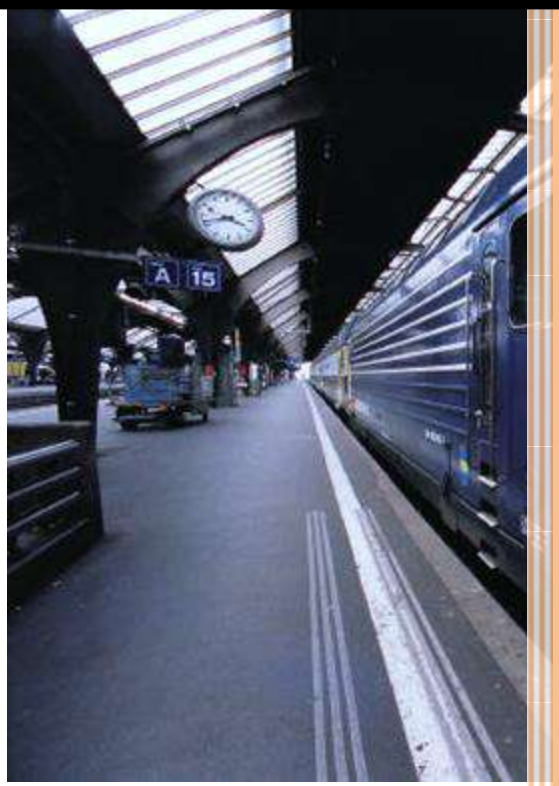
2017

Arduino C



Ahmed Ibrahim

1/1/2017

Contents

*****

هذا الكتاب من اعداد احمد ابراهيم ، وهناك كتب اخرى كتبها الكاتب لتعلم لغات برمجة مختلفة منها (HTML CSS, JavaScript, jQuery, PHP, Java) يمكنكم تحميلها وقراءتها من هذا الرابط :
https://drive.google.com/open?id=0B2aI_a6mphOUQi1jdzlYSFhITWs
كل هذه الكتب مجانية اي يمكن قراءتها واعادة نشرها .

لاي سؤال او استفسار يمكنكم التواصل مع الكاتب عبر الحساب الشخصي في فيسبوك :
https://www.facebook.com/ah.ib.93

كل برنامج يجب ان يتكون من ثلاثة مراحل:

المرحلة الاولى

في هذه المرحلة نعرف المتغيرات والثوابت التي سنستخدمها في البرنامج.

هناك عدة انواع من المتغيرات يمكن ان نتعامل معها في اردوينو، منها:

النوع	مثال	قيمة
int	int x = 5 ;	اي عدد صحيح بين سالب 32768 الى موجب 32768
unsigned int	unsigned int x = 44;	لا يأخذ قيمة سالبة والعدد يتراوح بين 0 الى 65535
word		يتراوح بين 0 الى 65535
Float	float x = 22.7 ;	يستخدم مع الاعداد التي تحتوي كسور
double		هو نفس المتغير float
char	char x = 'A' ; char x = 65;	حروف نصية، ويمكن وضع ارقام لتمثل الحروف وكودها يتراوح بين -128 الى 127
unsigned char	unsigned char x = 240;	يأخذ حروف كودها يتراوح بين 0 الى 255
long	long x = 99999999 ;	رقم طويل جدا يتراوح بين سالب 2,147,483,648 حتى موجب نفس الرقم
usingned long	usingned long x = 4343;	لا يأخذ قيمة سالبة والرقم يتراوح ما بين 0 و 4,294,967,295
byte	= 44;ءbyte	رقم صغير يتراوح بين 0 الى 255
boolean	bool x = true;	اما يحتوي على true او false

const

يقوم هذا الامر بعمل ثابت constant غير قابل للتغيير، مثال:

```
const int LED=13;
```

حيث ان اسم الثابت LED وقيمته 13 , اي اننا بذلك عرفنا ان المدخل رقم 13 في الاردوينو سيكون اسمه LED وفي اي مكان من البرنامج عندما نريد ان نتعامل مع هذا المدخل 13 سنذكر فقط الاسم LED وهو سيفهم اننا نقصد المدخل 13

مثال اخر:

```
const int sensor = A0 ;
```

هنا اسم الثابت sensor وقيمته A0 وهي تشير الى المنفذ التماثلي الموجود في بوردة الاردوينو

define

يستخدم هذا الامر للبحث والاستبدال حيث تعطى له قيمة واسم وأينما سيجد هذا الاسم سيقوم باستبداله بالقيمة وهذا يشمل حتى المسافة الفارغة، مثال:

```
#define NAME  563
```

ولاحظ اننا لم نضع فارزة منقوطة بعدها.

include

يقوم هذا الامر بتضمين ملف او مكتبة خارجية (من المكاتب الكثيرة الموجودة في اردوينو) حيث سنكتب اسم الملف او المكتبة وبعدها نقطة وبعدها h بين قوسي الامر المعقوفين , مثال :

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

في هذا المثال سيتم تضمين المكتبة التي اسمها Servo وهي مسئولة عن اوامر Servo Motor , ولاحظ اننا لم نضع فارزة منقوطة بعدها .

