LM7805** وهو متوفر بكثرة ومنخفض السعر.

ليعمل مثبت الجهد يجب توصيله في دائرة بسيطة كما يظهر بالشكل التالي:



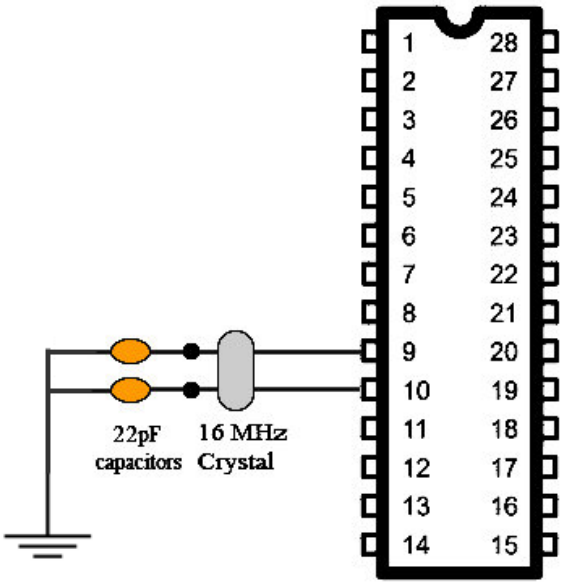
ولتعمل الشريحة **ATMega** يجب توصيل خرج الدائرة السابقة كالتالي:

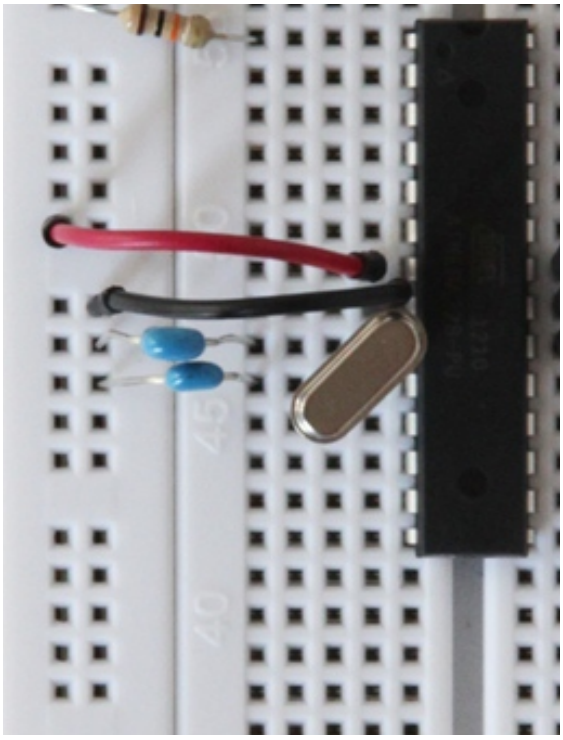
الجهد 5v إلى الأطراف رقم : **7 و 20**

وتوصيل الأرضي 0v إلى الأطراف رقم : **8 و 22**

2-الكريستالة : 16MHz Crystal

الكريستالة هي عنصر إلكتروني هام لضبط تردد (سرعة) عمل الأجهزة الإلكترونية. لتعمل شريحة ATMega328p يجب توصيل كريستالة بين طرفيها رقم 9 و 10 . شاهد الصورة.





المكثفات Capacitors

لعلك لاحظت عنصرين متصلين مع مثبت الجهد ، و عنصرين متصلين مع الكريستالة. هذا عنصر إلكتروني بسيط اسمه مكثف . ويعمل على استقرار الجهد الكهربائي عند نقطة في الدائرة . تتوفر المكثفات في السوق بأسعار منخفضة وقيم كثيرة. وفي الجدول التالي سنكتب لك القيم التي ينصح بها المصنعون

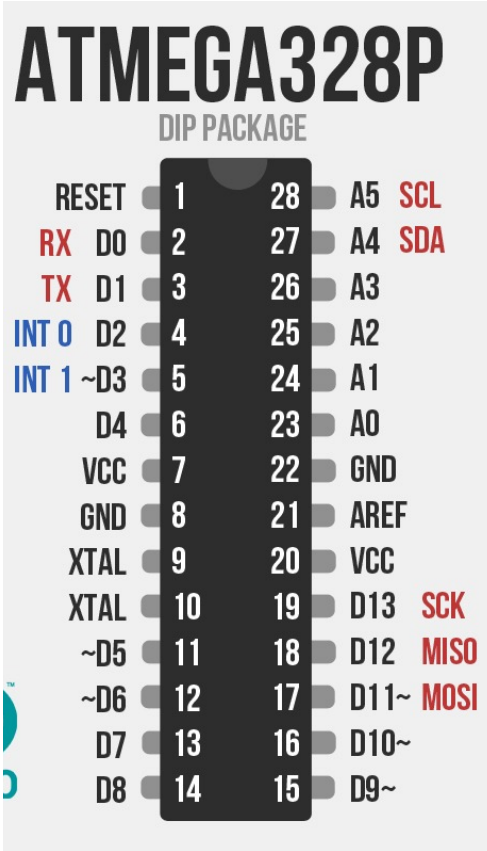
قيم المكثفات الموصى بها من بعض جهات التصنيع	
المكثف قبل مثبت الجهد (جهة الدخل _ البطارية)	330nF
المكثف بعد مثبت الجهد (جهة الخرج_الشريحة Atmega)	100nF
المكثفين المتصلين بالكريستالة	22pF

(في كثير من الأحيان استخدام قيم مختلفة لن يؤثر على عمل الدائرة) لذا فحسب تجربتي يمكن استخدام قيمة واحدة متوسطة مكان جميع المكثفات اللازمة هذا يجعل بناء الدائرة أسهل.

والقيمة المقترحة هي **1nF** ويكتب عليه الكود **102** كما يظهر في صورة المكثف أعلاه.

فكرة تشغيل الشريحة بدون البورد ببساطة:

1- برمج شريحة الـ ATMega328P بشكل عادي وهي مركبة على الأردوينو أونو.
2- انزع الشريحة من الأردوينو و وصلها في مكانها (على التيستبورد أو الدائرة المطبوعة)
3- قم بتوصيل العناصر اللازمة لتشغيل الشريحة (مثبت الجهد و الكريستالة و 4 مكثفات)
4- وصل الأسلاك اللازمة للأطراف المطلوبة لتشغيل الدائرة (المنافذ الرقمية أو التماثلية)



حتى تعرف أماكن الأطراف (المنافذ)

شاهد الرسم يسار الصفحة

ملاحظة: ينصح بتوصيل الطرفين 1 و 21 إلى الـ VCC لتعمل الشريحة بشكل طبيعي.

تمرين: اكتب كود بسيط يعمل على تشغيل ضوئين LED بحيث يومض أحدهما و الآخر تتغير شدته ببطء pwm ثم انزع الشريحة Atmega و شغل الدائرة على لوحة الاختبار الـ testboard مع العناصر اللازمة.

الملتيميتر الرقمي Digital Multimeter

الملتيميتر الرقمي جهاز قياس متعدد الفوائد. فهو يقيس الجهد الكهربائي (الفولت) التيار الكهربائي (الأمبير) المقاومة الكهربائية (الأوم) و التردد (الهيرتز) و اختبار التوصيلية (الزنان) و غيره من القياسات الكهربائية الهامة .

لا تقلق إذا لم تفهم الفرق بين الجهد و التيار و المقاومة الآن . سيشرح لك المدرس الفرق لاحقاً 😊

الآن يهمنا بعض القياسات فقط :

1- اختبار التوصيلية buzzer

2- قياس المقاومة OHM

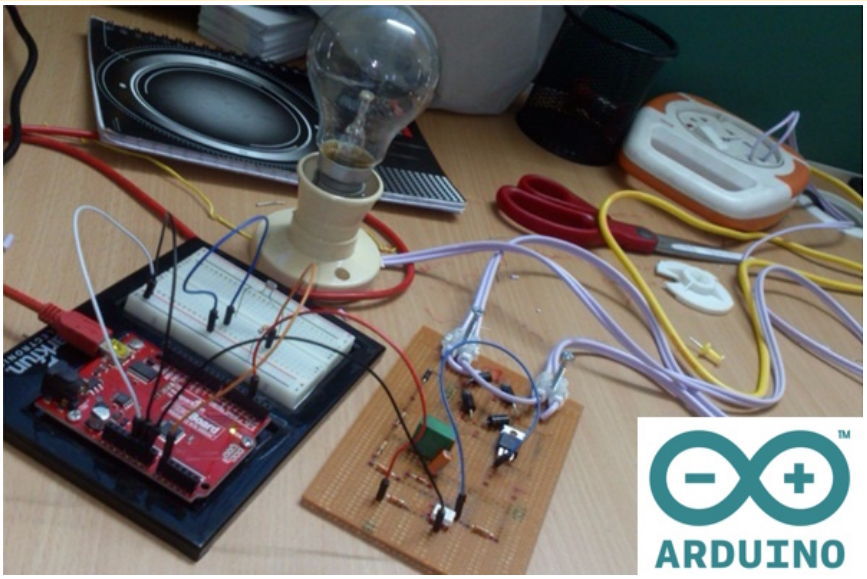
3- قياس الجهد المستمر DC voltage

4- قياس التيار DC Current

5-قياس سعة المكثفات Farad

التفصيل في شرح كل طريقة تتجاوز مجال هذا الكتاب . إذا أحببت الاستزادة أقترح عليك مشاهدة كورس أساسيات الكهرباء في موقع jeem2.com ✌

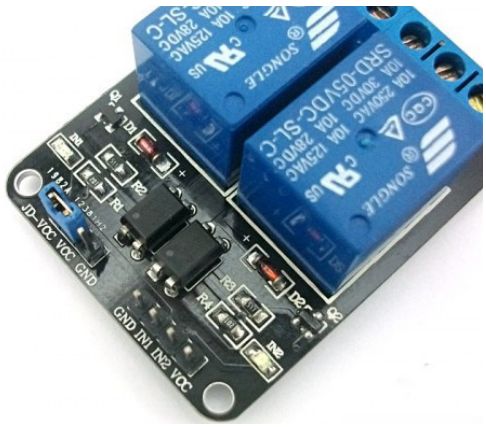
تكبير الإشارة الكهربائية Power Amplification

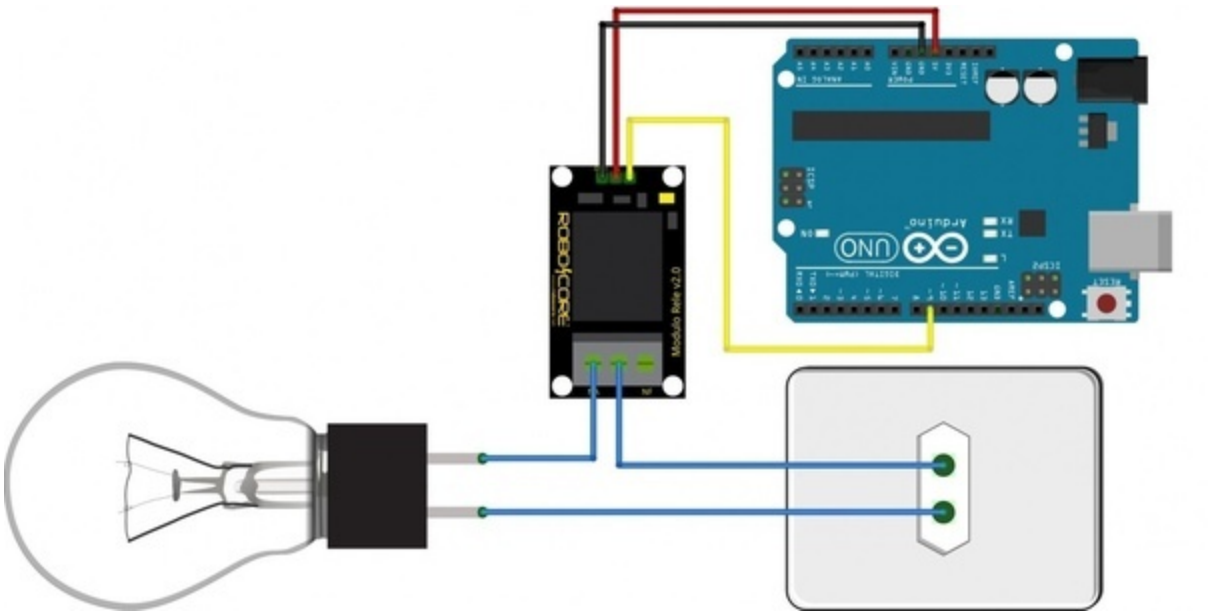


كما ذكرنا فالأردوينو يعمل على إصدار إشارات كهربائية على أطرافه. لكن هذه الأطراف لها قدرة كهربائية محدودة. يمكنك تشغيل ضوء بسيط LED من الطاقة الصادرة من الأردوينو ، لكنك بالتأكيد لن تتمكن من تشغيل محرك بهذه الطاقة البسيطة (5v وحوالي 40mA) لذا فكثيرا ما نحتاج لدوائر التكبير.

سوف نبدأ بشرح الطريقة الأسهل (استخدام دائرة مرحل) ثم سنذكر طرق متقدمة إلكترونياً استخدمها إذا أحببت ، و تجاهلها إذا أحببت 😊

الطريقة الأسهل هي شراء دائرة مرحلات مخصصة للأردوينو Arduino Relay Module
ستجد مقاسات مختلفة تناسب التطبيقات المختلفة (مرحل واحد ، 2 ، 4 ، 8)





لاحظ في الصورة : الأردوينو يتصل بدائرة المرحل بثلاثة أسلاك (+5v GND ومخرج رقمي)
الآن بإمكانك بكل بساطة التحكم بالأجهزة الكهربائية المنزلية بواسطة الأردوينو.

تنبيه _ كل مرحل له طاقة تحمل (جهد و تيار) تأكد أن المرحل الذي تستخدمه يتحمل تشغيل الجهاز الذي تريد تشغيله ... مثلا بعض كثير من المرحلات تتحمل 5A بينما المكيف يستهلك 15A !!..

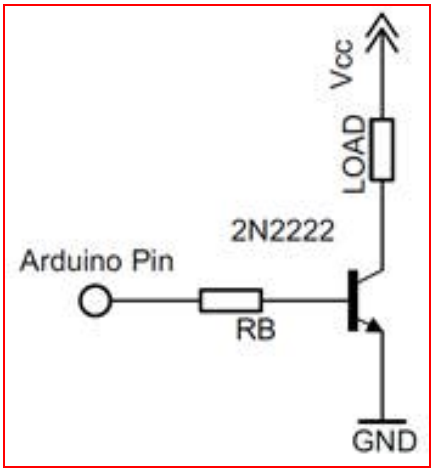
التكبير باستخدام الترانزيستور: Transistor

الطريقة السابقة سهلة وجميلة لا شك . سوى أنها قد لا تكون الأنسب في بعض التطبيقات. لأسباب مثل:

- **دائرة المرحل السابقة كبيرة الحجم نسبياً ، واستهلاكها للطاقة عالي نسبياً أيضاً**
- **ايجاد دائرة المرحل قد يكون صعب أحياناً فهي لا تتوفر في محلات قطع الغيار الإلكترونية عادة.**
- **منظر الدائرة يبدوا كدائرة مصنعة مسبقاً وأنت لم تقم بتصنيع الدائرة و إنما توصيلها فقط.**

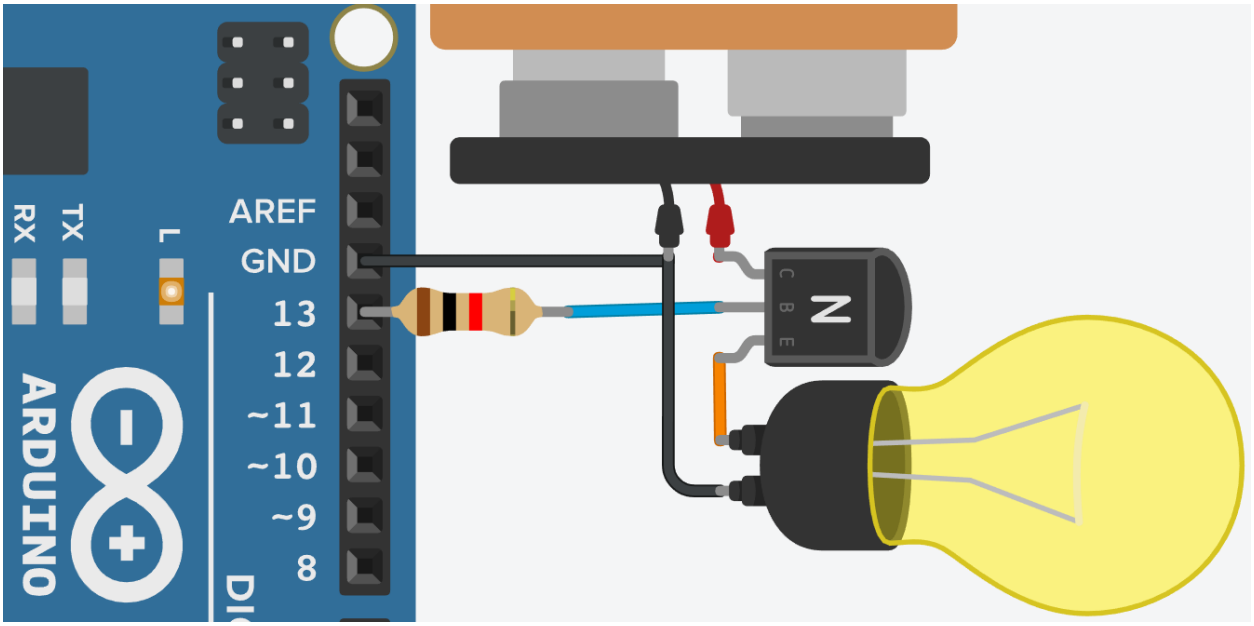
لهذه الأسباب في بعض الأحيان يستحسن أن تصنع دائرة تكبير خاصة بك . ولبعض التطبيقات ستكون سهلة جداً و مكونة من عنصرين فقط.

الترانزيستور عنصر صغير و هام في الدوائر الإلكترونية و من الممكن أن تـُضيع الكثير من الوقت و أنت تتفحص مئات بل آلاف الأنواع من الترانزيستورات. طريقة عملها وخصائصها .
الحقيقة؛ أنت لن تحتاج لدراسة كل شيء عن الترانزيستورات .
الترانزيستور يمكنه ببساطة تكبير الإشارة الخارجة من الأردوينو .
انظر الشكل:--

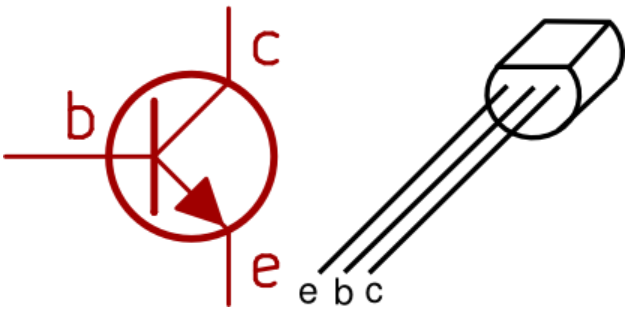


للترانزيستور 3 أطراف (E B C)
وطريقة استخدامه بسيطة.
1-وصل مخرج الأردوينو إلى طرف الـ B عبر مقاومة (نقترح 2.2K)
2-استخدم بطارية خارجية كمصدر طاقة. أو استخدم المنفذ vcc لتغذية الحمل (الشيء المراد تشغيله)_
vcc للأردوينو يتمكن من إخراج 200mA

3-وصل الموجب للحمل (محرك مثلا) والطرف الثاني للمحرك وصله بالطرف C في الترانزيستور.
4-وصل الطرف E بالأرضي 0v للبطارية و للأردوينو. (انتهينا 😆)



الترانزيستورات أنواع كثيرة جداً ... سوف ننصحك بنوعين شائعين.



ترانزيستور 2n2222

ويتمكن من توصيل تيار يصل إلى

IC=600mA

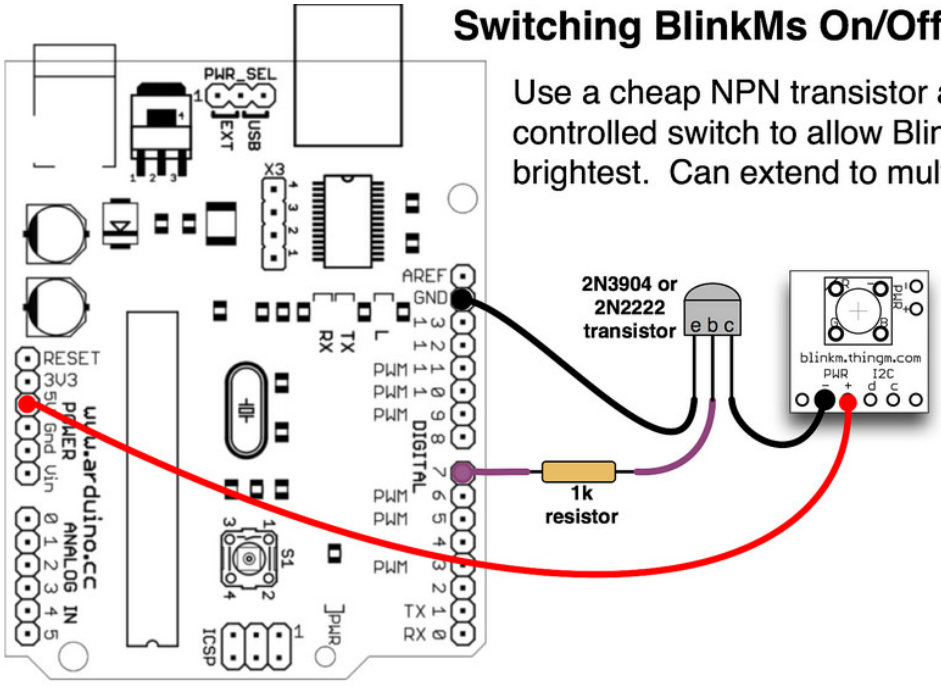
مقارنة بـ 40mA فقط يتمكن منفذ الأردوينو من إخراجها.