المرحلة الثانية

بما ان الـ Digital Pins في المتحكمات الدقيقة يمكن ان تستخدم كمخارج او كمداخل فيجب ان نبين ايها ستكون مداخل وأيها ستكون مخارج، وتكتب هذه المعلومات في داخل الدالة setup() , حيث ان الدالة setup() مسئولة عن ضبط اعدادات المتحكمة وتحديد المداخل والمخارج .

pinMode

هذا الامر مسئول عن تحديد وظيفة كل Pin وضبطها للعمل كمدخل او كمخرج، مثال:

```
void setup(){

pinMode(12, INPUT);

}
```

حيث ان 12 تمثل رقم الـ Pin ولاحظ انه يمكن كتابة اسم الـ Pin بدلا من رقمه اذا كنا قد عرفناه في المرحلة الأولى، والـ INPUT تعني انه سيكون مدخل وإذا اردنا ان يكون مخرج سنكتب بدلا منها OUTPUT , وانتبه لحالة الاحرف الكبيرة والصغيرة .

المرحلة الثالثة

هنا نكتب البرنامج نفسه وما نريد من المتحكمة فعله، وتكتب داخل الدالة loop , وبعد ان تنفذ الدالة الكود الموجود داخلها سوف تعود وتقرا وتنفذ الكود من جديد وتبقى تتكرر وتعيد نفسها طالما البوردة مزودة بالطاقة .

digitalWrite

يأخذ هذا الامر قيمتين بين قوسيه الاولى تمثل رقم الـ Pin او اسمه إذا كان معرف، والثانية تمثل القيمة التي ستخرج من هذا الـ Pin حيث ان HIGH ستعطي قيمة 5 فولت و LOW ستعطي قيمة 0 فولت، مثال:

digitalWrite(LED, LOW);

حيث ان LED تمثل اسم الـ Pin المعرف مسبقا.

digitalRead

يقوم هذا الامر بقراءة الجهد المدخل على الـ Pin المكتوب اسمه او رقمه بين قوسي هذا الامر وسيعيد القيمة HIGH اذا كان الجهد المسلط 5 فولت حتى 3.8 فولت او سيعيد القيمة LOW اذا كان الجهد المسلط صفر حتى 1.8 فولت، مثال:

X = digitalRead(13);

هنا سيقرأ قيمة الجهد المسلط على الـ Pin رقم 13 وسيخزن القيمة المعادة في المتغير X.

analogRead

يقوم هذا الامر بقراءة فرق الجهد بصورة تماثلية وتستطيع المتحكمة ان تقيس فرق جهد من 4.8 ملي فولت الى 5 فولت تقريبا وتقوم بتحويل الاشارة التماثلية الى قيمة رقمية من 0 الى 1023، مثال:

X = analogRead(A0);

هنا سيقيس فرق الجهد التماثلي على الـ Pin رقم A0 ويخزنه بشكل رقمي في المتغير X المعرف مسبقا.

مثلا إذا كان الجهد الداخل الى A0 يساوي القيم التالية:

4.8 ملي فولت = 1 رقمي

48 ملي فولت = 10 رقمي

480 ملي فولت = 100 رقمي

واحد فولت = 208.33 رقمي

2 فولت = 416.66 رقمي

5 فولت = 1023 رقمي

analogWrite

وهذا الامر مسؤول عن توليد خرج تماثلي ويمكن تطبيقها على المخارج التي تدعم خاصية التعديل الرقمي المعتمد على عرض النبضة، وهي المخارج التي يكتب بجانبها العلامة ~ وهي ست مخارج كالتالي:

11 , 10 , 9 , 6 , 5 , 3

خاصية التعديل الرقمي PWM

خاصية تمكنك من انتاج فرق جهد على هيئة موجة قابلة للتغيير عن طريق اشارة رقمية، وبذلك تستطيع ان تحول الاوامر الرقمية الى موجة تماثلية ويمكن استغلال تلك الخاصية في توليد جهد متغير قيمته بين 0 فولت و 5 فولت عن طريق استخدام قيم رقمية من 0 حتى 255، مثلا:

5 فولت = 255

2.5 فولت = 128

0 فولت = 0

مثال:

```
analogWrite(9, 128);
```

هنا سيخرج 2.5 فولت من الـ Pin رقم 9 ولاحظ ان الـ Pin يدعم الخرج التماثلي analog .

tone

تستخدم هذه الدالة لإخراج تردد معين من أحد المنافذ وتأخذ بين قوسيها ثلاثة قيم الاولى تمثل رقم الـ pin والثانية تمثل التردد والثالثة وهي اختيارية وهي تمثل المدة الزمنية بالملي ثانية المراد من الدالة ان تبقى تخرج هذا التردد من المنفذ واذا لم نعطي لها الوقت ستبقى تعمل الى ان نستخدم الدالة tone مرة اخرى او نستخدم الدالة notone، تستخدم هذه الدالة في الغالب لإخراج الصوت، مثال:

```
tone(9, 262, 500);

delay(500);
```

ملاحظة/ عند تشغيل هذه الدالة فإن PWM سوف لن يعمل على الـ Pin رقم 3 ورقم 11.

notone

تستخدم هذه الدالة لإلغاء تشغيل الدالة tone ويكتب بين قوسيها رقم الـ Pin الذي كان مستخدم في الدالة tone، مثال:

```
notone(9);
```

ملاحظة/ هذه الدالة قد تتطلب منا استخدام الدالة delay عند استخدام الدالة tone.

pulseIn

تستخدم هذه الدالة لتعيد المدة الزمنية (بالميكرو ثانية) لـ pin معين لكي يتغير من HIGH الى LOW او بالعكس، مثلا اذا كان pin معين LOW ستحسب هذه الدالة كم من الوقت بقي هذا الـ pin الى ان تحول الى HIGH، تأخذ بين قوسيها 3 بارامترات الأول هو عبارة عن رقم الـ pin الذي نريد حساب الوقت له الثاني هو يمثل نوع النبضة التي نريد قراءتها أي اما تكون HIGH او LOW اما الباراميتر الثالث فهو اختياري وهو يمثل الوقت بالميكرو ثانية لكي يبدأ وقت قياس النبضة (بشكل افتراضي عند عدم وضعه يكون الوقت هو ثانية واحدة)، مثال:

```
long duration = pulseIn(7, HIGH);
```

في هذا المثال المتغير duration سيحمل رقم يمثل الوقت بالميكرو ثانية لتغير حالة الـ pin رقم 7 من الـ LOW الى الـ HIGH.

دوال الوقت Time

delay

يستخدم هذا الامر لإخبار المتحكمة كم من الوقت يجب ان تبقى قبل تنفيذ الامر التالي وهي تأخذ بين قوسيها قيمة بالملي ثانية، الثانية الواحدة فيها 1000 ملي ثانية , مثال :

digitalwrite(13, HIGH);

delay(1000);

digitalwrite(13, LOW);

هنا سيتم اخراج جهد بقيمة 5 فولت من الـ Pin رقم 13 وسيبقى لمدة ثانية وبعدها لن يخرج اي قيمة (0 فولت) .

delayMicroseconds

هذا الامر يشبه الامر delay ولكن الفرق بينهم هو ان هذا الامر يأخذ بين قوسيه ميكرو ثانية , كل ثانية فيها 1000000 ميكرو ثانية .

millis

تحسب هذه الدالة الوقت منذ بدأ تشغيل الاردوينو بالملي ثانية وتبقى تعد الوقت لمدة 50 يوم تقريبا وبعدها تعود لتبدأ العد من جديد، مثال:

long X;

X = millis( );

- لاحظ انه استخدمنا متغير من نوع long لأنه ستعيد لنا رقم كبير يمثل الزمن بالملي ثانية والمتغير من نوع int لا يمكنه سوى استيعاب 32.767 ثانية.

micros

تحسب هذه الدالة الوقت منذ بدأ تشغيل البرنامج بالميكرو ثانية وتبقى تعد الوقت لمدة 70 دقيقة (حسب نوع الاردوينو) وبعدها تعود لتبدأ العد من جديد، مثال:

unsigned long time;

time = micros();

الدوال الرياضية Math

abs

تستخدم هذه الدالة لإيجاد القيمة المطلقة للرقم الذي يكتب بين قوسيها، مثال:

```
abs(x);
```

pow

تستخدم هذه الدالة لرفع رقم الى اس معين، وتأخذ بين قوسيها بارامترين الأول يمثل العدد والأخر يمثل الاس، يمكن ان يكون الرقم والاس من النوع float والدالة ستعيد قيمة من النوع double، مثال:

```
int x = pow(2, 5);
```

سيكون الناتج 32.

sqrt

تستخدم هذه الدالة لإيجاد الجذر التربيعي لأي عدد، مثال:

```
double x = sqrt(4);
```

سيكون الناتج 2.

constrain

هذه الدالة توضع بين قوسيها ثلاثة قيم، الاولى هي القيمة الاولية المراد العمل عليها، الثانية هي اصغر قيمة والثالثة هي اكبر قيمة، وهذه الدالة في داخلها كود كالتالي:

```
int constrain( int value, int min, int max) {
if( value > max ) {
value = max;
}
if( value < min ) {
value = min;
}
return value;
}
```

ولاستخدام هذه الدالة في برنامجنا نكتبها بهذه الطريقة، مثال:

```
void loop() {

int x = 0;

int y = constrain(x, 128, 255);

analogWrite(3, y);

}
```

في هذا المثال ستعيد الدالة القيمة 128 وتضعها في المتغير y.

map

تستخدم هذه الدالة للتغيير من مدى معين الى اخر، وهي مفيدة جدا لان analogRead تحول القيم في مدى من 0 الى 1023 ولكن الـ analogWrite تأخذ قيم في مدى من 0 الى 255 , الصيغة العامة لهذه الدالة هي

```
map( value, fromLow, fromHigh, toLow, toHigh );
```

مثال:

```
int X = 0;

int y;

X = analogRead(A0);
```

```
y = map(X, 0, 1023, 0, 255);

analogWrite(9, y);
```

sin

تستخدم هذه الدالة لإيجاد الجيب لزاوية معينه بالـ radians، وهي تعيد قيمة من النوع double بين 1 و -1 ، مثال:

```
double x = sin(45);
```

cos

تستخدم هذه الدالة لإيجاد الجيب تمام لزاوية معينه بالـ radians، وهي تعيد قيمة من النوع double بين 1 و -1 ، مثال:

```
double x = cos(45);
```

tan

تستخدم هذه الدالة لإيجاد الظل لزاوية معينه بالـ radians، وهي تعيد قيمة من النوع double بين سالب مالا نهاية وموجب مالا نهاية، مثال:

```
double x = tan(45);
```

الشروط وحلقات التكرار

- رموز المقارنة

الرمز	الشرط
==	يساوي
!=	لا يساوي
<=	اقل من او يساوي
>=	أكبر من او يساوي
<	اقل من
>	أكبر من