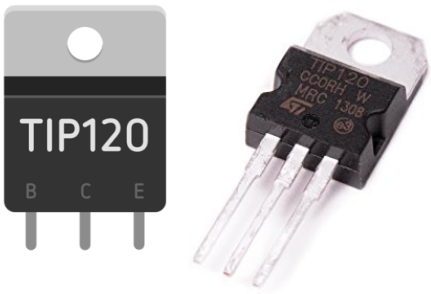
عادة نوصل مقاومة RB=1K لضبط قيمة التيار من الأردوينو. شاهد الرسم

تمرين : شغل لمبة DC من كشاف (9v) بواسطة الأردوينو + ترانزيستور للتكبير.

Switching BlinkMs On/Off with Arduino

Use a cheap NPN transistor as Arduino-controlled switch to allow BlinkM to be its brightest. Can extend to multiple BlinkMs.



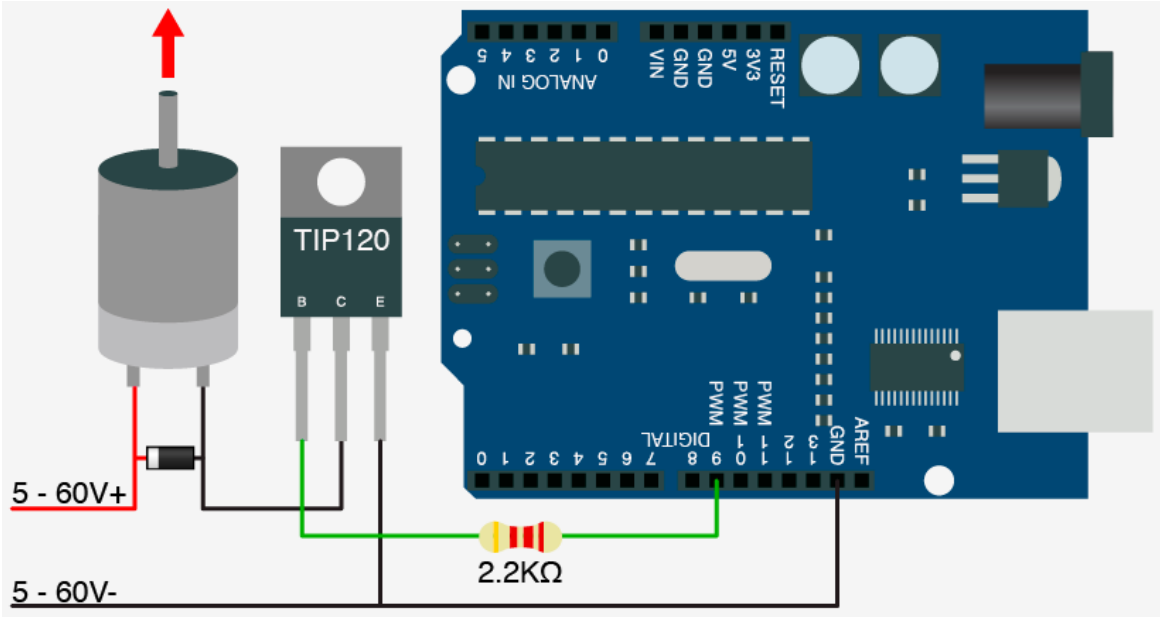


ترانزيستور TIP120

هذا الترانزيستور أكبر و يتحمل طاقة أعلى
يتحمل مرور تيار عالي IC=5A

لاحظ في بعض التطبيقات يجب تركيب مشتت حرارة على الترانزيستور.

لاحظ الرسم التالي يوضح كيفية تشغيل محرك بالأردوينو . مع استخدام TIP120 كمكبر للتيار.



ينصح بوضع (دايود) بين طرفي المحرك كما يظهر بالصورة هذا يعمل على حماية الدائرة

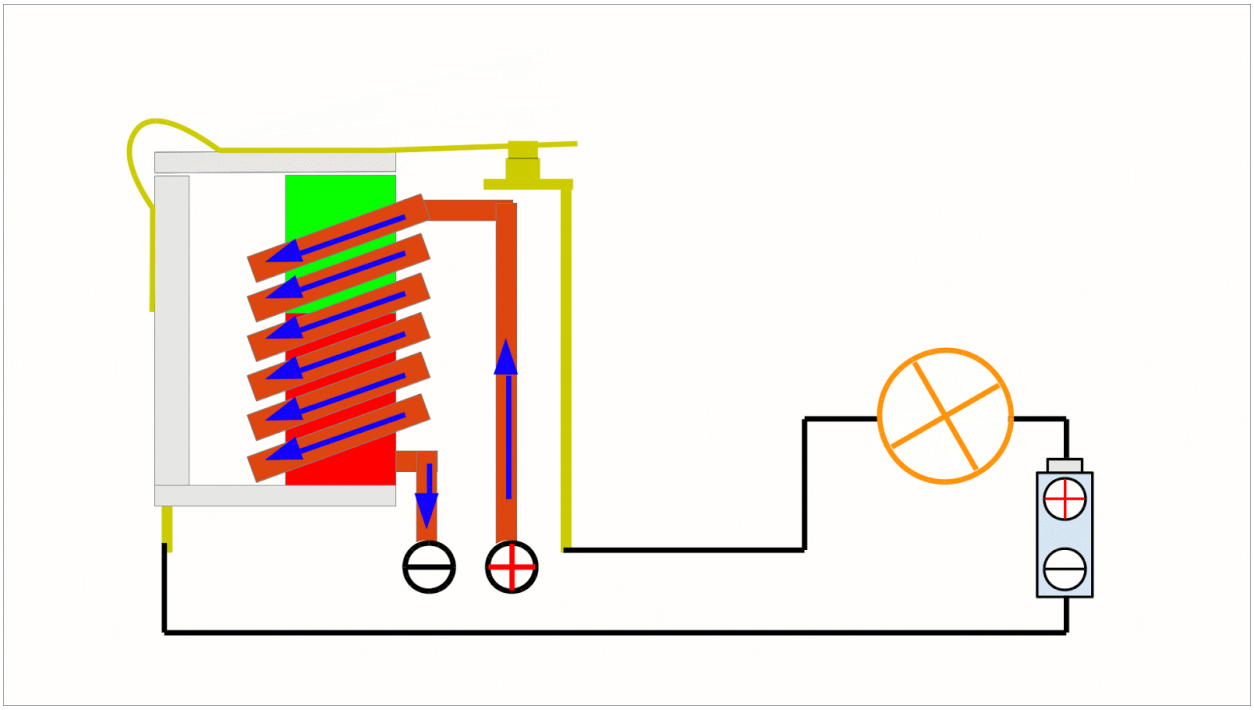
المرحلات :: Relay

يوجد عيبين في تكبير الطاقة باستخدام الترانزيستور .

- **السبب الأول أن الترانزيستور يتيح تشغيل الأجهزة التي تعمل على الطاقة المستمرة فقط DC بينما معظم أجهزة المنزل تعمل على الطاقة المترددة AC.**

(الـ DC والـ AC) هما نوعان من الكهرباء شرحنا الفرق بينهما في كتاب اساسيات الكهرباء باختصار شديد كهرباء المنزل من نوع AC المتردد ، بينما كهرباء البطاريات والشواحن DC مستمرة.

نظرية عمل المرحل بسيطة ، ولن نشرحها في هذا الكتاب . لكن ببساطة المرحل عبارة عن مفتاح يتم توصيله و غلقه بالكهرباء و ليس يدوياً . شاهد الصورة

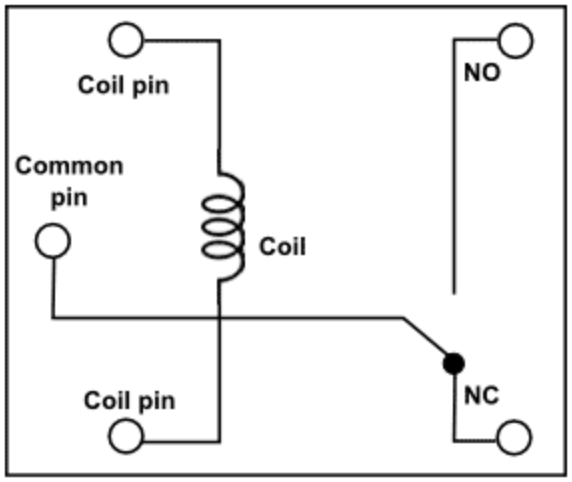


هذا يجعل المرحل (الريلاي) يتمكن من تشغيل الأجهزة سواء كانت تعمل بالكهرباء المستمرة DC أو الكهرباء المترددة AC

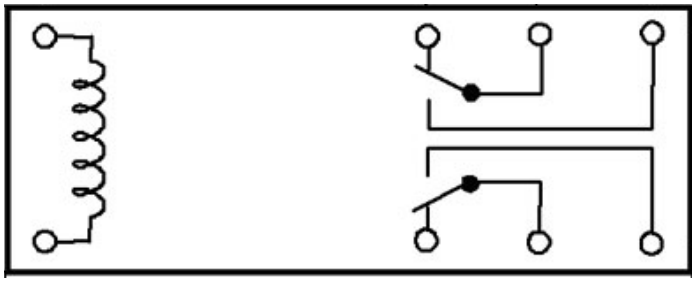
- **السبب الثاني أن المرحل (relay) يتحمل مرور تيار أعلى بكثير (مثلا 10A) من الترانزيستور عادة.**

المرحل العادي SPDT وتكون عادة 5 أطراف

3 أطراف للمفتاح : هي C و NC و NO في معظم التطبيقات سنستخدم C,NO فقط وللتحكم بالمفتاح سندخل إشارة التحكم إلى طرف من أطراف الملف Coil والطرف الآخر للملف يتصل بالأرضي.

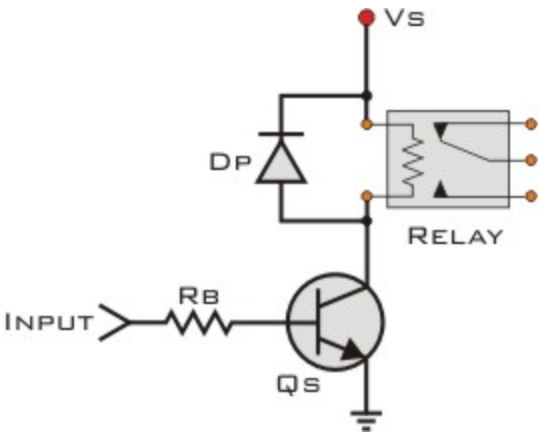


المرحل المزدوج DPDT و يكون له عادة 8 أطراف



يشبه عمل المرحل السابق سوى أنه يحتوي مفتاحين بدل مفتاح واحد. وهذا مفيد في بعض التطبيقات.

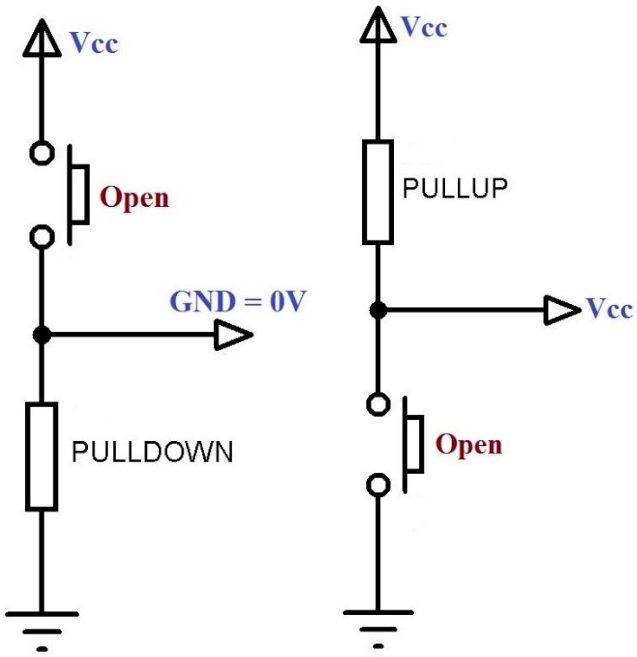
يسحب المرحل الصغير عادة 50mA وهذه القيمة نسبيا عالية على مخرج الأردوينو.
بإمكانك توصيل ملف المرحل إلى الأردوينو مباشرة ولكنني لست مسؤول إذا تلف الأردوينو بسبب سحب التيار العالي.
الحل الأسلم هو تشغيل المرحل بواسطة ترانزيستور (مثلا 2n2222)



نصيحة: يستحسن إضافة دايود بين طرفي الملف ، هذا يعمل على حماية الدائرة.
(انتبه على اتجاه الدايود يجب أن يكون الآنود متصل مع جهة الأرضي)

مقاومة رفع الجهد و مقاومة خفض الجهد Pull down , pull-up

بعض الأشياء تعمل بطريقة أكثر تعقيدا مما يظهر عليها من الوهلة الأولى . وقد تكون مقاومات الرفع و مقاومات الخفض أحدها .



عندما نريد إدخال إشارة رقمية إلى الأردوينو فإننا نستخدم مفتاح ضاغط push button لكن الملاحظ أنه في حالة عدم الضغط على الزر فإن الجهد يكون غير واضح القراءة و قد يعطي نتائج خاطئة.
ينصح باستخدام أحد هاتين الطريقتين بإضافة مقاومة (تقريباً 10K)

ملاحظة: يمكن الاستعاضة عن مقاومة رفع الجهد باستخدام الأمر

```
pinMode(10,INPUT_PULLUP)
```

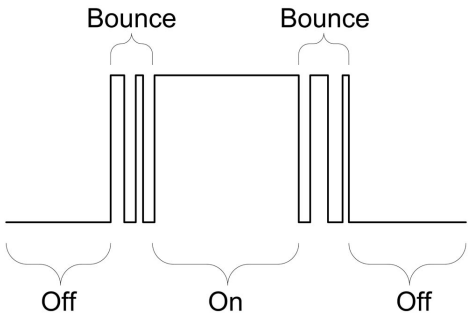
بدلاً من :

```
pinMode(10,INPUT);
```

الأمر PULLUP يعمل على توصيل مقاومة رفع داخلية. و تغني عادة عن مقاومة الرفع الخارجية.

ملاحظة:
قد لا يعمل هذا الأمر بشكل جيد مع الطرف 13 بسبب وجود LED متصل بالمنفذ 13 على البورد.

تـذبذب إشارة الدخل Bouncing

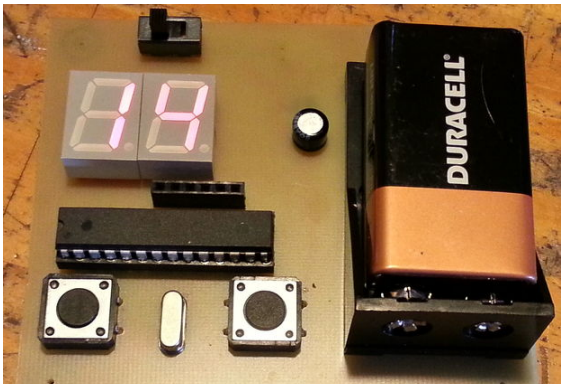


عندما تضغط على زر رقمي فإنك تتوقع أن يتغير الجهد من 0 إلى 5v بشكل واضح ودقيق. الحقيقة أن هذا قد لا يحدث كما تتمنى بسبب ظاهرة بسيطة تسمى Debouncing انظر الشكل.

لو أحببت تنفيذ مشروع يستخدم مفتاح واحد ضاغط لتشغيل و إطفاء الجهاز (وليس مفتاحين) فإنك ستعاني بسبب هذه الخاصية. لأن الضغطة الواحدة ستظهر بشكل عدد عشوائي غير محدد من الضغطات السريعة. وقد تتغير حالة الجهاز كما تحب و قد لا تتغير 😟 . شاهد هذا الكود الذي يحل المشكلة بتأخير زمني.

```
bool x=0;  //هذا المتغير سيحمل حالة الضوء لاحقا
void setup(){
  pinMode(13,OUTPUT);
  pinMode(2,INPUT_PULLUP);}
void loop(){
  while(digitalRead(2)==1){}//إيقاف البرنامج بانتظار ضغطة
  x= !x; //تبديل حالة إكس
  digitalWrite(13,x); //اخراج إكس على المنفذ 13
  delay(500);} //هذا التأخير يعمل على حل المشكلة
```

أيضاً لو كنت تعمل على تنفيذ عداد إلكتروني فإنك تتوقع أنه كلما ضغطت على الزر فإن العد سيزيد 1 فقط ... ولكن بسبب مشكلة الـ تذبذب bouncing الغير مرغوب بها . فقد تجد أن العداد يعد 3 أو 5 عدات بدل واحدة عند كل ضغطة على الزر.



يمكنك أن تحل هذه المشكلة بأكثر من طريقة _ أسهلها هو التأخير الزمني بعد كل حافة صاعدة.

**إضافة حل مشكلة الديباونس بمقاومة و مكثف.

```
int  x=0;
void setup(){
  Serial.begin(9600);
  pinMode(2,INPUT_PULLUP);}//منفذ2 متصل بالمفتاح
void loop(){
  while(digitalRead(2)==1){}
  x++;
  Serial.println( x );
  delay(500);} //التأخير هنا يعمل على تلافي المشكلة
```

الطريقة السابقة مفيدة سوى أنه توجد طريقة (أفضل من الناحية البرمجية)

تنفيذ العمل بدون استخدام الأمر (delay)

هل تتذكر الكود الذي شرحناه سابقا (الوميض بدون استخدام الأمر delay) الفكرة مشابهة كثيراً.

تشغيل و إطفاء بزر واحد مع حل مشكلة التذبذب (مع تأخير زمني صغير جدا)

فكرة الكود كتابياً :- نحتاج 3 متغيرات : -حالة الضوء - حالة المفتاح -الحالة السابقة للمفتاح.

إذا تغير حال المفتاح من 1 إلى 0 ؛ غير حالة الضوء

إذا استمر المفتاح على حالته سواءً 0 أو 1 لا تغير حالة الضوء.

إذا تغيرت حالة المفتاح من 0 إلى 1 لا تغير حالة الضوء.

بعد كل تغير على حالة المفتاح نحتاج لتأخير زمني صغير لنتجاوز فترة التذبذب

```
bool  LedState=0;
bool  buttonState=1;
bool  lastButtonState=1;

void setup(){
  pinMode(2,INPUT_PULLUP);
  pinMode(13,OUTPUT); }
void loop(){
  buttonState=digitalRead(2);
  if(buttonState==0 && last buttonState==1){
    LedState= !LedState;
    digitalWrite(13,LedState);
    lastButtonState=buttonState;
    delay(50);}
  else if(buttonState==1 && lastButtonState==0){
    lastButtonState=buttonState;
    delay(50);} }
```

مميزات الكود (السابق) :

1- عدم التوقف عند الأمر delay ماعدا وقت قصير جداً (50ms) فقط للابتعاد عن وقت التذبذب

2- يمكنك تبديل الحالة بسرعة ولا يلزمك الانتظار لنصف ثانية كل مرة.

3- إذا استمريت في الإمساك بالزر لن يستمر الضوء في تبديل حالته ، بل سيبقى على حالته حتى ترفع يدك و تضغط مرة ثانية.

عداد تصاعدي مع حل مشكلة التذبذب مع تأخير زمني صغير جداً

فكرة الكود كتابياً :- تشبه فكرة الكود السابق يجب أن يوجد متغيرين (حالة المفتاح ، حالة المفتاح السابقة) و حسب المقارنة بينهما يتم العد .

```
bool SW=1;
bool LSW=1;
int counter=0;
void setup(){
  pinMode(2,INPUT_PULLUP);
  Serial.begin(9600); }
void loop(){
  SW=digitalRead(2);
  if(SW==0  && LSW==1){
    counter++;
    LSW=SW;
    delay(50);
    Serial.println( counter ); }
  else if(SW==1 && LSW==0){
    LSW=SW;
    delay(50);}  }
```

مميزات هذا الكود عن العداد السابق:-

1- لا يستخدم التأخير الزمني إلا وقت قصير جداً

2- إذا أمسكت على الزر فإن العد لا يزيد بل ينتظر أن ترفع يدك و تضغطها مرة أخرى.

3- بإمكانك أن تضغط ضغطات سريعة و سيعد معك ، لن ينتظر نصف ثانية بين كل عدة.

إذا أحببت يمكنك استخدام مكتبة من الأوامر مخصصة لحل مشكلة التذبذب : زر الرابط هنا

تمرين1	استخدم مفتاح الضاغط كمفتاح تشغيل و إطفاء ON/off انتبه لمشكلة الـ Debounce
تمرين2	استخدم المفتاح الرقمي للعد التصاعدي على شاشة الـ Serial ، انتبه لمشكلة الـ Debounce

مسجلات الإزاحة shift register

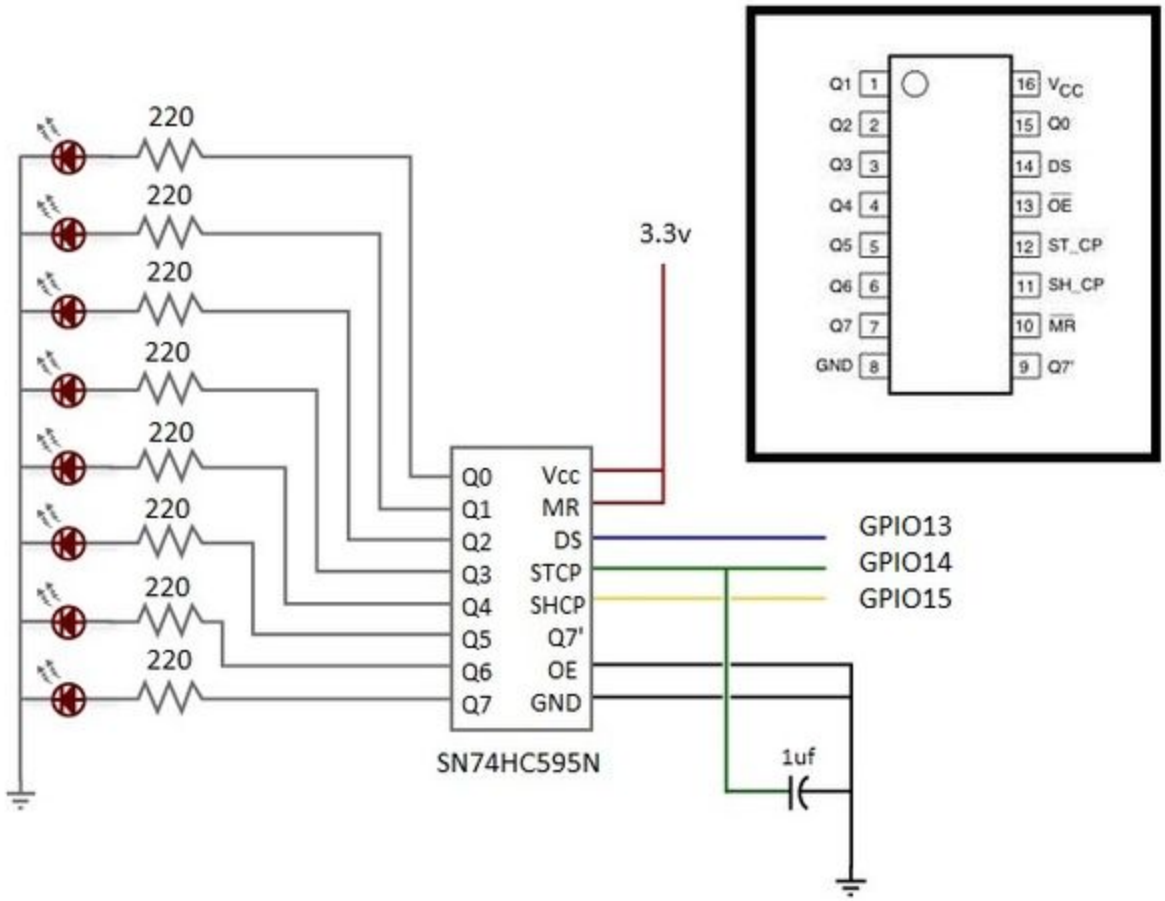
نموذجاً : 74HC595

مسجلات الإزاحة هي شرائح إلكترونية تعمل بطريقة معينة (لن نشرح طريقة عملها بالتفصيل هنا) سوى أنها مفيدة جدا في بعض تطبيقات الأردوينو.

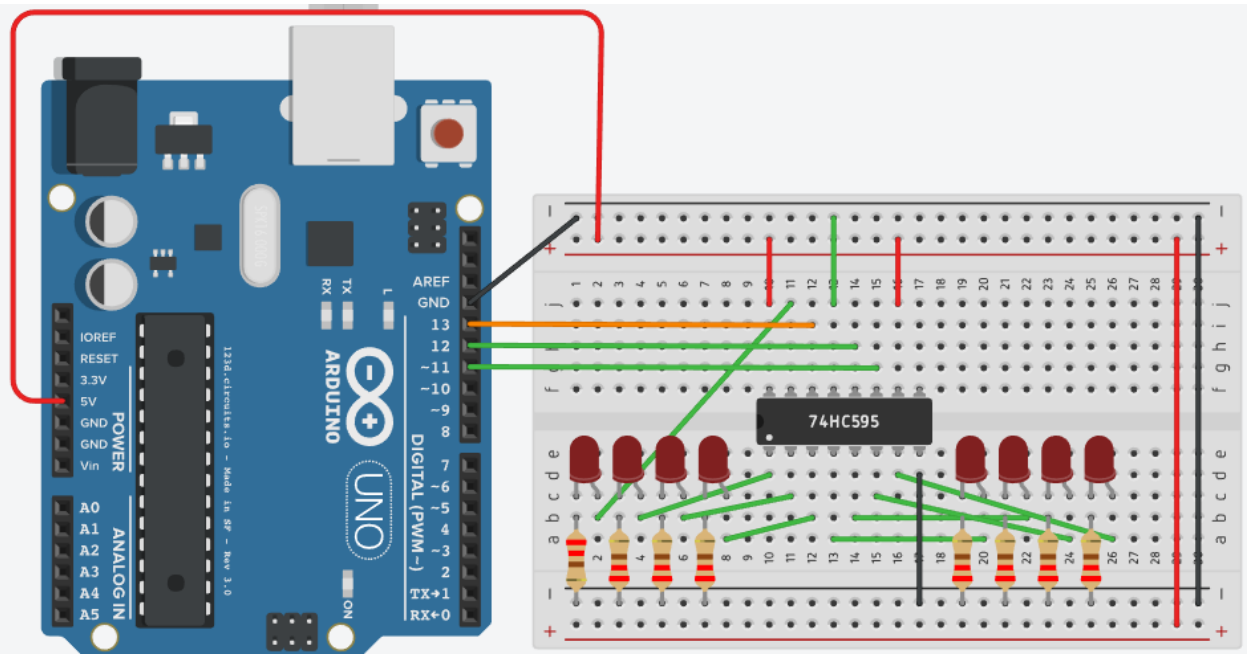
تخيل مثلا إشارة مرور بها 4 اتجاهات _ كل جهة فيها 5 إضاءات (3 للسيارات و 2 للمشاة) المطلوب هو 20 مخرج رقمي ! بينما الأردوينو أونو يحتوي 14 منفذ رقمي فقط !

بالتأكيد يمكنك شراء نوع أكبر من الأردوينو (أردوينو ميغا مثلا) لكن هناك حل أسهل عادة و هو مسجلات الإزاحة.

أحد أشهر أنواع مسجلات الإزاحة هي الشريحة 74HC595 و فيما يلي توضيح أطرافها:



تحتوي الشريحة 74HC959 على 3 أطراف أساسية يجب توصيلها لمخارج الأردوينو DS لنقل البيانات ، STCP يجب أن يكون 0 أثناء نقل البيانات ثم 1 لعرض البيانات ، و SHCP لضبط التوقيت clk انظر للدائرة التالية و الكود . و أعتقد أنك ستفهم ما تحتاج.

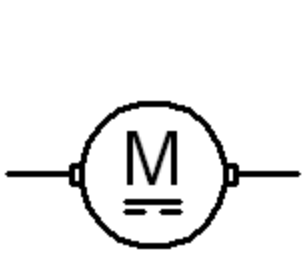


كود تجريبي يعرض القيمة (11001010) على مخرج المسجل.

```
int latchPin = 12;    //STCP
int clockPin = 11;    //SHCP
int dataPin = 13;     //DS
void setup() {
  pinMode(latchPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
  digitalWrite(latchPin, LOW);
  shiftOut(dataPin, clockPin, LSBFIRST,0b11001010);
  digitalWrite(latchPin, HIGH);}
void loop() {}
```

تمرين	لدينا مسجل إزاحة 74HC595 ونود أن يعرض 3 أعداد ثنائية بتتابع يغيرها المستخدم أعلى الكود.
تمرين	لدينا 3 مسجلات إزاحة ، ونود عرض أرقام ثنائية عليها.

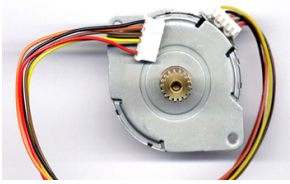
محركات دي سي _ DC motors



توجد أنواع عديدة من المحركات ؛ أشهرها وأبسطها هو محرك (التيار المستمر) DC-motor . يتميز محرك دي سي بأن له سلكين فقط . و عند تطبيق الجهد المناسب عليهما فإن المحرك سيبدأ بالدوران ، و عند عكس قطبية الجهد (الموجب و السالب) سوف ينعكس اتجاه الدوران.

ستجد محرك دي سي في العديد من الأجهزة الكهربائية (مكيف ، مسجل ، داخل الكمبيوتر...) . وفي كثير من التطبيقات ستجد المحرك متصل معه علبة تروس تجعل الحركة أبطأ و لكنها أقوى كثيرا .

سنتحدث في أجزاء متقدمة من الكتاب عن أنواع أخرى :

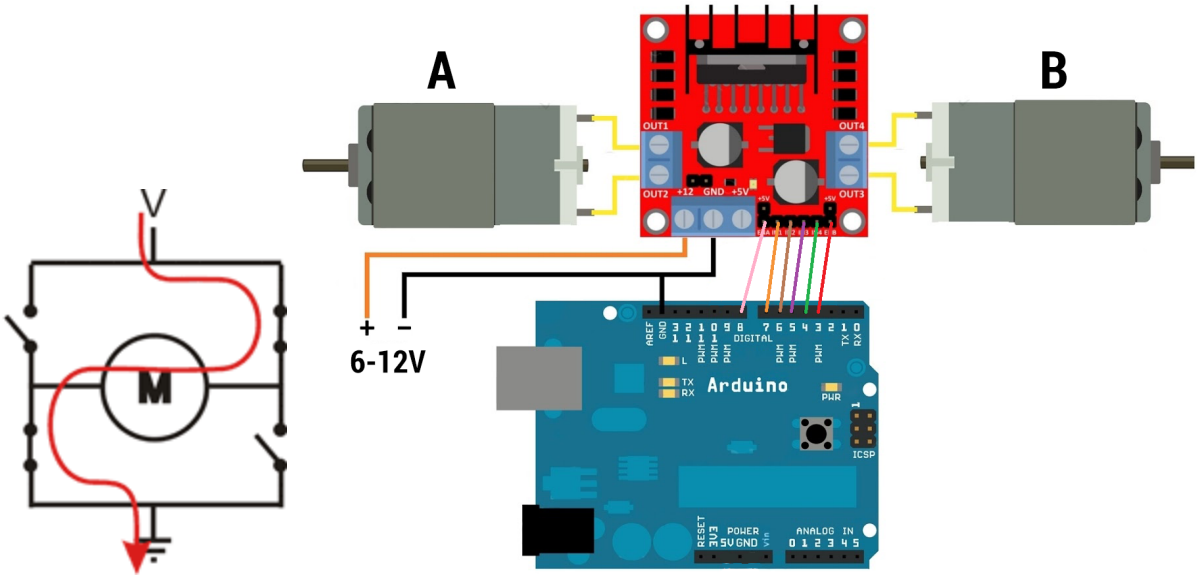
محرك سيرفو	servo motor	له ثلاثة أسلاك	
محرك خطوة	stepper motor	له خمسة أسلاك عادة	
محرك خطي	solenoid	له سلكين لكنه يتحرك بشكل خطي وليس دوراني	

أهم الملاحظات عند تشغيل محرك دي سي هو سحبه لتيار (أمبير) عالي (أعلى من قدرة الأردوينو) لذا نستخدم طرق مختلفة لتكبير التيار (والجهد أحيانا) : ترانزيستور قدرة ، مرحل (ريلاي) ، أو متحكم H

في محرك دي سي عادي قسنا التيار فكان
في التشغيل الحـُر(بدون حمل) : 50mA
في حالة الحمل الأقصى: 700mA

الطريقة الأسهل هي استخدام دائرة متحكم بالمحرك دي سي : وتعرف عادة بـ H-Bridge فقط تأكد من خصائص دائرة التحكم أنها تمد تيار كافي لسحب التيار المتوقع.

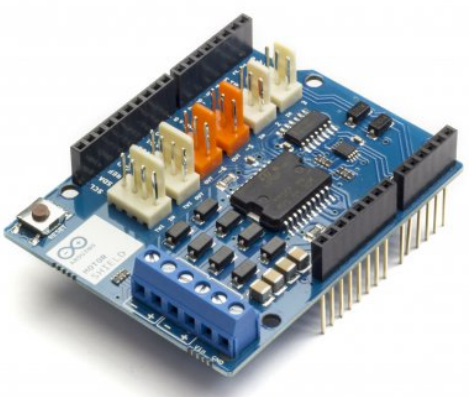
الموديل في الصورة **L298N** يتمكن من تشغيل محركين دي سي ويتحمل تيار يصل إلى 3A



IN2	IN1	في الرسم IN1 متصل مع 7 و IN2 متصل مع 6
0	0	لن يدور المحرك الأول
1	0	سيدور المحرك الأول مع عقارب الساعة
0	1	سيدور المحرك الأول عكس عقارب الساعة
1	1	لن يدون المحرك الأول

المنفذين **IN3** و **IN4** يتحكمان بالمحرك B
المنافذ **ENA** و **ENB** (متصلة مع 3 و 8) خاصة بالتحكم بالسرعة لكنها لا تستخدم عادة

بالتأكيد توجد موديلات مختلفة من متحكمات المحركات _ لذا ينبغي فهم خصائصها قبل استخدامها.
إذا أحببت تصنيع الدائرة الإلكترونية فإن من أشهر العناصر الإلكترونية (الشرائح) المخصصة للتحكم بمحركات دي سي هي الشريحة **L293D** ولكن الكتاب لن يتسع لنشرحها هنا.



ARDUINO MOTOR SHIELD REV3

بورد (دائرة إلكترونية) تم تطويرها من أردوينو لتتمكن من تشغيل عدة أنواع من المحركات .

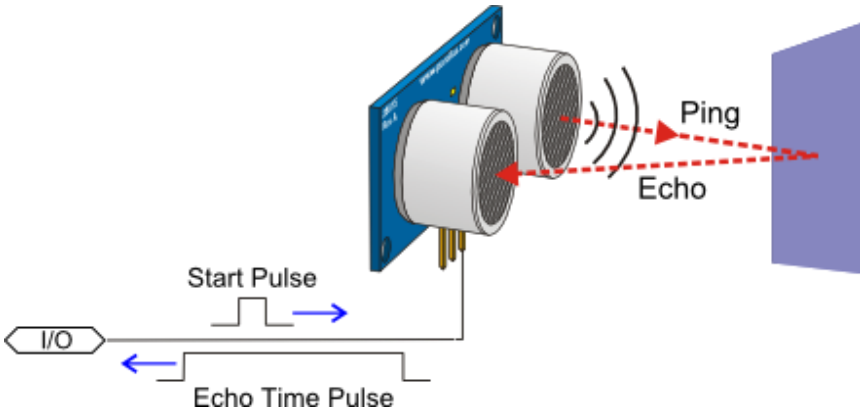
صندوق التروس Gearbox :

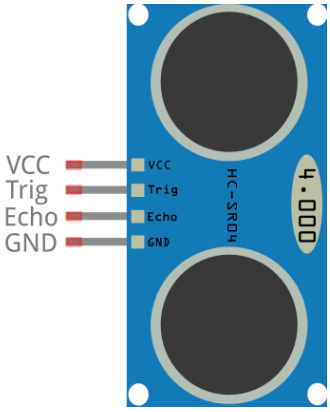
تعمل التروس على إعطاء قوة أعلى بتقليل السرعة.
وتختلف التروس من ناحية النسبة بين دورات المحرك الداخلة و دورات الذراع الخارجة من صندوق التروس. تكون العلاقة مثلاً 20:1 أي 20 دورة من المحرك = دورة واحدة من مخرج التروس
أو 30:1 يعطي سرعة أقل و قوة أعلى.

تمرين : شغل محرك دي سي بواسطة الأردوينو ليدور في الاتجاهين _ استخدم H-bridge

Ping Sensor _ حساس المسافة

ULTRASONIC Distance sensor





هذا الحساس يعمل على قياس بعد الأجسام الكبيرة (مثل جدار أو لوح) عن الحساس بطريقة انعكاس موجة فوق صوتية. شاهد الرسم في الأعلى لفهم طريقة عمل الحساس .
بمعرفة سرعة الصوت ، يمكن حساب المسافة التي استغرقها الصوت للانعكاس.

الطرف trigger هو الدخل من الأردوينو للحساس و عادة نرسل نبضة H زمنها 2µs
الطرف Echo هو الخرج و ينصح بتوصيل مقاومة 1K معه.

لاستخدام الحساس يفضل الاستفادة من الأمر `pulseIn` وهو يعمل على قياس عرض النبضة القادمة من الحساس إلى الأردوينو بالمايكرو ثانية .

```
int x=pulseIn(10,HIGH);
```

السطر الماضي يقيس عرض (زمن الإشارة) القادمة من الحساس و يضع القيمة بالمايكروثانية في x
تذكر سرعة الصوت: **340m/s** و بعرفة الزمن و السرعة يمكن حساب المسافة.

للانتقال لصفحة تشرح أمر `pulseIn` : اضغط هنا

شاهد فيديو قصير تعريفي بالحساس

يوجد أنواع من هذا الحساس بـ 3 أطراف فقط ، ويجب إرسال النبضة (trigger)
ثم استقبال نبضة الـ (echo) مع نفس المنفذ ! يجب تحويل المنفذ من خرج إلى دخل !
شاهد الكود في الأسفل أو **شاهد الكود في المعمل الافتراضي**

```
unsigned long du; // du : is the duration of the echo
void setup(){
  Serial.begin(9600); }
void loop(){
  pinMode(3,OUTPUT);
  digitalWrite(3,LOW);
  delay(2);
  digitalWrite(3,HIGH);
  delayMicroseconds(2);
  digitalWrite(3,LOW);
  pinMode(3,INPUT);
  du=pulseIn(3,HIGH);
  Serial.println("\n");
  Serial.println(du);
  delay(3000);   }
```

تمرين : صمم جهاز يشبه حساس السيارة _كلما اقترب الحساس من جسم يصدر صوت متقطع أسرع

الباب الرابع: المكتبات LIBRARIES و C++

#include < ... > #define Object

المكتبات ليست إلا عدد من الدوال (functions) تمت إضافتها معاً في ما يسمى بالمكتبة لتسهيل الوصول لها حال الحاجة لها . المكتبات تستخدم لغة C (مثل الأوامر التي شرحناها سابقاً) لكنها في العادة تستخدم طرق برمجة متقدمة ضمن لغة C++ . في بعض الحالات تكون المكتبات ضرورية لتنفيذ أمر معين (مثل الكتابة على ذاكرة EEPROM أو SD card)

فبدل أن تكتب كل شيء و تفهم كل شيء ، أضف المكتبة ، استخدم الأوامر اللازمة . و استفد من خبرات المبرمجين السابقين الذين صمموا لك المكتبات.

أمثلة لدوال مضافة في برنامج الأردوينو ولا تحتاج لتضمين أي مكتبة:

pinMode(,) digitalWrite(,) max(,) tan( ) , tone(, ,) , delay( )

أمثلة لدوال Functions لن تعمل حتى تضيف المكتبات التي تتضمن هذه الدوال:

EEPROM.read( ) , motor.attach( ) , lcd.clear( )

س/ لماذا لا أستخدم الدوال مباشرة ؟ لماذا أحتاج لتضمين (إضافة) المكتبة أولاً ؟

عدد الدوال كبير جدا جداً ، بل كل يوم تظهر دوال جديدة لعمل أعمال معينة ، ليس من المنطقي تعريف البرنامج على جميع الدوال التي قد لا تحتاجها.

فالذاكرة محدودة والمطلوب تضمين المكتبات التي تحتاجها فقط.

يمكن تقسيم المكتبات إلى قسمين:

مكتبات رسمية معتمدة من شركة أردوينو Official Arduino Libraries

و مكتبات تم كتابتها من مبرمجين لا يتبعون لشركة أردوينو: community contributed libraries for Arduino

لا يمكنك القول أن هذه المكتبات أفضل من تلك ، فالعبرة بالحاجة. مثلا لو قامت شركة **adafruit** بتصنيع شاشة صغيرة مميزة يمكن تشغيلها بالأردوينو ، فإنها في الأغلب ستنشر على الانترنت مكتبة من الأوامر الخاصة بالتحكم بهذه الشاشة. هذه المكتبة ليست رسمية من شركة أردوينو لكنها ضرورية إذا أردت استخدام الشاشة الجديدة .

المكتبات كثيرة ولا يمكن شرحها جميعا في هذا الكتاب. سنشرح أشهرها و أهمها. لمعرفة المزيد من المكتبات. راجع الصفحة على موقع الأردوينو:

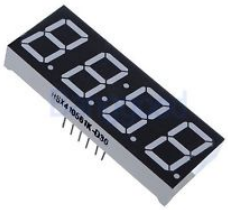
https://www.arduino.cc/en/Reference/Libraries	صفحة تحتوي المكتبات الرسمية المعتمدة من أردوينو
https://playground.arduino.cc/Main/GeneralCodeLibrary	صفحة تحتوي عدد كبير من المكتبات التي طورها المهتمين بالأردوينو في مختلف المجالات
https://playground.arduino.cc/Main/LibraryList	صفحة اضافية...