- التراكيب المنطقية

التركيب	المعنى	مثال
&&	and ستعيد القيمة TRUE إذا كان الشرط الاول والشرط الثاني صحيحين وبخلاف ذلك ستعيد القيمة FALSE	(A > 10) && (B < 5)
||	or ستعيد القيمة TRUE إذا كان أحد الشرطين او كلاهما صحيح وستعيد القيمة FALSE فقط اذا كان كلا الشرطين خاطئين	(A > 10) || (B < 5)
!	not ستعيد القيمة TRUE إذا كان الشرط خاطئ وبخلاف ذلك ستعيد القيمة FALSE	! (A > 10)

مثال:

```
int X = LOW;

void  loop( ) {

X = ! X;

digitalWrite(13, X);

delay(1000);

}
```

هنا عندما نعكس قيمة المتغير X والذي كان يحمل القيمة LOW ستكون قيمته HIGH.

if ... else

يستخدم هذا الامر لعمل مقارنات وجعل المتحكمة تفعل شيء ما اذا حدث شيء اخر محدد , مثال :

```
if ( X == HIGH ) {
digitalWrite(2, HIGH);
} else {
digitalWrite(2, LOW);
}
```

- يمكن كتابة فقط if بدون else إذا لم نكن بحاجة لها.

مثال اخر:

```
if ( digitalRead(13) == HIGH ) {
digitalWrite(2, HIGH);
} else if (digitalRead(12) == HIGH ) {
digitalWrite(2, LOW);
} else {
digitalWrite(3, HIGH);
}
```

وكما تلاحظ من المثالين يمكن كتابة الشرط بأكثر من طريقة بالإضافة الى انه يمكن عمل سلسلة غير منتهية من else if.

switch

والصيغة العامة لها:

```
switch (variable){
case value1 :
statement ;
break ;
case value2 :
statement ;
break ;
default :
statement ; }
```

حيث ان variable يمثل اسم المتغير المطلوب اجراء الاختبار على قيمته، ويشترط ان يكون من النوع int او char ,value1 و value2 عبارة عن قيم يتم مقارنتها مع قيم المتغير فاذا تطابقت قيمة المتغير مع قيمة احد هذه القيم ستنفذ الجمل التي بعدها الى حد عبارة break , اما في حال عدم تطابق قيمة المتغير مع اي من القيم الموضوعة ستنفذ الجمل بعد عبارة default، مثال:

```
int X = 4 ;
switch(X){
case 5 :
System.out.println(X);
break;
case 4 :
 System.out.println(X);
 break;
 default :
  System.out.println("None");  }
```

يمكن ان تكون switch في داخل switch اخرى عند كتابتها في الجملة case، مثال:

```
switch(ch1) {
```

```
case 'A': System.out.println(" This A is part of outer switch. ");
switch(ch2) {
case 'A':
System.out.println(" This A is part of inner switch ");
break;
case 'B': // ...
} // end of inner switch
break;
case 'B': // ...
```

for

وهي حلقات تكرار تبقى تدور الى عدد معين طالما الشرط بين قوسيها متحقق، مثال:

```
for( i = 0;  i < 255;  i++ ) {

analogWrite(9, i);

delay(10);

}
```

حيث هنا المتغير i المعرف مسبقا يبدأ بالقيمة 0 وسيزداد في كل مرة واحد وسيبقى الشرط متحقق طالما المتغير i أصغر من 255 وكل مرة سينفذ الامر ما بين القوسين المعقوفين .

while

وهي حلقة تكرار تبقى تدور طالما الشرط بين قوسيها متحقق، مثال:

```
int X = 20;

While ( X > 10 ) {
```

```
X = X – 1;

digitalWrite(9, HIGH);

}
```

do while

الصيغة العامة لها هي:

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

break

تستخدم هذه العبارة للخروج من حلقة التكرار او من البلوك { الاقواس المعقوفة }، مثال:

```
int t = 0;
while( t < 100 ) {
if(t == 10) break;
System.out.println( t );
t++; }
```

continue

تستخدم هذه العبارة للخروج من تكرار معين وليس من حلقة التكرار بالكامل، مثال:

```
int i;
for(i = 0; i<=100; i++) {
if((i%2) != 0) continue;
System.out.println(i); }
```

في هذا المثال سيطبع فقط الارقام الزوجية

goto

تستخدم لنقل البرنامج الى مكان مسمى مسبقا، مثال:

```
for(byte r = 0; r < 255; r++){
   for(byte g = 255; g > 0; g--)
        if (analogRead(0) > 250){ goto bailout;}
         // more statements ...
   }
}
bailout:
```

المصفوفات Arrays

يمكن عمل المصفوفات والتي تحتوي بداخلها على عدة متغيرات وبين قوسيها نكتب عدد المتغيرات التي بداخلها، ولاحظ ان ترتيب المتغير الاول هو صفر والثاني هو واحد: مثال/

```
const int X = 4;
const int Ar[X] = { 2, 3, 4, 5 };
void setup (){
for ( int i = 0; i < X; i++ ){
pinMode(Ar[i], OUTPUT);
} }
void loop(){
for ( int i = 0; i < X; i++ ){
digitalWrite( Ar[i], HIGH );
delay(1000);
}
for ( int i = X - 1; i >= 0; i-- ){
```

```
digitalWrite( Ar[i], LOW );
delay(1000);
} }
```