جدول يبين لك أهم المكتبات الرسمية للأردوينو مع شرح مختصر لاستخداماتها

EEPROM	القراءة و الكتابة على الذاكرة الدائمة EEPROM داخل الأردوينو
LiquidCrystal	للتحكم بشاشات LCDs
SD	للقراءة و الكتابة من ذاكرات SD
Servo	للتحكم بالمحركات من نوع سيرفو Servo motors
Stepper	للتحكم بمحرك خطوة stepper motor

** للاستزادة : اضغط على اسم أي مكتبة للانتقال لصفحة خاصة بها.

مكتبات هامة (غير رسمية) وسوف نشرح كل مكتبة في الصفحات القادمة

IRremote		لجعل الأردوينو يستقبل إشارات الريموت كنترول (تحت الحمراء)
Keypad		لجعل الأردوينو يتعامل بسهولة مع لوحة المفاتيح (الرقمية)
Sevseg		لتسهيل التحكم بشاشة السيفن سيقمنت لعرض الأرقام

ملاحظة: معظم المكتبات تستخدم خصائص لغة C++ ، لذا سنتعلم بعض خصائصها.
مثل : تضمين مكتبة ، تعريف كائن ، تحديد خواص الكائن، وتعريف ماكرو

فهم أوامر المكتبات : المكتبات عادة تساعدك كثيراً لتنفيذ العمل . لكن في استخدامها عيب . وهـو وجود عدد كبير جدا من المكتبات ، و في كل مكتبة عدد كبير من الأوامر (الدوال) وطرق استخدامها طرق مختلفة وليست دائماً بنفس طرق استخدام الأوامر المعتادة. كما أن المكتبات يتم تحديثها باستمرار عادة ، لذلك قد تستخدم أوامر جديدة أو قديمة (حسب المكتبة المضافة) مصادر تعلم المكتبات عادة :

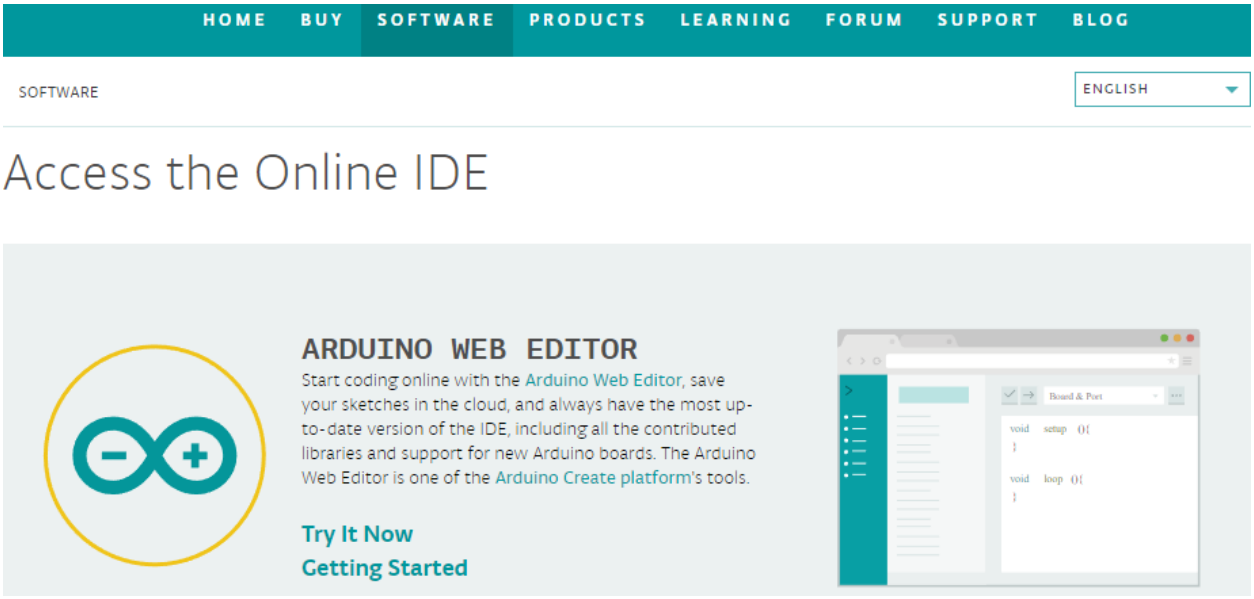
1- ادرس الأمثلة المرفقة مع كل مكتبة بعناية

2- ملف pdf بإسم documentation يأتي مع بعض المرفقات و عادة يكون فيه شرح مفصل

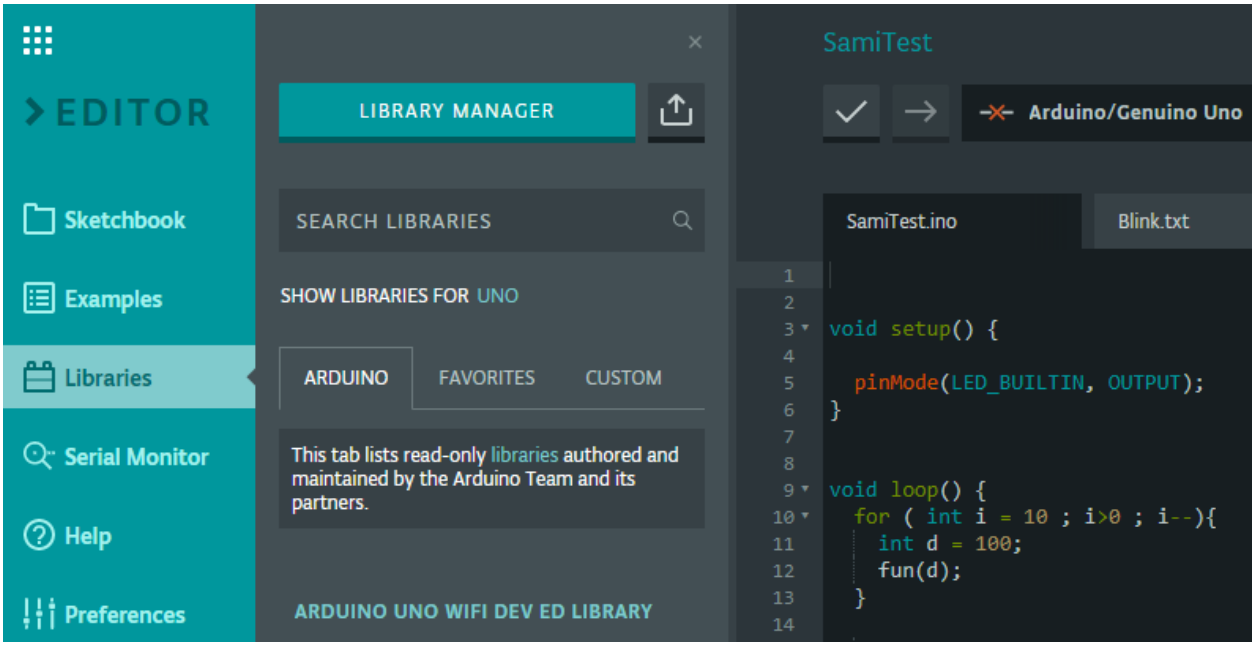
أحد الحلول للتعامل مع المكتبات الحديثة (الإصدار الحديث) هو البرمجة باستخدام **Arduino create** والذي شرحناه سابقاً.

1- اذهب إلى موقع الأردوينو الرسمي **Arduino.cc**

2- من الأعلى اختر القسم **software** ثم انقر على **Try it now**

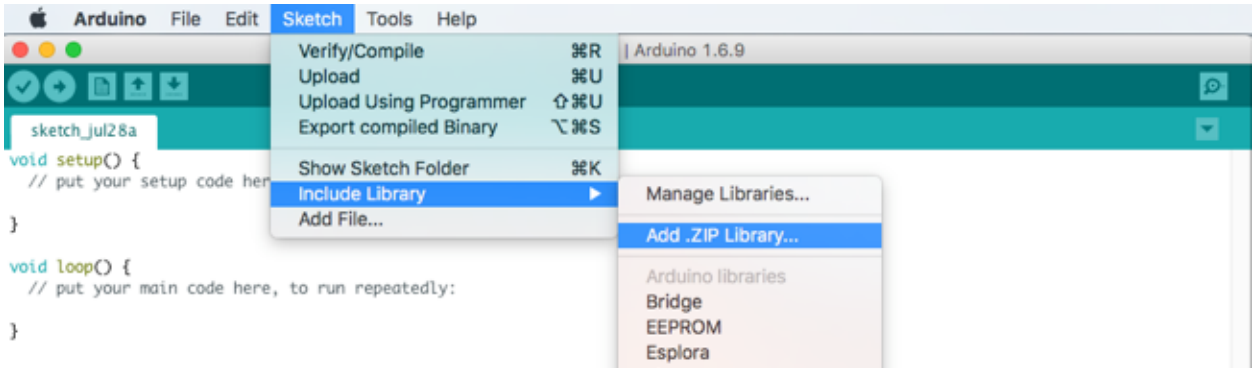


3-في صفحة Arduino Create ابحث عن المكتبة المطلوبة في Library manager و أضفها إلى المكتبات المفضلة.



4- بعد اضافة المكتبة للمفضلة . ستجد العديد من الأمثلة مرفقة مع المكتبة. شاهدها و اقرأ الشروحات والملاحظات المرفقة مع كل كود.

تضمين مكتبة including a library



هذا الموضوع قد يكون سهل جدا وقد يكون صعب 😛 في الحقيقة معظم الأشخاص الذين زاروني طلبا للمساعدة (لتنفيذ مشاريع الأردوينو الخاصة بهم) كانت مشكلتهم عدم إضافة المكتبة الصحيحة فقط !

فمثلاً عندما تكتب الأمر:

```
#include <LiquidCrystal.h>
```

في برنامج برمجة الأردوينو IDE فإنه سيتقبل إضافة المكتبة بدون مشاكل 👍

بينما لو كتبت الأمر التالي:-

```
#include <Debounce2.h>
```

فالأغلب أن البرنامج سيعيد لك رسالة خطأ . لماذا ؟!!

السبب أنه عند تثبيت البرنامج (Arduino IDE) فإن البرنامج قد أضاف بعض المكتبات الشائعة في مجلد البرنامج . ولتستخدم الأوامر يكفيك أن تكتب سطر واحد فقط في أعلى الكود . ثم تستخدم أوامر المكتبة. بينما لو كانت المكتبة غير موجودة في مجلد برنامج الأردوينو IDE فيجب إضافتها .

توجد أكثر من طريقة لإضافة مكتبة إلى برنامج IDE :-

- استخدام مدير المكتبات manage libraries للبحث و إضافة المكتبات
- إضافة مجلد مضغوط add .ZIP library
- تحميل المكتبة وفك ضغطها ثم وضعها في مسار مجلد الأردوينو (يجب أن يكون اسم المجلد مطابق لاسم المكتبة)

لإيجاد مسار المكتبات إذهب إلى **File >> preferences**

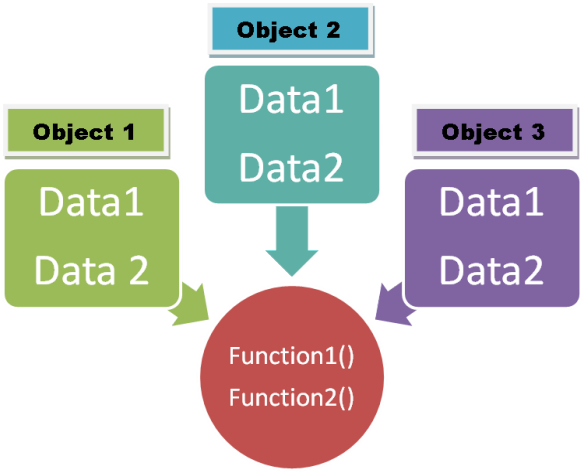
حسب تجربتي فاستخدام المبرمج على الانترنت Arduino Create أفضل و أسهل من استخدام البرنامج العادي. فهو يبحث عن آخر الإصدارات من الانترنت مباشرة.

Arduino.cc >> software >> Web editor (try now)

تمرين: قم بإضافة جميع المكتبات الموجودة في صفحتي : 112 و 113 في كود واحد

تعريف الكائنات Objects

Class's instance



الكائنات (objects) ليست إلا طريقة لتنظيم الأوامر وربطها مع مكونات النظام في لغة C++

مثلاً لديك شاشتين LCD أو 3 لوحات مفاتيح . أو 4 محركات. كل مكون يمكن التحكم به بطريقة:

1-ادراج المكتبة اللازمة include library

وقد شرحنا كيفية إدراج مكتبة في الصفحات السابقة

2-تعريف الكائن Object , instance

وهذا السطر يسمى **constructor** وفيه يجمع المعلومات اللازمة لتكوين الكائن Object

```
className   objectName = className(arguments);
className   objectName(arguments);//اختصار الأمر السابق
```

3-استخدام الدوال المطلوبة مرتبطة مع اسم الكائن

ويكون استخدام الدوال من المكتبة حسب البناء التالي:

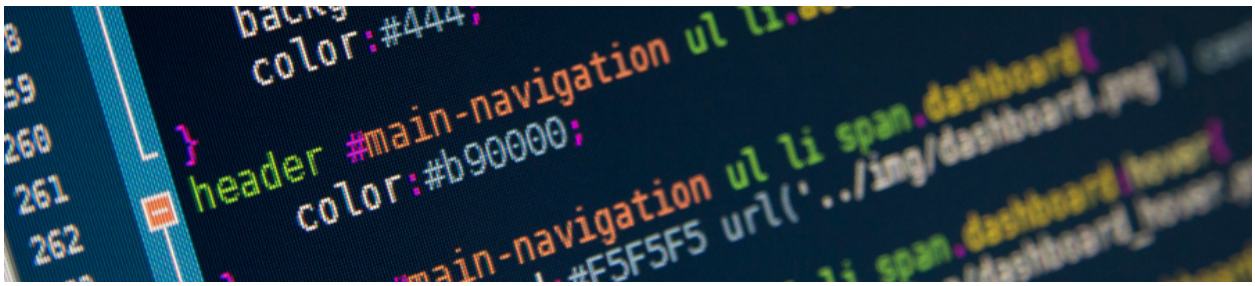
```
object.function(arguments);
```

هنا نضيف مكتبة محركات السيرفو للكود.	`#include <Servo.h>` `#include <LiquidCrystal.h>`

نعمل على تعريف كائن واسمه motor يمكننا تعريف كائن آخر من نفس المكتبة وهنا نعرف كائن lcd ، لاحظ بعض الكائنات تتطلب معطيات لتتعرف. في هذه الحالة: (RS,E,D4,D5,D6,D7)	`Servo  motor1;`
	`Servo  motor2;`
	`LiquidCrystal lcd(8,9,4,5,6,7);`

الان أوامر المكتبات ترتبط بالمكونات motor1 أو motor2 أو الشاشة التي أسميناها lcd	`motor1.attach(5);`
	`motor2.WriteMicroseconds(1300);`
	`lcd.begin(2,16);`
	`lcd.print("my name is Sami ");`

تعريف الماكرو __ #define :-



الماكرو هو كود جانبي (قصير عادة) يعمل مثل الدوال أحياناً .

في هذا الكتاب لن نتعمق في لغات البرمجة المتقدمة ولا في تعريف الماكرو .

لكن في الأردوينو نستخدم (`#define`) في أعلى الكود لربط إسم بقيمة عادة . فالبرنامج قبل **التنفيذ** سوف يستبدل العبارة اليسرى بالقيمة اليمنى مثلاً **(#define sam 10)**

في أي مكان من الكود عندما يوجد عبارة (sam) سوف يحذفها و يضع مكانها (10)

السطرين التاليين يقومان بنفس العمل . سوى أن الطريقة `#define` أوفر في استخدام الذاكرة.

```
#define  led   1
#define  led2  5
```

```
int  led  = 1;
int  led2 = 5;
```

فكلما كتبنا كلمة (`led`) في الكود فإن البرنامج سيحولها إلى القيمة (1) قبل رفعها للأردوينو

مثال: اعمل برنامج الوميض Blink و استخدم طريقة `#define` لاختصار الكود

```
#define ON   digitalWrite(13,HIGH)
#define OFF  digitalWrite(13,LOW)
#define DD   delay(500)

void setup(){ pinMode(13,OUTPUT);}
void loop(){ON; DD; OFF; DD; }
```

لوحة الأزرار keypad

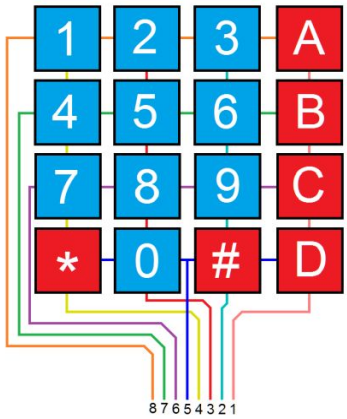
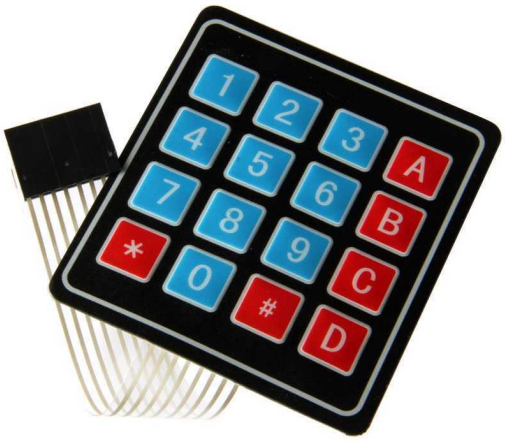


Figure 1: Matrix Keypad Connections



المفتاح (الزر) الرقمي طريقة ادخال شائعة و بسيطة. لكن في الكثير من التطبيقات نحتاج لإدخال معلومات أكثر (أرقام وحروف) للتحكم بتشغيل الأردوينو. ومن أشهر الحلول استخدام شبكة من الأزرار الرقمية Keypad تحتوي 16 زر لكنها تستخدم 8 منافذ رقمية فقط.

لمشاهدة صفحة تشرح المكتبة و الأوامر وتحتوي رابط التحميل اضغط هنا :
https://playground.arduino.cc/Code/Keypad

مثال1: يعرض الزر الذي ضغطته على شاشة السيريال:

```
#include <Keypad.h>   //تذكر أن حجم اللوحة 4*4
char Keys[4][4] = {   //[Rows]   [Columns]
  {'D','#','0','*'},
  {'C','9','8','7'},
  {'B','6','5','4'},
  {'A','3','2','1'}};
byte RP[4] = {6, 7, 8, 9};
byte CP[4] = {2, 3, 4, 5};
Keypad KP=Keypad(makeKeymap(Keys),RP,CP,4,4); // تكوين الكائن
void setup(){Serial.begin(9600);}
void loop(){
  char x = KP.getKey();
  if (x){Serial.println(x);}}
```

ملاحظات:

- أهم سطر يجب عليك فهمه هو الذي يبدأ بـ Keypad و يسمى سطر الـ constructor

في هذا السطر يتم تعريف البرنامج بكل خصائص لوحة المفاتيح المستخدمة : عدد الصفوف ، الأعمدة ، المنافذ المستخدمة للصفوف و الأعمدة و القيم (الرموز) التي توازي كل ضغطة زر .

- القيم التي تريد استخدامها عند الضغط على كل زر موجودة في المصفوفة (Keys) و يجب الانتباه أنك ستستخدم نوع واحد من البيانات :char في المثال السابق .
- يمكنك جعل عدد الأعمدة 3 إذا لم تستخدم الأحرف. ويجب تعديل الكود قليلاً.
- في العادة ستجد عدم توافق بين الضغطات و الناتج (مثلا تضغط (1) فيظهر (A) الأفضل التجربة ثم التعديل على المصفوفة (Keys) حتى تتوافق الضغطات مع الناتج على شاشة السيريال.