انشاء دالة function

يمكننا انشاء دالة خاصة بنا نكتب اسمها ونضع بين قوسيها اسماء المتغيرات التي ستستخدم داخلها وبين قوسيها المعقوفين نكتب ما نريد من الدالة عمله، اذا لم تكن هذه الدالة تعيد اي شيء سنضع قبل اسمها void، مثال:

```
#define NOTE_C4  262
Const int KPin = 9;
void setup(){
pinMode(KPin, OUTPUT);
}
void loop(){
myFunction(NOTE_C4, 500);
noTone(KPin);
delay(2000);
}
void myFunction(int X, int y){
tone(KPin, X, y);
delay(y);
}
```

او اذا كانت الدالة التي نريد عملها تعيد لنا قيمة فسنكتب نوع القيمة (المتغير) الذي ستعيدها بدلا من void ونكتب داخل الدالة return وبعدها القيمة التي نريد الدالة اعادتها , لاحظ هذا المثال لدالة تعيد قيمة من نوع float :

```
float myFunction(){
int X = analogRead(13);
float y = X / 3;
return ( y + 5.4 ) * 3;
}
```

ولاستدعاء هذه الدالة ووضع القيمة التي تعيدها في متغير نكتب:

```
float Name = myFunction();
```

مجرى البيانات Serial

- لمزيد من الاوامر المستخدمة في Serial communication اذهب الى هذا الرابط

https://www.arduino.cc/en/Reference/serial

مجرى البيانات Serial هو عبارة عن منفذ اتصال بين الاردوينو والأجهزة الأخرى، مثلا يمكن ان يكون المنفذ للاتصال بالحاسوب عبر الكيبل من خلال الـ serial monitor ، او الاتصال عبر البلوتوث.

هذا المنفذ سيرسل أي حرف نرسله من الاردوينو على شكل رقم الـ ASCII الخاص بالحرف مثلا إذا كتبنا a سيرسل الرقم 97 اما إذا ارسلنا الرقم صفر سيرسل الرقم 48 وهكذا.

للـ serial عدة دوال منها:

الدالة begin

تستخدم هذه الدالة لفتح منفذ الاتصال، وهي تأخذ بين قوسيها رقم يمثل رقم المنفذ حيث ان لكل نوع من الاتصال له رقم خاص به، وهذه الدالة اجبارية إذا كنا نريد ان نعمل اتصال ويجب ان تكتب في المرحلة الثانية أي في داخل الدالة void setup، مثال:

```
Serial.begin(9600);
```

الرقم 9600 يمثل منفذ الاتصال بالحاسوب عبر الـ serial monitor ويستخدم نفس هذا المنفذ للاتصال عبر البلوتوث HC-06 اما قطعة البلوتوث HC-05 رقم المنفذ لها هو 38400.

الدالة available

تستخدم هذه الدالة للتحقق من ان منفذ الاتصال يستقبل البيانات ويتم مقارنتها بالصفر فاذا كانت أكبر من صفر فان هناك بيانات قادمة من الجهاز الاخر، مثال:

```
if(Serial.available() > 0){ // Checks whether data is coming from the serial port
   // do some thing
}
```

الدالة print

تستخدم هذه الدالة لإرسال قيم الى الحاسب الالي (او أي جهاز سيستقبل البيانات) لعرضها على بيئة التطوير باستخدام Serial Monitor او حسب نوع الجهاز المستلم، مثال :

```
Serial.print(voltage);
```

حيث ان voltage هو متغير معرف مسبقا يحمل قيمة فولتية، وهنا سترسل قيمة هذا المتغير ليستقبلها الجهاز الاخر (وإذا كانت عبر الـ serial monitor فان قيمة المتغير ستعرض على الشاشة).

الدالة printIn

تستخدم هذه الدالة لأرسال قيمة عبر منفذ الاتصال للجهاز المتصل بالاردوينو من خلال الـ USB على بيئة التطوير Serial Monitor او عبر البلوتوث، فرقه عن الدالة السابقة هو ان هذا الامر يطبع في سطر جديد، مثال:

```
Serial.printIn(“ Hello “);
```

الدالة read

يستخدم هذا الامر لاستلام قيم من الجهاز الاخر المتصل بالاردوينو (الحاسب الالي او غيره) الى الاردوينو عبر المنفذ المفتوح، مثال:

```
int X = Serial.read();
```

هنا سيتم تخزين القيمة المرسلة من الجهاز الاخر (الحاسوب او غيره) في المتغير X ، لاحظ ان المتغير x هو من النوع int لأنه أي حرف او رقم يرسله الجهاز الاخر سيتم تحويله الى رقم الـ ASCII الخاص به، وسيتم استقبال كل حرف على حدة أي حرف بعد حرف.

الدالة readString

يستخدم هذا الامر لاستلام قيم من الجهاز الاخر المتصل بالاردوينو (الحاسب الالي او غيره) الى الاردوينو عبر المنفذ، مثال:

```
String X = Serial. readString ();
```

هنا سيتم تخزين القيمة المرسلة من الحاسوب في المتغير X ، لاحظ ان المتغير x هو من النوع String لأنه أي حرف او رقم يرسله الجهاز الاخر سيتم استقباله على شكل متغير نصي.

مثال متكامل

هذا المثال يستخدم مجرى البيانات Serial لإرسال واستلام البيانات عبر البلوتوث حيث يشيل LED مربوط على الـ pin رقم 7 اذا استلمنا الرقم 1 وعندها سيرسل الى جهاز الأندرويد كلمة ON اما إذا استلمنا الرقم 0 سيطفئ الـ LED ويرسل كلمة OFF ، مثال برنامج الاندرويد تجده مكتوب في كتاب الاندرويد الذي كتبته انا.

```
#define ledPin 7

int state = 0;

void setup() {

  pinMode(ledPin, OUTPUT);

  digitalWrite(ledPin, LOW);

  Serial.begin(9600); // Default communication rate of the Bluetooth module HC-06

}

void loop() {
```

```
 if(Serial.available() > 0){ // Checks whether data is comming from the serial port
   state = Serial.read(); // Reads the data from the serial port
}
if (state == '0') {
 digitalWrite(ledPin, LOW); // Turn LED OFF
 Serial.print("OFF"); // Send back, to the phone, the String "LED: ON"
 state = 0;
}
else if (state == '1') {
 digitalWrite(ledPin, HIGH);
 Serial.print("ON");
 state = 0;
}
}
```

Sensors

Ultrasonic Sensor

هو عبارة عن متحسس للموجات فوق الصوتية، ومن خلاله يمكن تحسس وجود او بعد الاجسام عنه، به اربع اطراف، الأول vcc يأخذ جهد 5v الثاني trigger الطرف المتحكم بسماعة اصدار الصوت Output الثالث Echo الطرف المتحكم بسماعة استلام الصوت Input الرابع هو Gnd يربط بالأرضي.

لحساب بعد جسم عن المتحسس نصدر موجة عبر المنفذ trigger لمدة زمنية قصيرة بالميكرو ثانية ومن ثم نراقب المنفذ Echo (والذي نضعه بشكل افتراضي LOW) من خلال الدالة pulseIn لنحسب الوقت الذي استغرقه الصوت للذهاب والاصطدام بالجسم ثم العودة، لذا لحساب المسافة نضرب في سرعة الصوت ونقسم على 2.

مثال: لقياس بعد الاجسام بالسنتمتر عن المتحسس وارسالها الى الـ Serial Monitor كل نص ثانية.

```
// defines pins numbers

const int trigPin = 9;

const int echoPin = 10;

// defines variables

long duration;

int distance;
```

```
void setup() {
pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
pinMode(echoPin, INPUT); // Sets the echoPin as an Input
Serial.begin(9600); // Starts the serial communication
}
void loop() {
// Clears the trigPin
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
// Sets the trigPin on HIGH state for 10 micro seconds
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
// Reads the echoPin, returns the sound wave travel time in microseconds
duration = pulseIn(echoPin, HIGH);
// Calculating the distance
distance= duration*0.034/2;
// Prints the distance on the Serial Monitor
Serial.print("Distance: ");
Serial.println(distance);
delay(500);
```

}

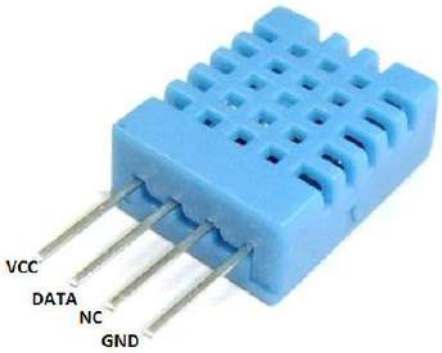
Buzzer

هو عبارة عن سماعة لإصدار الصوت والنغمات، لتشغيله نربط الـ pin الذي بجانبه علامة الناقص بأحد بنات الديجيتال للاردوينو اما الطرف الاخر الذي بجانبه الحرف S نربطه بالأرضي.

مثال:

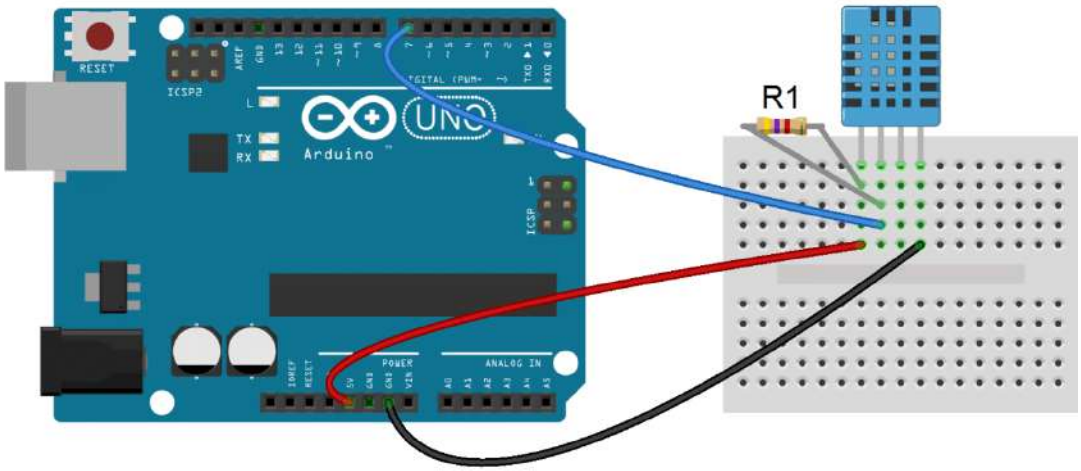
```
const int buzzer = 9;
void setup() {
  pinMode(buzzer, OUTPUT);
}
void loop() {
  digitalWrite(buzzer, HIGH);
  delay(50);
  digitalWrite(buzzer, LOW);
  delay(50);
}
```

DHT (Temperature & humidity)



يستخدم هذا المتحسس لقياس درجة الحرارة والرطوبة، يوجد نوعين رئيسيين منه هما DHT11 و DHT22، ولنتعامل معه علينا ان نضيف في الأول المكتبة الخاصة به (توجد عدة مكتبات للتعامل مع هذا المتحسس لكننا سنعتمد على المكتبة SimpleDHT والتي شرحنا عنها في قسم المكتبات في هذا الكتاب).