مثال 2: استخدم لوحة المفاتيح keypad بحيث تشغل LED بالضغط على (*) و تطفئه بـ (#)

أول 9 أسطر تتطابق مع التمرين السابق

```
void setup(){pinMode(13,OUTPUT);}
void loop(){
  char x = KP.getKey();
  if (x=='*'){digitalWrite(13,HIGH);}
  if (x=='#'){digitalWrite(13,LOW);}}
```

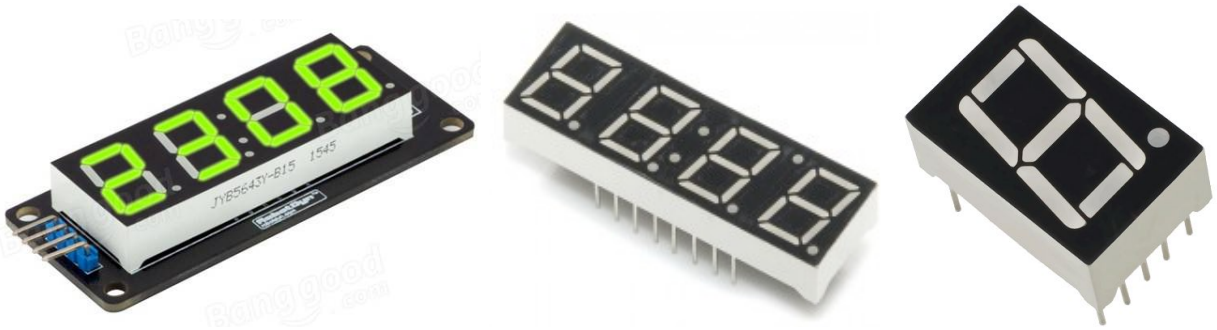
مثال3: استخدم لوحة المفاتيح لإدخال قيمة التأخير الزمني (ms) و تشغيل وميض Blink

أول 9 أسطر تتطابق مع التمرين السابق

```
int x1=0,x2=0,x3=0,x4=0,x=0;
void setup(){
  while(x1==0){x1=kp.getKey();}  //لتسجيل خانة الألوف
  x1=x1-48;            //لأننا نريد استخدام قيمة رقم وليس كود أسكي
  while(x2==0){x2=kp.getKey();}   //لتسجيل خانة المئات
  x2=x2-48;    //change ASCII code to number INT
  while(x3==0){x3=kp.getKey();}  //لتسجيل خانة العشرات
  x3=x3-48;
  while(x4==0){x4=kp.getKey();}   //لتسجيل خانة الآحاد
  x4=x4-48;
  x=x1*1000+x2*100+x3*10+x4;}   //تكوين الرقم النهائي من جميع الخانات
                    الباقي سهل ، فقط استخدم قيمة x كتأخير في كود وميض
```

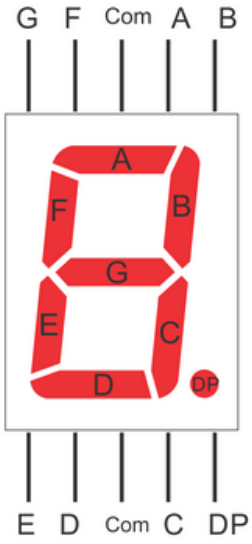
تمرين: استخدام لوحة الأرقام لإدخال قيمة التردد للأمر tone ثم تشغيل تنبيه صوتي بالتردد المدخل

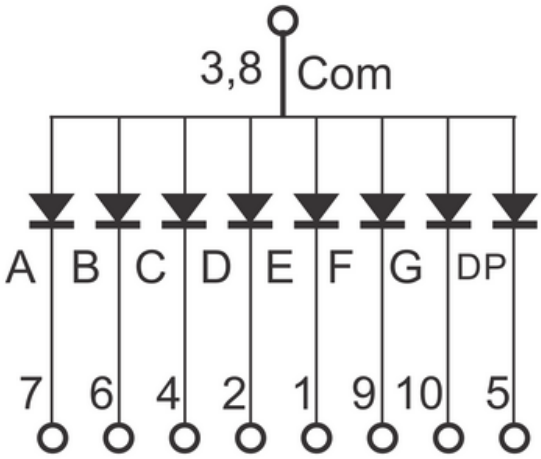
شاشة الأضواء السبعة Seven segment



الإشارات السبعة (seven segment) هي شاشة بسيطة في التكوين تعمل على عرض الأرقام. بالتأكيد رأيتها سابقا في الكثير من الأجهزة. فكرتها أنها تتكون من 7 أضواء منفصلة +1 للنقطة و يمكنك تشغيل أي ضوء تحب لكي يظهر الرقم المطلوب. وأحياناً توجد وحدتين 7seg أو أربع وحدات متصلة معاً.
الأنواع الموجودة في السوق تكون عادة أحد نوعين :

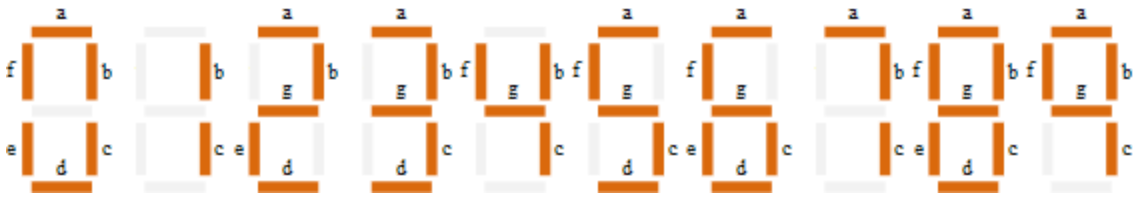
آنود مشترك	Common Anode	يجب أن ترسل 0v لتشغيل أي ضوء وهو الأكثر انتشاراً
كاثود مشترك	Common Cathode	يجب أن ترسل 5v لتشغيل أي ضوء وهو النوع الأسهل



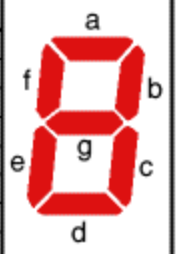


يجب التأكد بالتجربة من الجهد المناسب للتشغيل وفي حال زيادة الجهد عن المناسب يجب استخدام مقاومات لتقليل مرور التيار.

في البداية نود فهم كيف يتم عرض الأرقام حسب الجدول التالي:



Outputs from the 4026 counter and display driver IC								
Count	a	b	c	d	e	f	g	h
0	●	●	●	●	●	●		●
1		●	●					●
2	●	●		●	●		●	●
3	●	●	●	●			●	●
4		●	●			●	●	●
5	●		●	●		●	●	
6	●		●	●	●	●	●	
7	●	●	●					
8	●	●	●	●	●	●	●	
9	●	●	●	●		●	●	



● = segment on. h is used to drive other counters.

تخيل أن لديك متغير تزيد قيمته كل ثانية : 0 ، 1 ، 2 ، ..... 9 وتريد عرض هذه القيمة على سيفين سيقمينت. seven segment كيف سيكون الأمر باستخدام الأوامر المعروفة فقط (بدون مكتبة)
توجد أكثر من طريقة لكتابة كود يعمل على عرض الأرقام بالتسلسل على الـ سيفن سيقمينت. هذه أحدها.

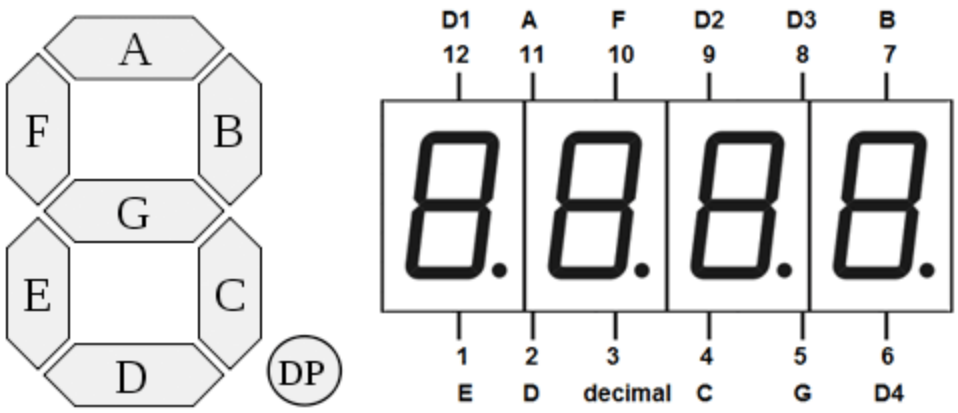
```
int A=2,B=3,C=4,D=5,E=6,F=7,G=8;
int x=0;  // x will be the counter
void setup(){ //we used for loop to make the code shorter
  for(int i=2;i<9;i++){pinMode(i,OUTPUT);}}
void loop(){
  for(int i=2;i<9;i++){digitalWrite(i,LOW);}
  digitalWrite(E,HIGH);//because E is mostly off unlike the others
  if(x==1 || x==4){ digitalWrite(A,HIGH);}
  if(x==5||x==6){ digitalWrite(B,HIGH);}
  if(x==2) {digitalWrite(C,HIGH);}
  if(x==1||x==4||x==7){ digitalWrite(D,HIGH);}
  if(x==0||x==2||x==6||x==8){ digitalWrite(E,LOW);}
  if(x==1||x==2||x==3||x==7){ digitalWrite(F,HIGH);}
  if(x==0||x==1||x==7){ digitalWrite(G,HIGH);}
  delay(1000); x++; if(x>9){x=0;} }
```

شاهد تشغيل الكود: اضغط هـنا

لاحظ : الكود السابق هو الكود الأقصر بدون استخدام مكتبة Library وطوله 30 سطر. تم تنفيذ عدة طرق لجعل الكود أقصر (وربما أصعب للفهم) هذا الكود يستخدم لتشغيل وحدة واحدة فقط.
شاهدت بعض المبرمجين يستخدمون كود طوله (96) سطر ... لتنفيذ نفس العمل.😆

بالتأكيد أن الكود طويل و صعب التتبع _ أيضا هذا الكود يشغل خانة واحدة وليس 4 خانات ! مشكلة.

لنفهم تكوين وطريقة تشغيل الـ seven segment ذات الأربع خانات شاهد الصورة:



هذا الكود يشغل 4 خانات (بدون مكتبة) لاحظ مستوى التعقيد !

```
int  x=1983;  //غير القيمة التي تريد عرضها هنا
int A=7,B=8,C=2,D=3,E=4,F=6,G=5;
int D1=10,D2=11,D3=12,D4=13;
byte x4=x%10;
byte x3=x%100 /10;
byte x2=x%1000 /100;
byte x1=x%10000 /1000;
int X[]={x4,x3,x2,x1};
void setup(){for(int i=2;i<=13;i++){pinMode(i,OUTPUT);}}
void loop(){
  for(int i=10;i<=13;i++){ digitalWrite(i,LOW);}
  static int CS=0; //CS is chip select must loop 1-4
  if     (CS==4){CS=0;}
  if     (CS==0){digitalWrite(D4,HIGH);}
  else if(CS==1){digitalWrite(D3,HIGH);}
  else if(CS==2){digitalWrite(D2,HIGH);}
  else if(CS==3){digitalWrite(D1,HIGH);}
  for(int i=2;i<9;i++){digitalWrite(i,LOW);}
  digitalWrite(E,HIGH);//because E is mostly off unlike the others
  if(X[CS]==1||X[CS]==4){ digitalWrite(A,HIGH);}
  if(X[CS]==5||X[CS]==6){ digitalWrite(B,HIGH);}
  if(X[CS]==2){ digitalWrite(C,HIGH);}
  if(X[CS]==1||X[CS]==4||X[CS]==7){ digitalWrite(D,HIGH);}
  if(X[CS]==0||X[CS]==2||X[CS]==6||X[CS]==8){ digitalWrite(E,LOW);}
  if(X[CS]==1||X[CS]==2||X[CS]==3||X[CS]==7){ digitalWrite(F,HIGH);}
  if(X[CS]==0||X[CS]==1||X[CS]==7){ digitalWrite(G,HIGH);}
  CS++;
  delay(20);}
```

شاهد تشغيل الكود هـــنـــا

لتسهيل العمل مع الإشارات السبع يمكننا استخدام مكتبة **Library** تحتوي الأوامر اللازمة.

شاهد الأسطر التالية التي تشرح تعريف الأردوينو +المكتبة على طريقة توصيل الـ7seg

```
#include <SevSeg.h>
SevSeg ss;
void setup(){
  byte numDigits = 4;
  byte digitPins[] = {2, 3, 4, 5};//مقاومة عند كل ديجيت بين
  byte segmentPins[] = {6, 7, 8, 9, 10, 11, 12, 13};
  ss.begin(COMMON_ANODE, numDigits, digitPins,segmentPins);
  ss.setBrightness(10);} //هذا الأمر غير ضروري لمعظم الشاشات
```

ملاحظة: يجب أن تتأكد من نوع الـ SevSeg ضع COMMON_CATHODE بدل COMMON_ANODE حسب نوع السيفين سيقمينت

```
void loop(){
  ss.setNumber(1234,3);  //should show 1.234
  ss.refreshDisplay();}  // Must run repeatedly;
```

ملاحظات : هذا النوع من التشغيل لا يقبل وجود أمر يوقف دورة التنفيذ (مثل delay) إذا أردت العد كل ثانية فاستخدم طريقة مثل (الوميض بلا تأخير والذي شرحناه سابقا) (المزيد عن المكتبة اضغط هنا)

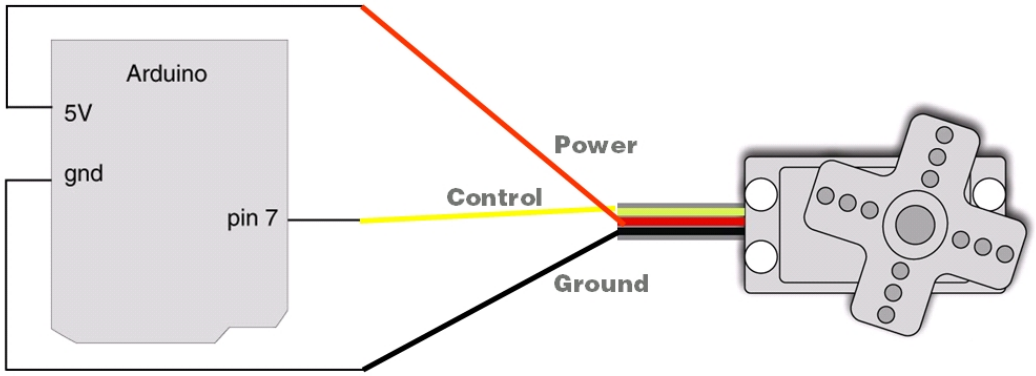
تمرين: صمم شاشة (سيفن سيق) عداد مع ثلاثة أزرار ضاغطة up, down , reset

تمرين: صمم ساعة رقمية لتعد الثواني و الدقائق

استخدام اللوحة المتكاملة max7219

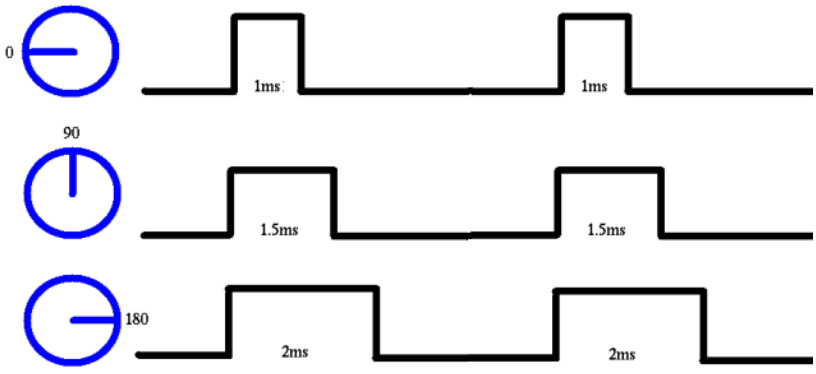
اذا كنت تحتاج 8 وحدات من السيفين سيقمينت أو 16 أو 24 و باستخدام 3 منافذ من الأردوينو فقط (+طرفي التغذية) فالأسهل و الأرخص استخدام هذه اللوحة المتكاملة . لن يتسع المجال في الكتاب لشرحها يمكنك مشاهدة الرابط هنا : https://www.youtube.com/watch?v=VgNIkGzxGAU

التحكم بمحرك سيرفو servo motors



توجد عدة أنواع من المحركات _أشهرها
1-محرك دي سي DC-motor
2-محرك خطوة stepper motor
3- محرك سيرفو servo motor
لاحظ أيضا أن محركات السيرفو تنقسم لقسمين

سيرفو الزاوية	hobby (angle) servo	حسب الدخل(عادة عرض نبضات) تتحرك الذراع إلى الزاوية المطلوبة و تقف عندها. هذا النوع أكثر شيوعا.
سيرفو الدورات الكاملة	continuous rotation servo	يدور مثل محرك دي سي. ويمكنك التحكم بسرعته واتجاهه يحتوي داخليا على H-bridge للتكبير والتحكم. (هذا النوع أصبح نادر الوجود)



انظر إلى رسم إشارة التحكم (بين كل بدايتي نبضة 20ms)هل بإمكانك توليد إشارة كهذه بدون استخدام مكتبة؟

لاحظ أن المحركات الرخيصة غير موثوقة كثيراً وبعضها تكون تالفة قبل استخدامها.
لاحظ أن المحركات عادة تأتي بنطاق تشغيل من الزوايا (مثلاً 0 إلى 180 درجة -نصف دورة فقط) لكن في الواقع هي لا تتحمل هذه الزوايا . مع التجربة يمكنك ملاحظة النطاق الآمن و الذي لا يسبب اهتزاز المحرك (تقريبا 15-170 درجة)
لمشاهدة أمثلة للتحكم بمحرك سيرفو يمكنك مشاهدة الأمثلة المرفقة مع برنامج الأردوينو

File >> Examples >> Servo >> Sweep

سنستخدم مكتبة Servo.h للتحكم بمحرك سيرفو بسيط (زاوية وليس دوراني).

```
#include  <Servo.h>
Servo  moto;  //any name should do
void setup(){
  moto.attach(7);
  moto.write(170);}   //لتحديد الزاوية
void loop(){}
```

شاهد كود قريب من السابق اضغط هنا

في حالة أن المحرك سيرفو دوراني (يدور دورة كاملة وليس زوايا فقط) full rotation servos تختلف طريقة التحكم (يمكنك التحكم بسرعة الدوران و الإتجاه بشكل أسهل من محرك الدي سي).
بعد تضمين المكتبة وتعريف الكائن (المحرك) بإمكاننا استخدام الأوامر التالية للتحكم بدوران المحرك:

تحديد المنفذ المتصل للتحكم بالمحرك	`moto.attach(10);`
الدوران بأقصى سرعة مع عقارب الساعة	`moto.write(170);`
الدوران بأقصى سرعة عكس عقارب الساعة	`moto.Write(15);`
أمر آخر للتحكم بدقة في حركة المحرك	`moto.writeMicroseconds(1300);`
عادة القيم (1000،1500،2000) تستخدم للحركة بأي اتجاه أو إيقاف المحرك ، لكن يجب عليك تجربة محركك	
إيقاف المحرك	`moto.detach();`

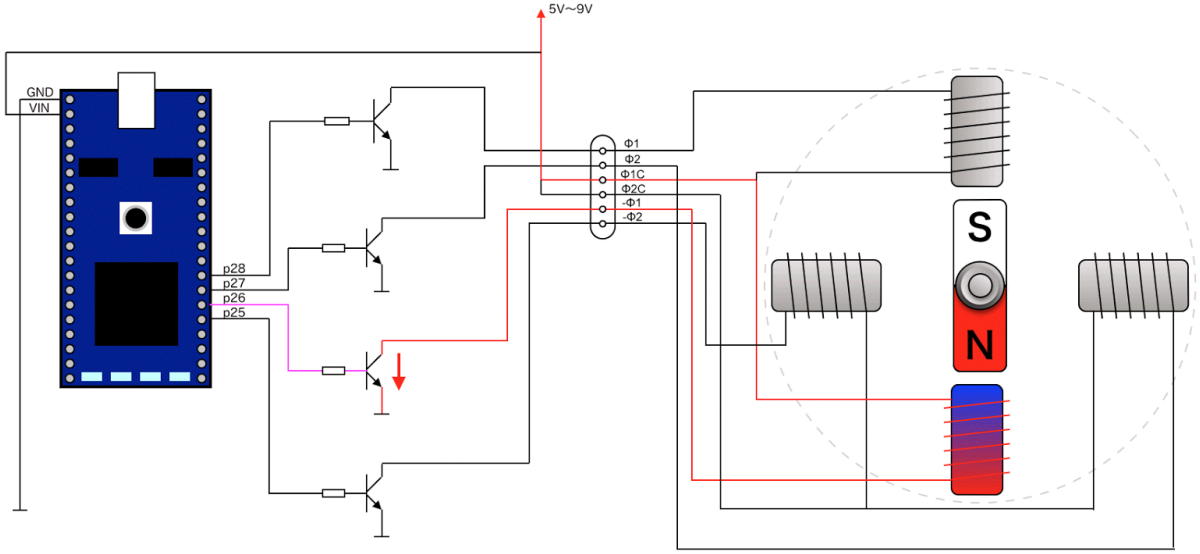
اقرأ المزيد عن مكتبة محرك السيرفو

مثال مناسب لمحرك سيرفو الدورة الكاملة full rotation:-

```
#include <Servo.h>
Servo moto;
moto.attach(2);  //pin connected to motor
moto.writeMicroseconds(1300); //1300=clockwise
delay(2000);
moto.writeMicroseconds(1700);//1700=counterClockWise
delay(2000);
moto.detach();  //stops the motor
```

محرك الخطوة stepper motor

محرك الخطوة هو أحد أنواع المحركات الكهربائية الميكانيكية . ولعله الأدق من ناحية التحكم بالسرعة و زاوية الوقوف.
محركات الخطوة لها تركيب و نظرية عمل تختلف عن المحركات السابقة و يكون معها عادة دائرة تحكم لتكبير التيار اللازم للتشغيل.
الصورة التالية توضح فكرة عمل Bipolar وأعتقد من الواضح أنه أعقد كثيرا من محرك دي سي.



يوجد ثلاث طرق للتحكم بخطوات المحرك (خطوة بقطب ، خطوة بقطبين ، نصف خطوة) لا تحتاج للتعمق في كل هذه الطرق شاهد الرابط التالي إذا أردت فهم الطرق المختلفة (هـــنا)
أيضاً يمكنك التحكم بمحرك الخطوة بسلكين أو بأربعة (سنركز على الأسهل هنا وهو 4 أسلاك)
يستهلك السلك الواحد لمحرك الخطوة حوالي 150mA ومنافذ الأردوينو لا تتمكن من إخراج هذا التيار.

يمكن تقسيم محركات الخطوة إلى قسمين بشكل عام :

Bipolar	وجود طرفين لكل ملف يمكن التحكم بهما وعكس القطبية	يحتاج لـ H-bridge مثل L2093D و يعطيك عزم أقوى (torque) المزيد
Unipolar	وجود طرف واحد للتحكم بكل ملف.	يحتاج دائرة تكبير بسيطة مثل ULN2003A المزيد

صندوق التروس Gear box وعدد الخطوات steps/rev:

كما رأينا في الرسم السابق فإن الدورة الواحدة تتكون من 4 أقطاب وتحتاج 4 خطوات. ولكن في معظم المحركات نحن نستخدم صندوق التروس gear box لزيادة القوة مع إنقاص السرعة. هذا يجعل الدورة الواحدة التي تراها تحتاج إلى 100 أو 200 خطوة ! ستجد عدد الخطوات اللازمة لإكمال دورة في مواصفات كل محرك. واذا لم تعرف خطوات المحرك الذي معك ولكن وجدت زاوية الخطوة (مثلا:7.2) فاقسم 360 على الزاوية لتعرف عدد الخطوات.

المحرك الذي سنستخدمه في التجربة (28BYJ-48) من نوع Unipolar وتنقسم الدورة الواحدة فيه إلى 64 خطوة وسوف نشغله بسرعة 60 خطوة في الثانية وبطريقة (4) أسلاك تحكم.

```
#include <Stepper.h>
int spr = 64;
Stepper moto(spr, 8, 10, 9, 11);
void setup() {
  moto.setSpeed(60);}
```

شرح هذا الجزء: 1-تضمين المكتبة

2- تحديد عدد الخطوات للدورة الواحدة في المحرك steps per revolution

3- إنشاء كائن object -instance مع تعريف الخصائص (الخطوات ، و المنافذ المستخدمة)

لاحظ: ترتيب الأرقام للمنافذ مهم _ إذا لم يدر المحرك بالشكل المطلوب غير ترتيب المنافذ.