اذا كان هذا الحساس مربوط بلوحة فكل ما علينا هو توصيل أطرافه بأطراف الاردوينو مباشرتا، اما اذا لم يكن مربوط بلوحة (كما في الصورة في الأعلى) هنا سنوصل الطرف VCC مع 5 او 3 فولت للاردوينو والطرف GND مع الأرضي للاردوينو والطرف DATA مع احد اطراف الـ digital للاردوينو ونترك الطرف NC بدون ربط، ثم نربط الطرف DATA والطرف VCC مع بعض عبر مقاومة من 5 الى 10 كيلو اوم، ليكون الربط بهذا الشكل:

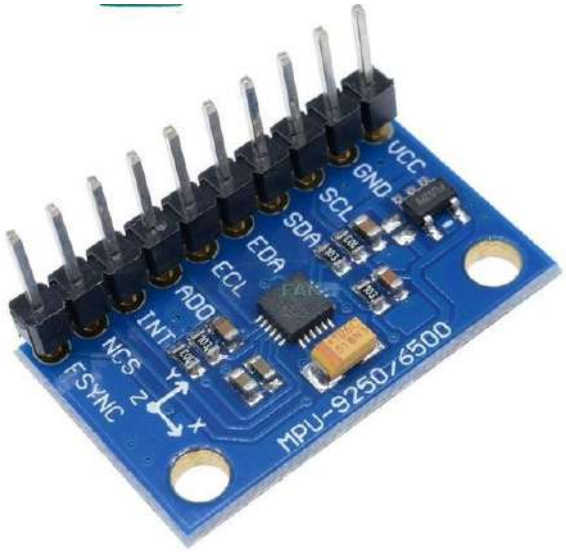


مثال: هذا المثال يعرض درجة الحرارة والرطوبة على شاشة الـ Serial Monitor كل واحد ثانية:

```
#include <SimpleDHT.h>
```

```
SimpleDHT11 s;
void setup() {
  Serial.begin(115200);
}
void loop() {
  byte temp = 0;
  byte hum = 0;
  s.read(2, &temp, &hum, NULL);
  Serial.print(temp); Serial.print(" *C, ");
  Serial.print(hum); Serial.println(" H");
  delay(1500);
}
```

MPU-9250/6500

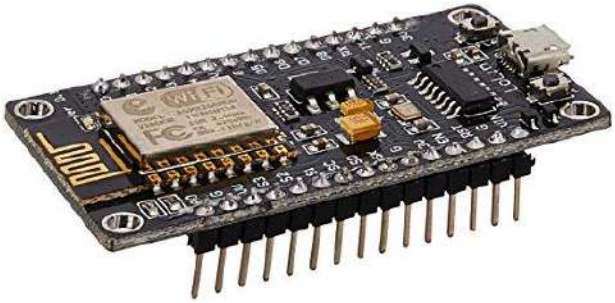


هذا المتحسس هو عبارة عدة متحسسات مع بعض حيث انه يحتوي على شريحة MPU-6050 وهذه الشريحة هي عبارة عن gyroscope و accelerometer ، كذلك يحتوي هذا المتحسس على شريحة أخرى وهي AK8963 وهي عبارة عن compass ، وكذلك يحتوي هذا المتحسس على حساس لدرجة الحرارة.

لكي نتعامل مع هذا المتحسس نربط المنفذ VCC بـ 3 او 5 فولت والمنفذ GND بالأرضي والمنفذ SCL بالمنفذ SCL للاردوينو والمنفذ SDA بالمنفذ SDA بالاردوينو، ويمكن ان نترك باقي المنافذ بدون ربط، العنوان address الافتراضي لهذا المتحسس هو 0x68 ويمكن تغييره الى 0x69 إذا ربطنا المنفذ ADO/SDO بـ 5 فولت، ولكي نتعامل بسهولة مع هذا المتحسس نضيف المكتبة الخاصة به MPU9250، الشرح الكامل لهذه المكتبة موجود في قسم المكتبات من هذا الكتاب.

ملاحظة/ يمكن التعامل مع هذا المتحسس من دون المكتبة MPU9250 وذلك من خلال ارسال واستلام البيانات باستخدام البروتوكول I2C من خلال المكتبة wire شرحنا عنها في قسم المكتبات.