4- داخل setup نحدد سرعة الدوران (كم دورة في الدقيقة rpm) لكل محرك حد أقصى للسرعة

ملاحظة: إذا أحببت متابعة كل طرف و التأكد من خطوات التشغيل ؛ أنقص السرعة إلى (1)

```
void loop() {
  moto.step(3*spr);
  delay(500);
  moto.step(-500);
  delay(500); }
```

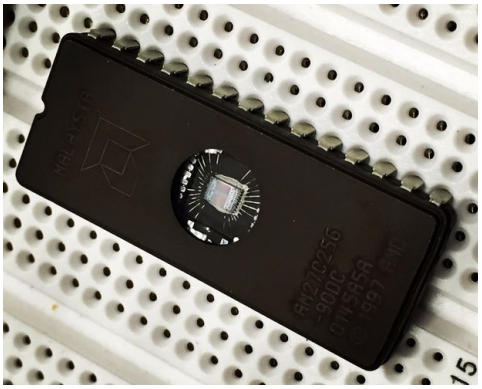
الأمر `object.step` تحدد له عدد الخطوات ، و في المثال سيدور المحرك ثلاث دورات .

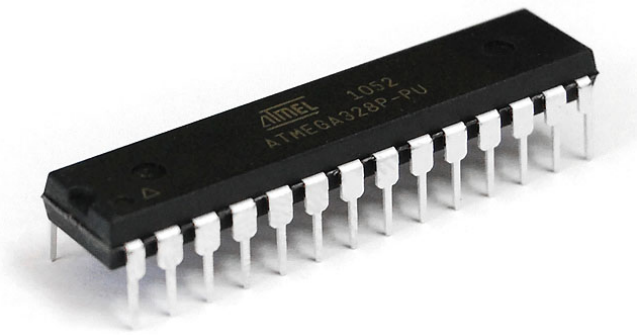
ثم يتوقف لنصف ثانية ، ثم يدور في الاتجاه المعاكس لـ 500 خطوة.

لقراءة صفحة محرك الخطوة اضغط هنا

تمرين : صمم كود للتحكم بمحرك خطوة بوجود 4 أزرار للتحكم
يمين سريع ، يمين بطيء ، يسار سريع ، يسار بطيء

الذاكرة الدائمة EEPROM





الذاكرات ... دعنا نتحدث قليلا عن الذاكرات قبل أن ندخل في الكود 😀

الذاكرات ضرورية لعمل أي نظام كمبيوتري . لا بد من تخزين البرنامج (الكود، الأوامر)، و لا بد من متابعة قيم المتغيرات أثناء عمل البرنامج (ذاكرة RAM) . وفي كثير من الأحيان تحتاج لتخزين القيم في ذاكرة دائمة لتعود و تفتحها لاحقاً (مثل الهارديسك في الكمبيوتر)

كما قلنا سابقاً يعتمد الأردوينو على شريحة واحدة (مايكروكنترولر) هي Atmega328p

هذه الشريحة تحتوي مكونات عديدة ومنها 3 أنواع من الذاكرة

ذاكرة الفلاش Flash memory	و تستخدم لتخزين الكود (sketch)	**32KB**
ذاكرة الـ رام RAM	و تنشط أثناء تنفيذ البرنامج مثل قيم المتغيرات	**2KB**
ذاكرة الايبروم EEPROM	وهي ذاكرة سنركز عليها في الصفحات التالية	**1KB**

الحاجة لذاكرة الـ ايبروم : تخيل أنك صممت جهاز ليقرأ درجة الحرارة كل 10 دقائق ويضع هذه القيم (درجات الحرارة) في متغيرات : `int x , int y , int z` مثلاً.

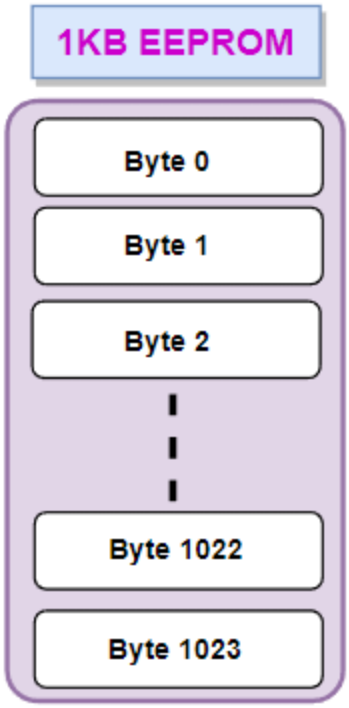
قمت بكتابة الكود ورفعته إلى الأردوينو ، فتم تخزينها في ذاكرة Flash memory

ثم وضعته في مكان التشغيل (مثلا مصنع) و شغلته باستخدام بطارية. ثم إنك أحببت أخذ هذه القيم إلى الكمبيوتر.

المشكلة أنك بمجرد فصل الطاقة عن الأردوينو فإن جميع هذه القيم ستختفي. لإنها فعلياً كانت مخزنة في الـ رام فقط. وذاكرة الـ RAM تفقد كل محتوياتها بمجرد فصل الطاقة عنها.

ذاكرة الإيبروم EEPROM مخصصة للاحتفاظ بالبيانات حتى بعد فصل الطاقة عنها. وهي مفيدة كثيرا في كثير من المشاريع المتقدمة.

الـ EEPROM هي ذاكرة مكونة من بايتات Bytes منفصلة. لكل بايت عنوان ويمكنه تخزين قيمة (0-255) فقط وتعتبر قيمة صغيرة.
في الأردوينو أونو (شريحة Atmega328p) تكون سعة الايبروم = 1KB و تكون البايتات مرقمة من 0 إلى 1023



هناك مكتبة رسمية من الأردوينو للتحكم بالـ EEPROM
لذا يجب إضافة السطر التالي لتضمين المكتبة أعلى الكود :

```
#include <EEPROM.h>
```

قبل أن نبدأ بكتابة الكود نود التنبيه على عدة أمور:

-توجد مكتبات عديدة غير رسمية للتحكم بالـ EEPROM ولديك الحرية في استخدامها وفي بعض الأحيان تكون أسهل من المكتبة الرسمية لكننا لن نشرحها في هذا الكتاب.

-تعمل ذاكرة الـ EEPROM بشكل أبطأ من الـ RAM (حوالي 3ms) لتخزين البايت الواحد.

-الإيبروم لها عمر افتراضي من مرات الكتابة أو القراءة (حوالي 100 ألف مرة) الرقم كبير لاشك. لكن في بعض التطبيقات ينبغي الانتباه لهذا الأمر حتى لا تتلف الـ ايبروم

الأوامر الأساسية:-

```
EEPROM.write(0,235);   //(address:0-1023 , value 0-255)
Byte x=EEPROM.read(10);   //(address)
```

الأمرين السابقين سهلين و واضحين سوى أن باستخدامهما مشكلة و هي أنهما يعملان على تخزين أو قراءة بايت واحد فقط . بينما معظم المتغيرات (int مثلا) تكون أكثر من بايت.

نوع المتغير	int	float	Byte	bool	long	char	String
عدد البايتات	2	4	1	1	4	1	any

شاهد المثال التالي لتخزين رقم int في ذاكرة الايبروم (x) ثم إعادة قراءته و وضع القيمة في (y)

```
int x=31287;
int y;
EEPROM.put(0, x); //(address,data)
EEPROM.get(0, y); //(address,destination variable)
```

يجب أن يكون المتغيرين من نفس النوع وفي حال اختلاف النوع قد تظهر قراءات خاطئة.

نفس هذه الطريقة يمكن استخدامها مع متغيرات من أنواع مختلفة مثل float , char
ولكن انتبه لعدد البايت المستخدمة لكل نوع . مثلا إذا خزنت (`char x [10]`) مكون من 10 حروف . لا تستخدم أول 10 بايت . فهي مستخدمة بالفعل.
ملاحظة: لا يمكن تخزين عبارة بصيغة String وبدلاً عن ذلك استخدم مصفوفة حروف محددة الحجم
مثال::

```
char x[12]="sami grami..";
EEPROM.put(500,x);  //Writing on location 500
char y[12];
EEPROM.get(500,y);  //Reading from location 500
```

لمعرفة عدد البايتات المستخدمة في أي متغير أو مصفوفة ؛ استخدم الأمر sizeof مثل :

```
float x=1.2;
String y="hello world";
Serial.println(sizeof(x));
Serial.println(sizeof(y));
```

ويفترض أن يظهر على الشاشة الرقمين : 4 و 11

مثال: خزن العبارة: sami grami في ذاكرة الـ EEPROM ثم استخرجها و اعرضها على الشاشة.

```
#include <EEPROM.h>
char x[10] ="sami grami";
char y[10];
void setup(){
  EEPROM.put(100,x); //used address : 100
  EEPROM.get(100,y);
  Serial.begin(9600);
  Serial.println(y);}
void loop(){}
```

كود الكتابة كل 3 ثواني من قراءة مقاومة متغيرة :

```
#include  <EEPROM.h>
int x=0;
int add=0;
void setup(){
  delay(10000); //so the Arduino doesn't write right away
```

```
  Serial.begin(9600);}
void loop(){
  x=analogRead(A0);
  Serial.println(x);
  if(add<1022){ EEPROM.put(add,x); add=add+2;}
  delay(3000);}
```

كود قراءة 40 رقم من ذاكرة ايبروم و عرضها على الشاشة:

```
#include  <EEPROM.h>
int x=0;
void setup(){
  Serial.begin(9600);
  for(int i=0 ; i<80 ; i=i+2){
    EEPROM.get(i,x);
    Serial.println(x);}}
void loop(){}
```

تمرين:-

أ- صمم كود يقيس شدة الاستضاءة في غرفة ويخزن القراءة في الايبروم ويكرر العملية كل 10 ثواني . وفي حال امتلاء الذاكرة الايبروم. يتوقف الكود عن التخزين.
أطفئ الأردوينو . الأفضل جعل العداد address مخزن في الايبروم بحيث في حال انقطاع الكهرباء فإن التخزين يكمل من مكان التوقف و لا يعيد من جديد.

ب- صمم كود آخر لقراءة محتويات الذاكرة EEPROM و عرضها على شاشة السيريال

LCD Display شاشة الكريستال

الشاشة من أهم المكونات التي تساعدك لعمل مشاريع رائعة فبدل أن تربط الأردوينو بالكمبيوتر لعرض القيم و العبارات . يمكنك أن تعرض العبارات و المعلومات على شاشة بسيطة في أي مكان و بتكلفة منخفضة.

السعر على أمازون حوالي ($8) = 30 ريال تقريباً

أشهر شاشات الكريستال تتمكن من عرض صفين في كل صف 16 حرف لذا تسمى LCD 2*16

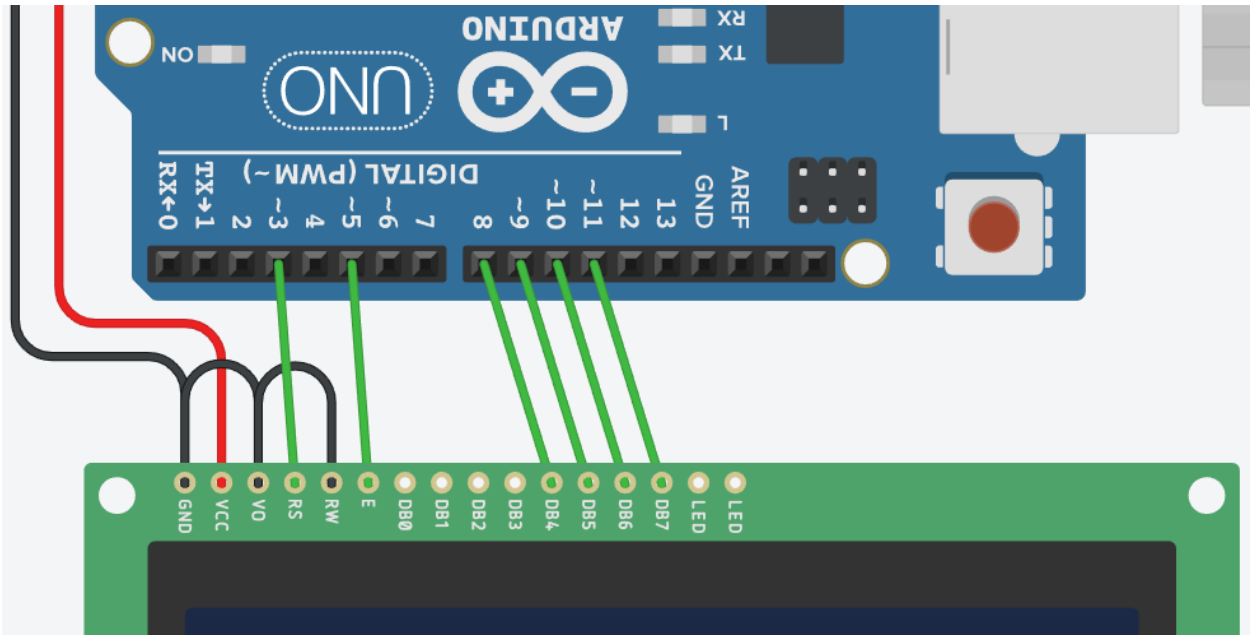
بينما توجد شاشات بمقاسات مختلفة مثل : 4*20

توجد شاشات أدق و لكنها أصعب تسمى OLED لن نشرحها في هذا الكتاب

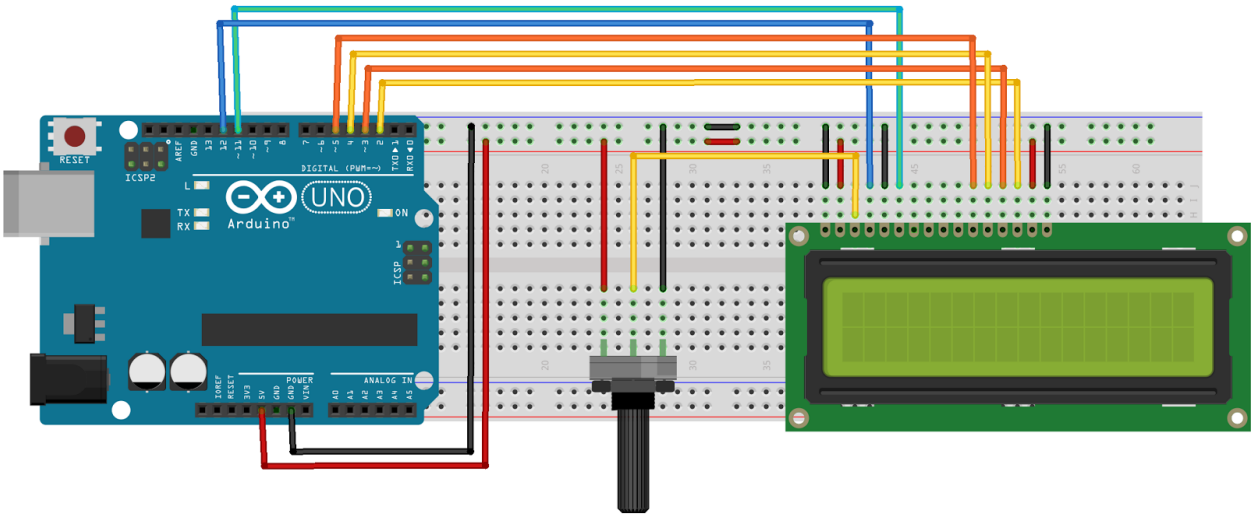
في البداية يجب عليك معرفة الأطراف و كيفية توصيلها . و بعض الشاشات تكتب لك أسماء الأطراف بوضوح ، بينما للبعض الآخر يجب عليك البحث في ورقة مواصفات الشاشة

(قد تكون مرفقة ، و قد يتوجب عليك ايجادها في الانترنت)

طريقة التوصيل لأحد الأنواع الشائعة



قمنا بتوصيل التغذية +5 و GND باللونين الأحمر و الأسود وصلنا الـ RW إلى GND الطرف VO (يسمى VEE أحياناً) يجب توصيله إلى الـ GND في بعض الشاشات. ويجب توصيله إلى مقاومة متغيرة في شاشات أخرى لضبط إضاءة الشاشة (التباين)



الجميل أن باقي الأطراف يمكنك توصيلها لأي منفذ رقمي بحيث تقوم بتعريفها في بداية الكود كما يظهر:

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(3,5,8,9,10,11); // (RS,E,D4,D5,D6,D7)
```

ترتيب الارقام هنا هام _ ويجب أن يتوافق حسب توصيلك (لاحظ ترتيب الأرقام في الكود مع التوصيل)

لقراءة المزيد عن خيارات الأمر السابق (constructor) اضغط هنا

بعد تعريف الشاشة ستحتاج لبعض الأوامر البسيطة للكتابة على الشاشة

```
int x=100;
void setup() {
  lcd.begin(16,2);  // (columns , rows)
  lcd.print("hello, world ! ");  // لإظهار عبارة مكتوبة
  lcd.setCursor(0,1);  //يذهب للسطر الثاني
  delay(3000);
  lcd.clear();  //لمسح كل ما يظهر على الشاشة
```

في حالة رغبتك في عرض عبارة طويلة (أكثر من 16 حرف للسطر) يمكنك الاستفادة من الأمر:

scrollDisplayLeft( )

```
  lcd.print("hello, world!  my name is Sami Grami");
  delay(500);
  for(int i=0;i<20;i++){  //20 عدد خطوات الازاحة هنا
    lcd.scrollDisplayLeft();
```

```
    delay(300);}
```

#include <LiquidCrystal.h>	تضمين المكتبة اللازمة
LiquidCrystal lcd (8,9,4,5,6,7);	تكتب قبل void setup و فيها تتغير الأرقام حسب الحاجة (RS,E,D4,D5,D6,D7)
lcd.begin(2,16);	2 هو عدد الأصفف و 16 عدد الأعمدة
lcd.clear();	مسح جميع محتويات الشاشة.
lcd.setCursor(0,0);	وضع المؤشر في أي مكان في الشاشة (c,r)
lcd.print("hello world");	يكتب عبارة على سطر في الشاشة أو قيمة متغير
lcd.write(x) ;	يستخدم لإظهار الحروف بصيغة char

lcd.home();
lcd.cursor();
lcd.noCursor();
lcd.blink();
lcd.noBlink();
lcd.noDisplay();
lcd.display();
lcd.scrollDisplayLeft();
lcd.scrollDisplayRight();
lcd.autoScroll();
lcd.noAutoScroll();
	أوامر إضافية (نادرة الاستخدام)

للمزيد زر صفحة مكتبة الـ LCD من:- هـــنـــا

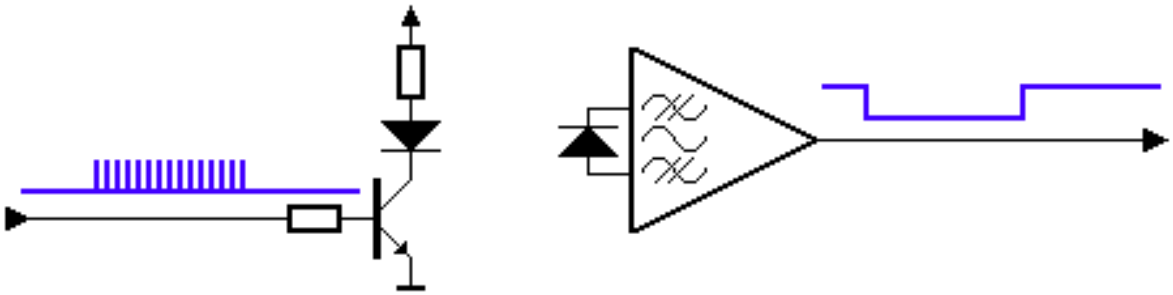
تمرين : صمم ساعة رقمية تعرض الثواني و الدقائق و الساعات على شاشة LCD

استخدام الريموت كنترول (IR) للتحكم بتشغيل الأردوينو

واو ...  **واااااااااو ... من أروع مميزات الأردوينو (في رأيي) أنه يتيح لك التحكم اللاسلكي باستخدام ريموت كنترول بسيط. يمكنك الآن بسهولة التحكم بتشغيل الأجهزة الكهربائية بريموت التلفزيون أو الرسيفر (ياناس أحب الأردوينو ذا)** 

سنشرح قليلاً عمل الريموت قبل أن نبدأ بالبرمجة و الاستخدام:

يعمل الريموت (المرسل) على إرسال إشارة (ذبذبات) بضوء غير مرئي (تحت الحمراء) وحسب كل زر تضغطه فإنه يرسل الذبذبات بطريقة معينة. هذه الذبذبات يستحسن أن تترجم من شكل الإشارة إلى رقم (بالصيغة السداسية عشر)



تختلف الريموتات حسب طريقة التشفير .

مثلاً في ريموت تلفزيون سوني عندما تضغط الزر (1) فإن الكود المرسل يكون : **0xFF12C409**

بينما في ريموت سامسونق عندما تضغط على الزر (1) فإن الكود المرسل يكون : **0x7070807F**

الخبر الجيد أنه أن الأردوينو يمكنه التعرف على أي ريموت . فقط اتبع الخطوات التالية :-

المرحلة الأولى: معرفة الكود الذي يتم ارساله عند الضغط على كل مفتاح
(مثلاً 0xFA58H3 عند الضغط على السهم الأيمن)

المرحلة الثانية: تغيير الكود بحيث نكتشف الزر الذي تم الضغط عليه (بقراءة الكود)
ثم تنفيذ العمل المطلوب (مثلاً تشغيل أو إطفاء لمبة)

الأوامر الرئيسية:-

`IRrecv rec(10);`	تعريف كائن مستقبل اسمه rec ومتصل بالمنفذ 10
`decode_results res;`	كائن اسمه res وهو مساحة في الذاكرة لتخزين الشفرة
`rec.enableIRIn();`	تشغيل المستقبل للـ IR وانتظار إشارة من الريموت
`while(rec.decode(&res)==0){}`	ايقاف البرنامج بانتظارإشارة من الريموت
`if(rec.decode(&res)){`	طريقة أخرى لكشف إشارة الريموت بدون ايقاف
`long x= res.value;`	بهذه الطريقة نحصل على القيمة من الكائن res
`rec.resume(); }`	انتظار إشارة جديدة من الريموت

شرح المرحلة الأولى :- قراءة الكود القادم من الريموت.

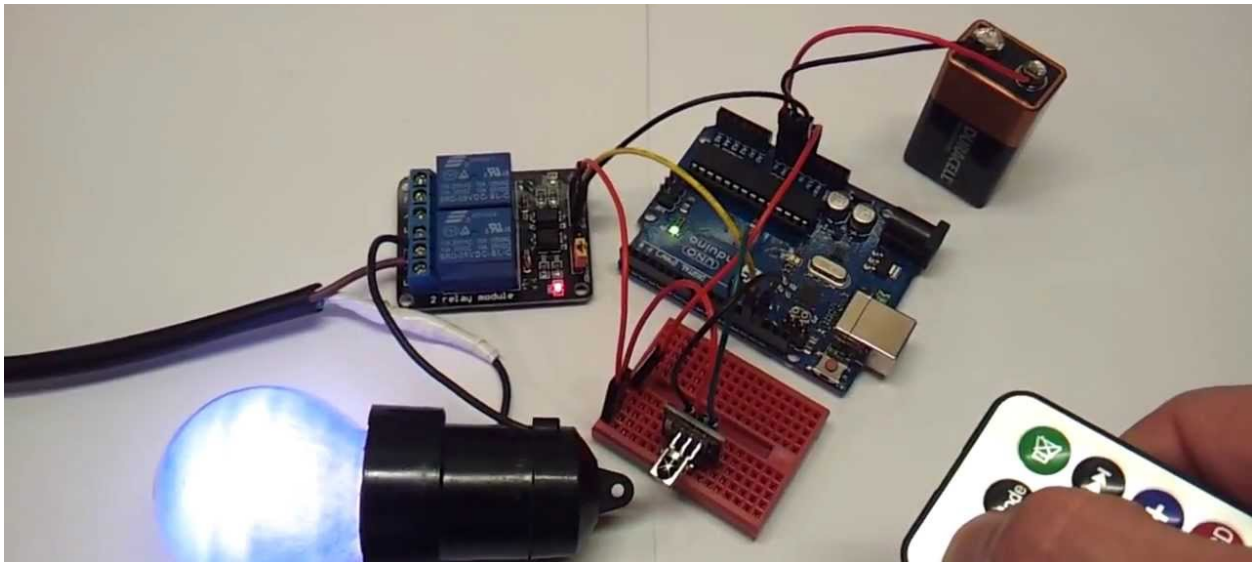
```
#include <IRremote.h>
IRrecv recv(3);  //هنا ضع رقم المنفذ المستخدم
decode_results res;
void setup() {
  Serial.begin(9600);
  recv.enableIRIn();}
void loop() {
  if (recv.decode(&res)) {
    Serial.println(res.value, HEX);
    recv.resume();}
  delay(100); }
```

افتح صفحة الـ سيريال و اضغط على أزرار الريموت (مع توجيه الريموت للحساس)
ستظهر الشفرات الخاصة بكل زر تضغطه.
في ريموتي أحتاج استخدام زرين فقط هما:- **707005FA** و **7070C53A**

سنقوم الآن بكتابة كود بسيط لتشغيل LED و إطفائه باستخدام نفس الزرين في الريموت.

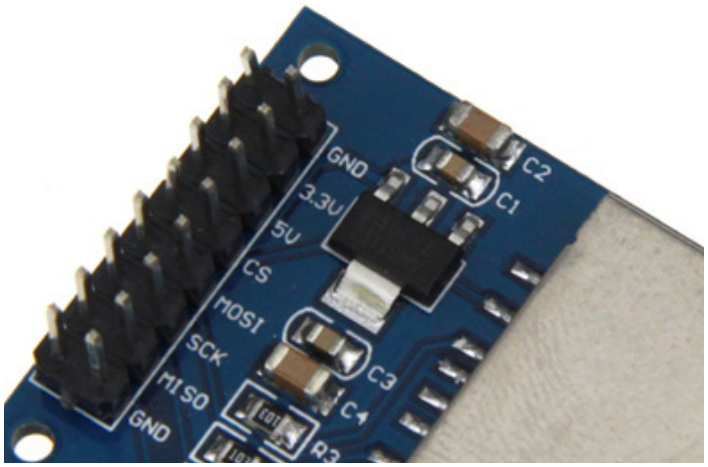
```
#include <IRremote.h>
IRrecv rec(7);  //here put receiver pin
decode_results res;
void setup() {
  rec.enableIRIn();
  pinMode(13,OUTPUT);
  digitalWrite(13,LOW);}
void loop() {
  if (rec.decode(&res)){
    if(res.value==0x707005FA){
    digitalWrite(13,HIGH);}
    else if(res.value==0x7070857A){
    digitalWrite(13,LOW); }
    rec.resume(); }  }
```

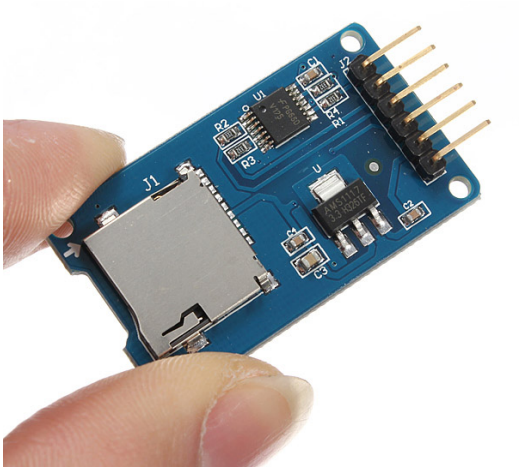
حمل المكتبة _ وشاهد بعض الأمثلة من ::- هـــنـــا



تمرين : صمم دائرة باستخدام الأردوينو بحيث تربط ريموت كنترول بالأردوينو . و تستقبل جميع الأرقام 0-9 و يكون الخرج هو سماعة بسيطة. و العمل أن تصدر نغمة مختلفة عند الضغط على كل رقم.

التخزين على ذاكرة SD





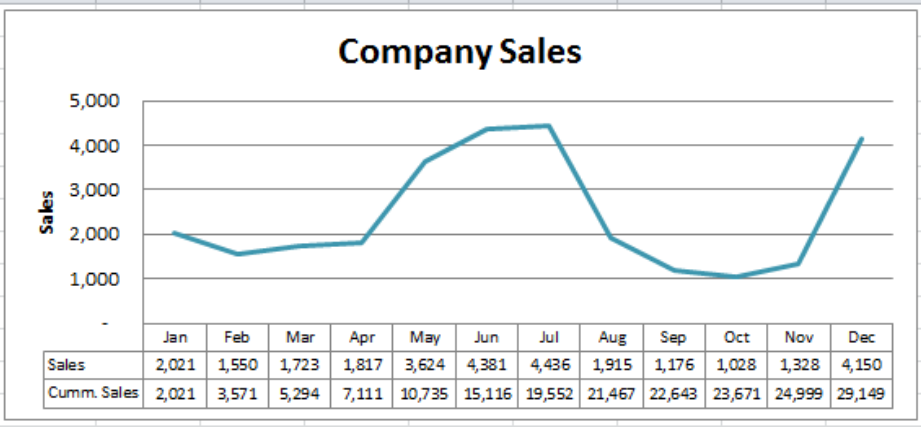
في بعض المشاريع تحتاج لتسجيل مقدار كبير من المعلومات (القراءات) بدون أن تبقى باتصال مع الكمبيوتر. مثال: قياس درجة الحرارة طوال السنة. هذا العمل يسمى logging .

بإمكانك استخدام الذاكرة الدائمة داخل شريحة الأردوينو EEPROM لكن حجمها صغير (1KB) كما أن استخراج المعلومات منها يعتبر نسبيا صعب لغير المتخصص.

الخيار الأمثل في هذه الحالة هو استخدام ذاكرات الـ SD فهي رخيصة واستخدامها سهل.

الرائع في الأمر أنه بإمكانك أخذ مئات أو آلاف القراءات بسهولة من الأردوينو إلى الذاكرة SD إلى الكمبيوتر إلى برنامج أكسيل ؛ ثم رسم شكل بياني يبين لك خلاصة كل تلك القراءات.

	A	B	C
1		Sales	Cumm. Sales
2	Jan	2,021	2,021
3	Feb	1,550	3,571
4	Mar	1,723	5,294
5	Apr	1,817	7,111
6	May	3,624	10,735
7	Jun	4,381	15,116
8	Jul	4,436	19,552
9	Aug	1,915	21,467
10	Sep	1,176	22,643
11	Oct	1,028	23,671
12	Nov	1,328	24,999
13	Dec	4,150	29,149



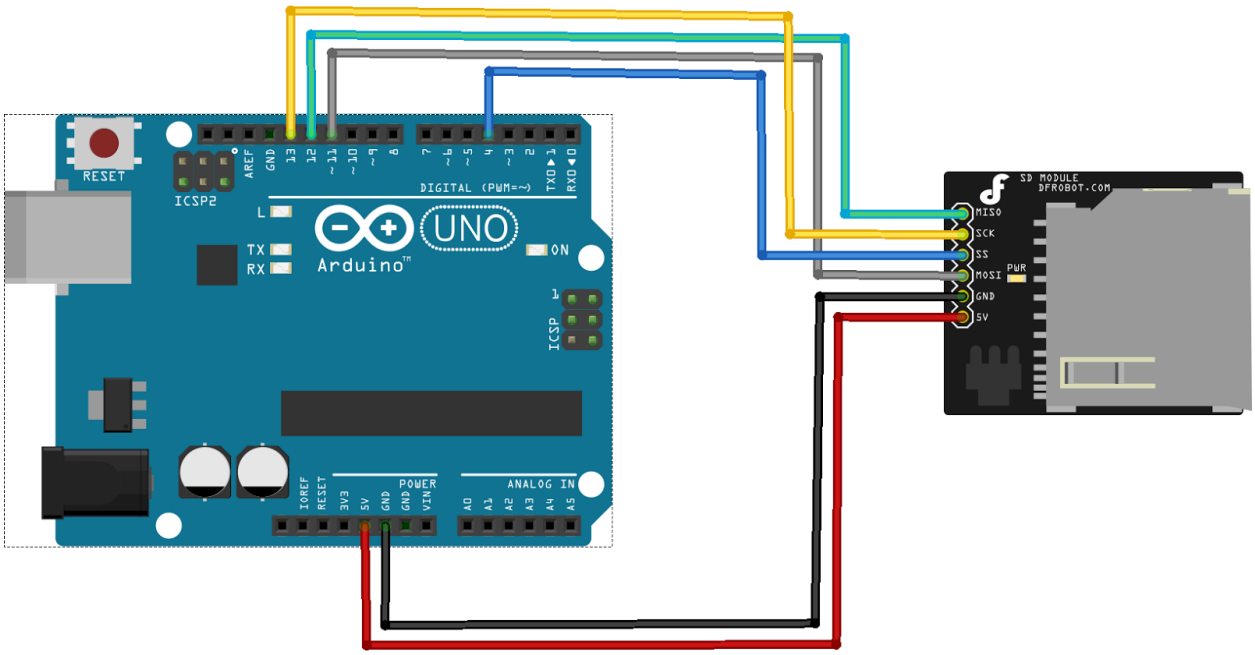
الأردوينو أونو لا يأتي مع منفذ للذاكرة لذا يجب شراء الدائرة الملحقة التي تربط الأردوينو بذاكرة الـ SD توجد أنواع عديدة من هذه الدوائر ، لذا تأكد من قراءة التعليمات قبل استخدامها وتوصيلها.

بروتوكول التواصل serial peripheral interface SPI

بروتوكولات التواصل موضوع كبير ولن يتسع الكتاب لشرحها بشكل متوسع سوى أننا سنشرح الفكرة باختصار. يوجد بروتوكولات عديدة لربط الأجهزة بالملحقات مثل : UART , I2C , SPI
لعل من أسهلها SPI ويستخدم بكثرة لربط مكونات الأنظمة الإلكترونية (أردوينو، حساسات، ذاكرات ، شرائح إلكترونية متنوعة)
يستخدم بروتوكول الـ SPI أربعة أسلاك

MOSI	master out/ Slave in	pin11	نقل البيانات تسلسليا من الأردوينو إلى الذاكرة
MISO	master in / Slave out	pin12	نقل البيانات من الذاكرة إلى الأردوينو
CLK	Clock	pin13	نبضات التزامن
CS	Chip select	any*	اختيار الملحق الفعال (في حال الربط بأكثر من ملحق)

توصيل مدخل الذاكرة بالأردوينو :



عند توصيل الدائرة الإضافية (مدخل الذاكرة SD) وصل جميع الأسلاك التي تحدثنا عنها في أماكنها المذكورة في الجدول: MOSI , MISO , CLK
CS في أي منفذ آخر
5v و GND بشكل طبيعي . ولا توصل 3.3v

الطريقة التي ينصح بها للكتابة على ذاكرة SD card تتكون من عدة خطوات : شاهدها في المثال.

```
#include <SPI.h>
#include <SD.h>

File sami;   //إنشاء كائن بأي إسم
int CS=10;   //تحديد المنفذ المستخدم لـ سي إس
int x=20178;

void setup() {
  if (!SD.begin(CS)) {return;} // التأكد من تشغيل الذاكرة بشكل سليم
  sami = SD.open("semsem.txt", FILE_WRITE); //فتح الملف
  if (sami) {   //لا تكتب إلا إذا فتح الملف
    sami.println("I will upload once only!");
    sami.println(x);//اكتب عبارة أو قيمة متغير
    sami.close(); }}//يجب إغلاق الملف بعد كل كتابة
void loop() {}
```

لاحظ أن **sami** هو اسم الكائن (في الكود) بينما اسم الملف في الذاكرة سيكون **semsem.txt**

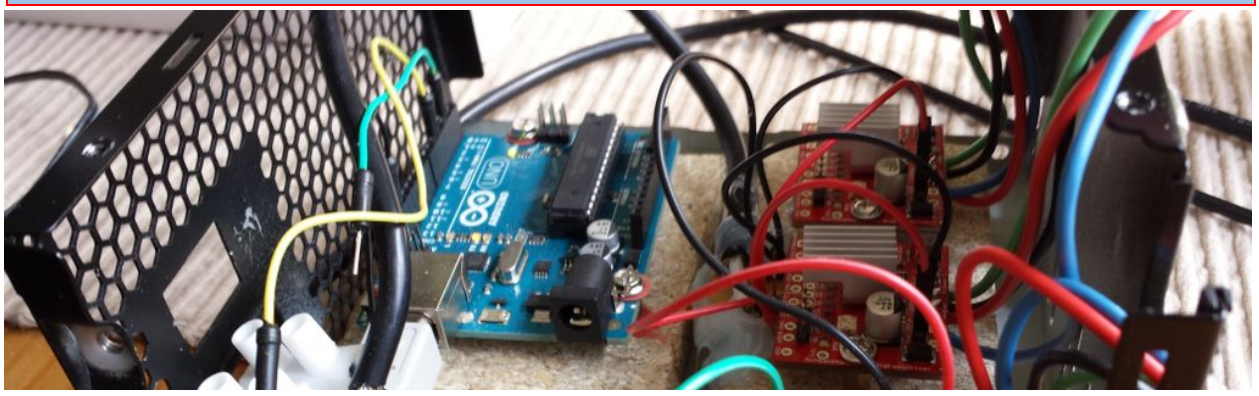
للاستزادة شاهد: درس يوتيوب أو اقرأ صفحة القراءة و الكتابة من الموقع الرسمي: هـنا
لمعرفة كيفية تحويل القراءات إلى رسم شاهد الفيديو (اضغط هنا)

تمرين: اربط مقاومة متغيرة بالأردوينو ، اقرأ قيمة الجهد التماثلية و اكتبها على ذاكرة SD
و اكتب معه قيمة الزمن بأمر millis كرر العملية كل 100 ميللي ثانية
ثم افتح الملف في الكمبيوتر .

الحل:- وصل موديول القراءة و الكتابة على الذاكرة SD إلى المنفذ الرقمي 4
وصل المقاومة المتغيرة على الطرف A5

```
#include <SPI.h>
#include <SD.h>
File sami;
int CS=4; //تحديد المنفذ المتصل بالذاكرة
int x;   //هذا المتغير سيحمل قيمة القراءة في كل مرة
void setup() {
   if (!SD.begin(CS)) {return;} }
void loop() {
  x=analogRead(A5);
  sami = SD.open("log3.txt", FILE_WRITE);
  if (sami) {
    sami.print(x);
    sami.print(",");
    sami.println(millis());
    sami.close();
    delay(100);}
 }
```

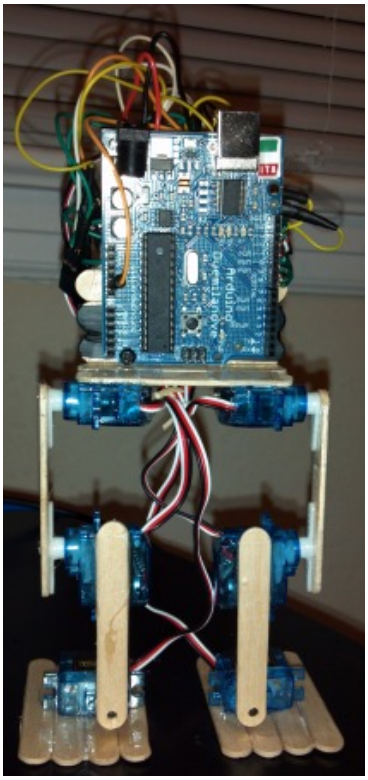
الباب الخامس: المشاريع الإلكترونية Electronic projects



هذا هو الباب الأخير في هذا الكتاب . وهو لن يكون عن الأوامر البرمجية والمكتبات . بل لن يكون درسا! سيكون أقرب لحوار خفيف على كوب شاي... أتمنى ذلك.

المشاريع الإلكترونية هي أجهزة إلكترونية يصنعها الهواة و الطلاب و المهندسين لأهداف معينة. أحيانا لتنفيذ نموذج مبدئي لجهاز يحل مشكلة معينة . و أحياناً لحل مشكلة يواجهها الشخص في البيت أو في الحياة عموماً. بعض المشاريع الهدف منها التعلم و زيادة الخبرة. وربما قد يعتبر مشروعك هو سيرتك الذاتية التي سوف تحصل بها على وظيفة أحلامك.

كثير من الأعمال التي نفذناها بالأردوينو سابقاً هي لا تحتاج في الحقيقة للأردوينو لتنفيذها !! بل يمكن تنفيذها بدوائر أبسط وأرخص !! مثل الوميض أو إصدار الصوت أو تشغيل محرك دي سي أو بعض الأوامر البسيطة الأخرى . الأردوينو مفيد جداً لا شك . سوى أن الفائدة الحقيقية في النهاية هي ماذا إستطعت أن تفعل بالأردوينو ؟



بالنسبة للمستخدم العادي: هل مشروعك يحقق له فائدة يحتاجها ؟ أم لا؟

بالنسبة للشركات تكون المعادلة: هل هذا المشروع قابل للتسويق و تحقيق أرباح؟ أم لا ؟

بالنسبة لي : المشاريع تنقسم لقسمين عموماً

1- مشاريع هدفها حل مشكلة أو تطوير جهاز موجود.

2- مشاريع هدفها شد الانتباه (في المعارض) بتطبيقات غريبة وجذابة للزوار.

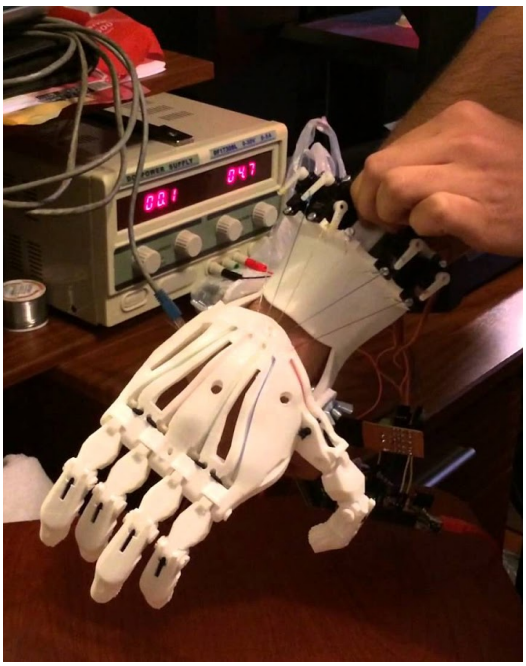
شاهد مثلاً كيف يستفيد هذا الفنان من الأردوينو لإنتاج مشاهد الدمى المتحركة (اضغط هنا)

في الحقيقة هناك مجالات عديدة عامة قد تندرج تحتها عشرات و مئات الأفكار و المشاريع مثل:

الأتمتة automation (الكلمة صعبة ، أعلم): جعل الكهرباء و الإلكترونيات تقوم بعمل بسيط اعتاد الإنسان القيام به مثل ري الأشجار ، أو تشغيل الإضاءة . أو إطعام الحيوانات. أو تنظيف البيت . الغسالة الفل اوتوماتيك مثال أيضا.

توفير الطاقة أو الماء : مجال كبير و هام فاستهلاك الطاقة الكهربائية كبير جداً ، و كثير من هذا الاستهلاك يضيع دون استفادة حقيقية . يمكن ببعض الذكاء الإلكتروني التقليل من هذا الفقد.

السلامة Safety : عادة لن نفكر بهذه المشكلة إلا بعد أن نرى الإصابات و الخسائر. توجد مخاطر عديدة ، في المصنع في المنزل في الحديقة في المسبح. يمكن تصميم أنظمة آلية لحماية الأشخاص و الممتلكات من هذه المخاطر.



خدمة أصحاب الاحتياجات الخاصة : يوجد أشخاص لديهم إعاقات سمعية أو بصرية أو حركية. و الآلة قد تجعل حياتهم أسهل بكثير. هذا مجال جميل و رائع.

تسهيل الوصول (مثل ريموت التحكم عن بعد) : أن تفتح الباب بدون استخدام المفتاح. أو أن تغير درجة حرارة المكيف دون أن تقف (لمكيف بدون ريموت) أو أن تستخدم الانترنت للتحكم بأجهزة كهربائية و أنت خارج المنزل.

جمع المعلومات : في كثير من التطبيقات يحتاج اتخاذ القرار إلى كم كبير من المعلومات . مثلاً نود بناء محطة طاقة شمسية أو محطة طاقة رياح . أين هو المكان الأمثل بعد دراسة لمدة سنة كاملة لمعرفة المكان الأمثل.

تقليل السعر : توجد منتجات كهربائية مرتفعة السعر، إذا استطعت إنتاج منتج مشابه بتكلفة أقل . فأنت رائع.

بإمكانك هذه اللحظة التوجه إلى صفحة البحث قووقل أو يوتيوب. و البحث عن Arduino projects ستجد مئات الأفكار بعضها استعراضية و بعضها ذات تطبيقات مفيدة في الحياة شاهد هذا الفيديو على سبيل المثال أيضاً شاهد : instructables موقع خاص بتنفيذ الأفكار و شرحها

مشاكل تصنيع المشاريع 🔨:

لا يتسع المجال هنا لشرح كيفية تصميم جهازك الإلكتروني و هل ستصنع دائرة إلكترونية؟ و كيف ستصنع الصندوق الحاوي لمكونات الجهاز . كيف ستثبت كل قطعة في مكانها. ستحتاج لمهارات فنية في تصميم الجهاز وتشكيل المواد و تجميع القطع ولحام العناصر الإلكترونية و التوصيلات. كل هذا يتجاوز مجال هذا الكتاب. وقد نناقشه في دورات و كتب قادمة.

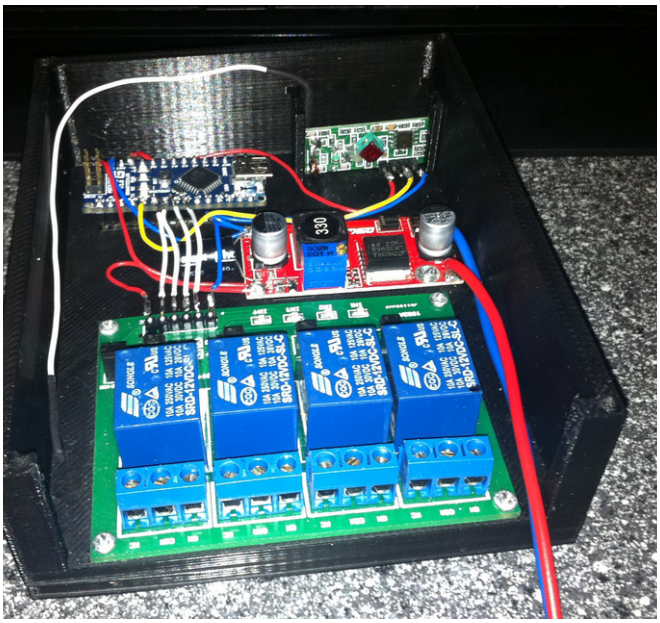
الرابط التالي لفيديو لعملية وضع دائرة إلكترونية في صندوق مناسب وتشكيل الواجهة:

Circuit Skills: Electronics Enclosures

من النصائح العامة لتقديم مشروعك بشكل لائق:-

1- ارسم وصمم الشكل قبل أن تبدأ بالتنفيذ

1-ايجاد صندوق بلاستيكي وتثبيت الأزرار اللازمة عليه فقط ، بحيث نخفي الدوائر الإلكترونية التي لا يحتاج المستخدم رؤيتها.



2-استخدم أنواع من الصمغ و اللواصق لوضع الأشياء مكانها و تثبيتها.

3-يمكنك الإستفادة من مكعبات الليقو LEGO كثيرا في عمليات التصنيع و التثبيت المنزلية.

4-جهز مكاناً (معملاً) يحتوي مئات المسامير و الصواميل، و العناصر الإلكترونية و الربطات.

5-ستحتاج كثيرا إلى تصنيع دائرة إلكترونية بدل توصيل الأسلاك على لوحة الاختبار دائماً.

أفكار سريعة لمشاريع:

تصميم إشارة مرور بسيطة : ستحتاج LEDs ملونة فقط و سيتحكم بتشغيلها الأردوينو (بسيط جدا)

تصميم إشارة للمشاة (إضافة زر للمشاة)

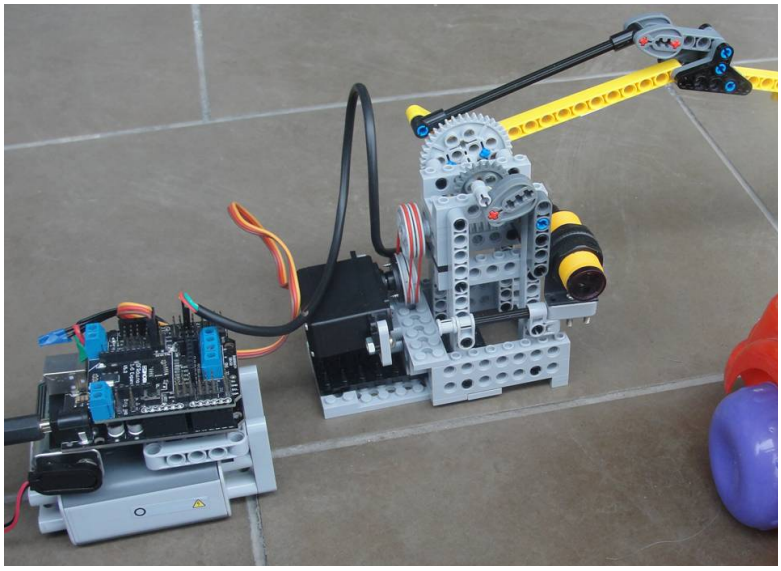
نظام يوفر الكهرباء بإطفاء الإضاءة و التكييف في حال عدم وجود أشخاص . مناسب للجامعات مثلاً

نظام لكشف تسريبات المياة

مصعد بسيط بحيث يكون الأردوينو هو المتحكم به (متقدم برمجيا)

نظام لزيادة السلامة في المسابح

إضافة خاصية التوقيت لأجهزة لا تحتوي عليها (مثل الشحن لساعة ثم الفصل) أو السخان .



التحكم بالجوال بالإضاءة و المكيف.

ربط سيارة إلكترونية بمتحكم (تطبيق جوال) أو يد بلايستيشن.

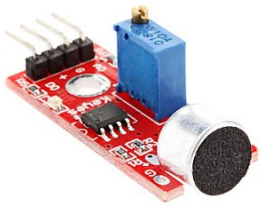
نظام سلامة في المطبخ مثل كشف تسريب الغاز.

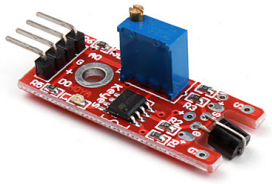
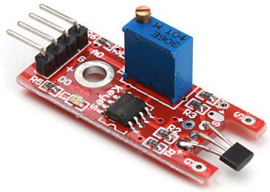
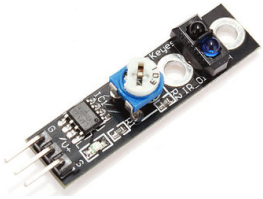
نظام تحريك الخلايا الشمسية حتى تتبع الشمس.

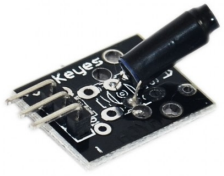
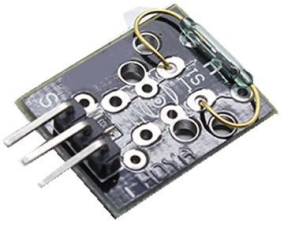
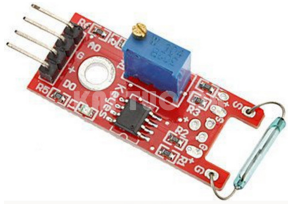
مشروعي أنا (راح أشتغل عليه قريباً) : اضافة خاصية (تعديل درجات الحرارة) أثناء النوم (لمكيف لا يحتوي على هذه الخاصية) و ربطها بالجوال. + التحكم بالإضاءة.

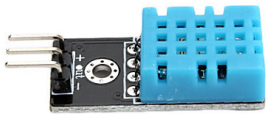
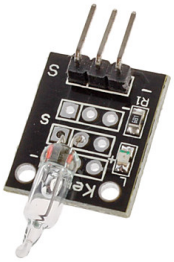
إضافات يمكن ربطها مع الأردوينو

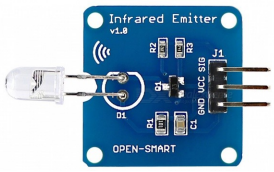
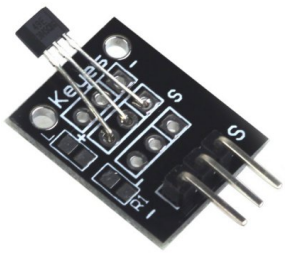
الصفحات التالية سنذكر عدد من الملحقات التي يمكن ربطها بالأردوينو . وباستخدامها يمكنك تنفيذ عدد كبير من المشاريع . و في مجالات كثيرة. قد تأتيك الفكرة و أنت تنظر إلى هذه الإضافات....

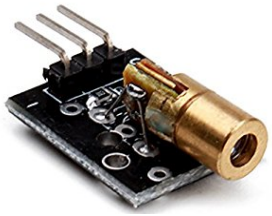
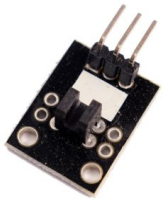
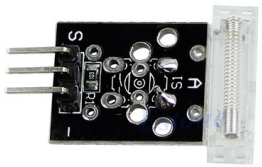
			
digital temp مقياس الحرارة الرقمي	flame sensor كاشف الحريق	mic - Sound حساس للصوت	joy stick عصا التحكم

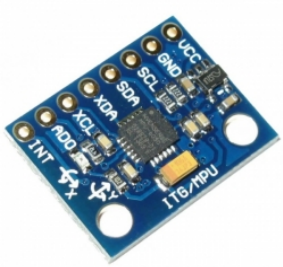
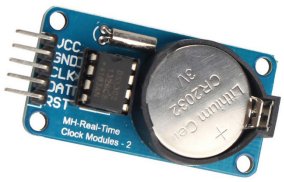
			
metal touch	linear hall كاشف المجال المغناطيسي	Avoidance حساس تلافي الاصطدام	tracking sensor حساس تتبع اللون

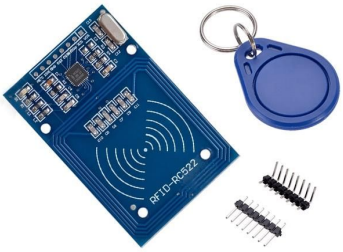
			
shook module كاشف اهتزازات	heart beat نبضات القلب	reed switch مفتاح مستشعر للمغناطيس	magnetic spring مفتاح يتأثر بالمغناطيس

			
Temperature and humidity مقياس حرارة و رطوبة	tilt switch **مفتاح ميلان**	Hydrargyrum-sw mercury SW	rotary encoder قياس سرعة واتجاه الدوران

			
IR receiver **مستقبل إشارة تحت حمراء**	IR Emission **مرسل إشارة تحت حمراء**	مقياس حرارة تماثلي	analog hall sensor قياس المجال المغناطيسي

			
مستقبل ليزر	مصدر ليزر	حساس ضوئي	tap / knock حساس الاهتزاز

			
مصفوفة من الـ LEDs	gyroscope حساس ميلان	PIR motion حساس حركة	RTC Real Time Clock ساعة لحفظ الوقت

		
حساس قياس الرطوبة في التربة	حساس للغازات السامة	RFDI لاسلكي قصير المدى

مجرد التأمل في الإضافات الكثيرة التي يمكن ربطها بالأردوينو قد يجعلك تجد أفكار عديدة قابلة للتحسين

موقع رائع لمشاهدة الإضافات للأردوينو	http://arduinomodules.info/

مواضيع متقدمة قد نضع شرحها مستقبلاً

استفدت أنا كثيرا و أنا أبحث في مواضيع هذا الكتاب، و أكتبه و أنسقه ، لكن مجالات الأردوينو أكثر بكثير.
إذا اتسع لي الوقت فسوف أنفذ دورة أخرى و أكتب كتاب آخر أتمنى أن يحتوي على المواضيع التالية والمزيد .

القراءة من ذاكرة SD

توصيل الأردوينو بالبلوتوث Bluetooth

توصيل الأردوينو بسلك الشبكة Ethernet

توصيل الأردوينو بالـ واي فاي

التحكم بالأردوينو عبر الإنترنت (جعل الأردوينو يعمل كسيرفر)

التواصل المتقدم مع المكونات والشرائح الإلكترونية SPI نموذجاً

استخدام الـ GPS

استخدام الساعة الدائمة RTC

استخدام شاشة الـ OLED

الاتصال مع شبكة الجوال GSM

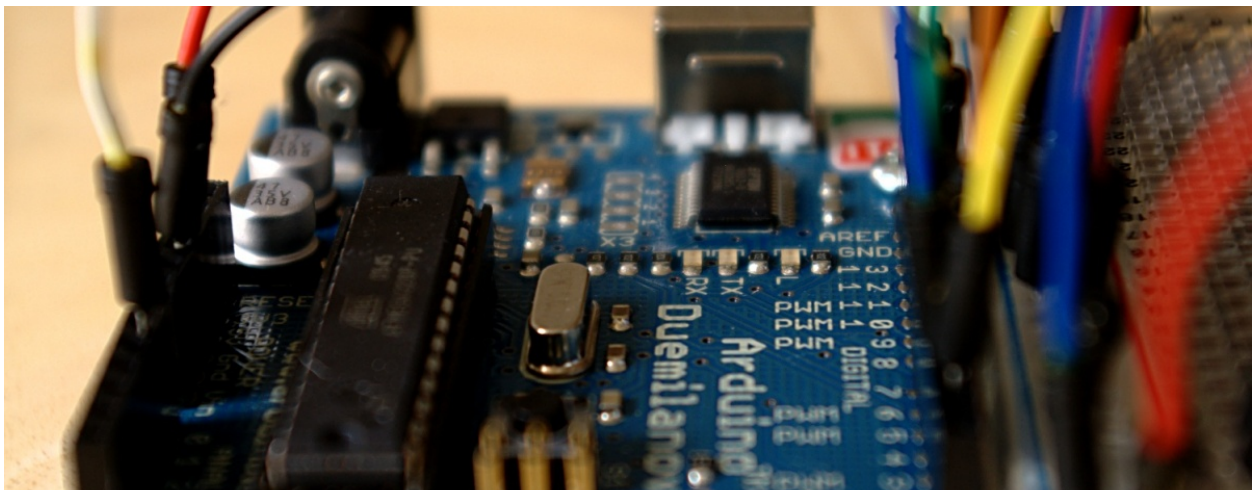
كيف تبني مكتبة Library خاصة بك

كيف تستخدم المداخل التماثلية كمنافذ رقمية

كيف تجعل الأردوينو يعمل كـ كيبورد أو ماوس

خاتمة:-

الأردوينو سهل الإستخدام (نوعاً ما) لكن لا حدود لاستخداماته وإمكانياته إلا الخيال ...
يمكنك أن تحل عشرات المشاكل في الحياة العامة باستخدام شريحة الأردوينو الجميلة والإضافات التي تعلمناها معاً. البرمجة هي لغة المستقبل ، هي اللغة التي نتحدث فيها مع الآلات مباشرة.
المشاريع الصغيرة تبني الخبرة و الثقة و تمرنك على أهم المهارات العملية التي يحتاجها المهندس في مسيرته المهنية: البرمجة ، الإلكترونيات ، الكهرباء ، القياس ، الاختبار، حلّ المشاكل والتحسين المستمر.
التواصل مع المهتمين بالمجال أهم مفاتيح التعلم و التطور : يوجد عدد كبير من المهتمين بالأردوينو عبر العالم. ستحتاجهم و سيحتاجوك . اللغة الانجليزية أيضاً مهمة جدا لايجاد الحلول بسرعة,



سوف أبقى معكم على تواصل بمشاريع ودروس و أخبار الأردوينو ...
لقد اخترت الأردوينو لأبدأ رحلتي في الأنظمة ذاتية التحكم الصغيرة. ولا أعلم إلى أين سأصل ...لكنني متأكد من شيء واحد ...
أنا استمتع بالرحلة كثيراً و أتمنى أن تستمتع بها معي . ✌

كتبه: **م. سامي قرامي**
في 2-8-2017 aug

تعريف بالمؤلف:

م.سامي محمد قرامي

مدرب هندسة إلكترونية منذ 2006

المؤسسة العامة للتدريب التقني و المهني بالمملكة العربية السعودية

المعهد الثانوي الصناعي بجدة

sami@jeem2.com +966554513632

المؤهلات العلمية:-

ماجستير في الهندسة الكهربائية _ جامعة كولورادو دينفر_2014

بكالريوس الكلية التقنية بالرياض_إلكترونيات صناعية و تحكم_2006

دبلوم كلية الاتصالات بجدة_إلكترونيات صناعية و تحكم_2003

المعهد الثانوي الصناعي بجدة_إلكترونيات صناعية_1999

دورات واهتمامات:-

الأنظمة الكمبيوترية الصغيرة (مثل الأردوينو و الرازبري باي)

البرمجة (البايثون ، الماتلاب، الـ C++)

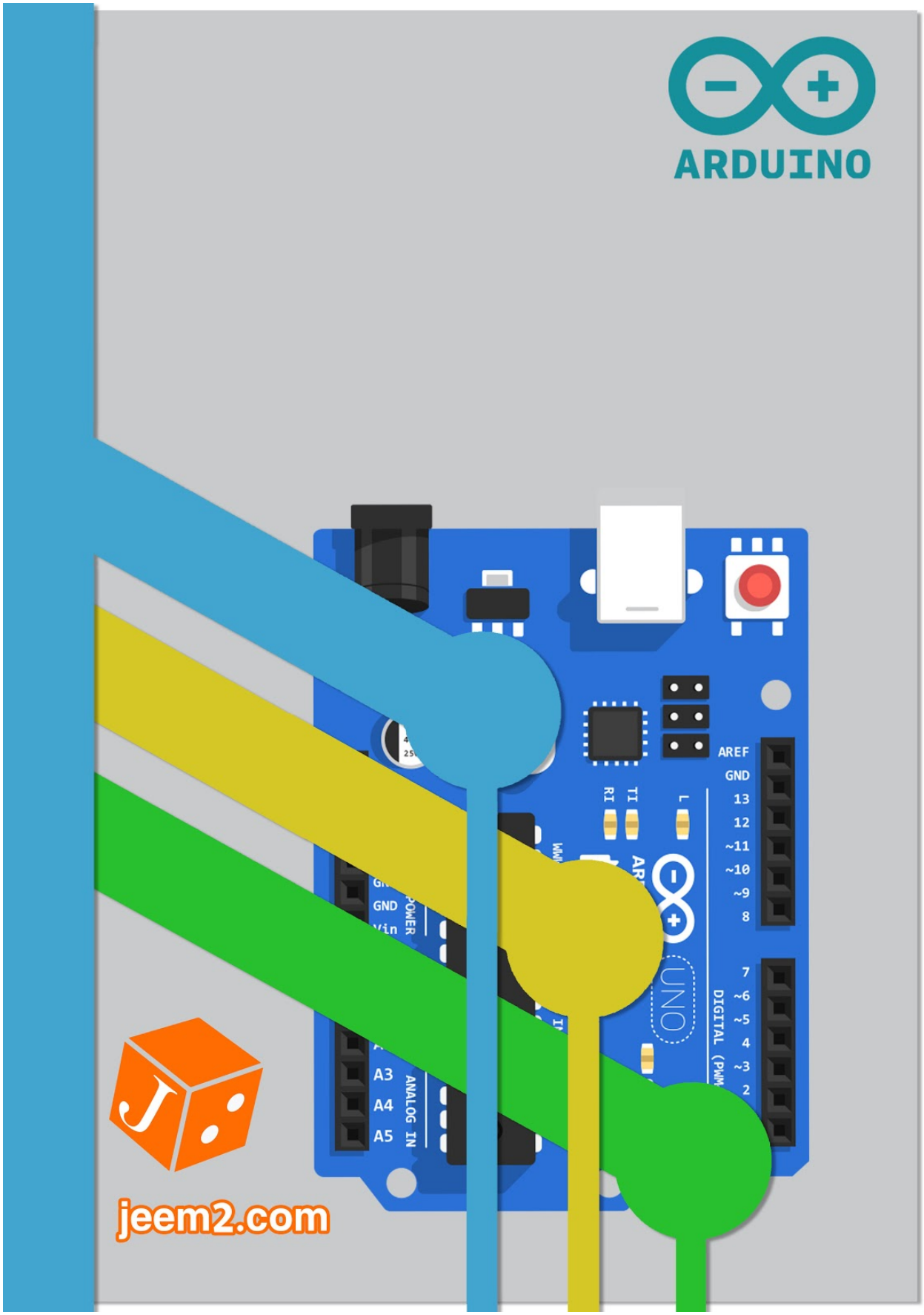
التعليم الإلكتروني - مؤسس موقع jeem2 التعليمي للهندسة

تصميم مواقع الانترنت (ووردبريس)

تصوير الفيديو + المونتاج + الفوتوشوب

التحدث (الخطابة) باللغتين العربية و الانجليزية

عضو مؤسس في نادي التوستماسترز بجدة (عربي/انجليزي)


ARDUINO
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
DIGITAL (PWM
A3
A4
A5
ANALOG IN
UNO
jeem2.com