NodeMCU ESP8266



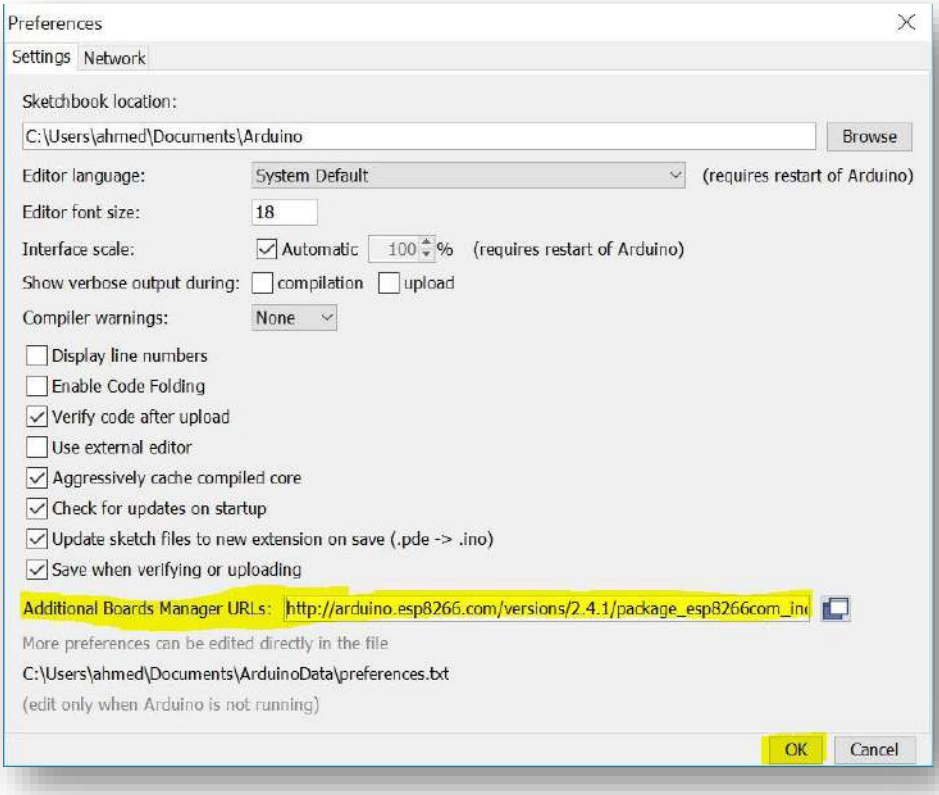
هذه هي لوحة Wi-Fi لمشاركة الملفات بين الأجهزة او عبر الانترنت، أولا علينا ان نثبت البرنامج التالي لكي يتمكن الحاسوب من التعرف على هذه القطعة عند ربطها عبر الـ usp، نجد البرنامج عبر الرابط التالي:

https://www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers

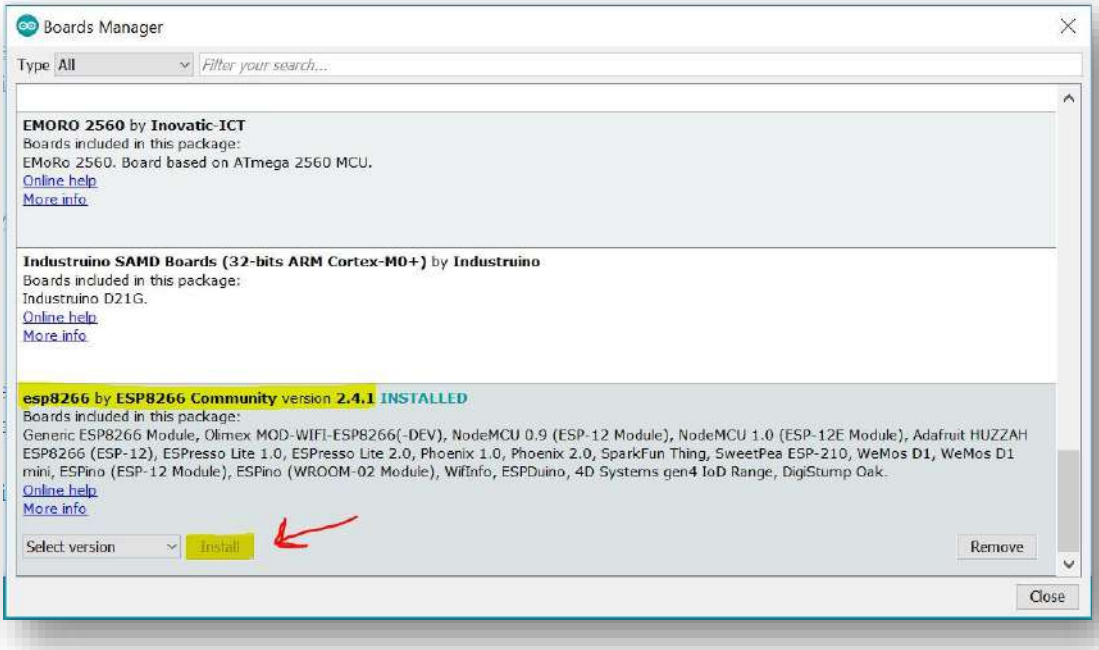
يمكن برمجتها من نفس برنامج الاردوينو لكن علينا في البداية ان نضيف هذه اللوحة الى البرنامج، لإضافتها نفتح برنامج الاردوينو ثم نذهب الى القائمة file منه نختار preferences لتظهر لنا نافذة في اسفلها حقل اسمه Additional Boards Manager URLs نضيف فيه هذا الرابط

http://arduino.esp8266.com/versions/2.4.1/package_esp8266com_index.json

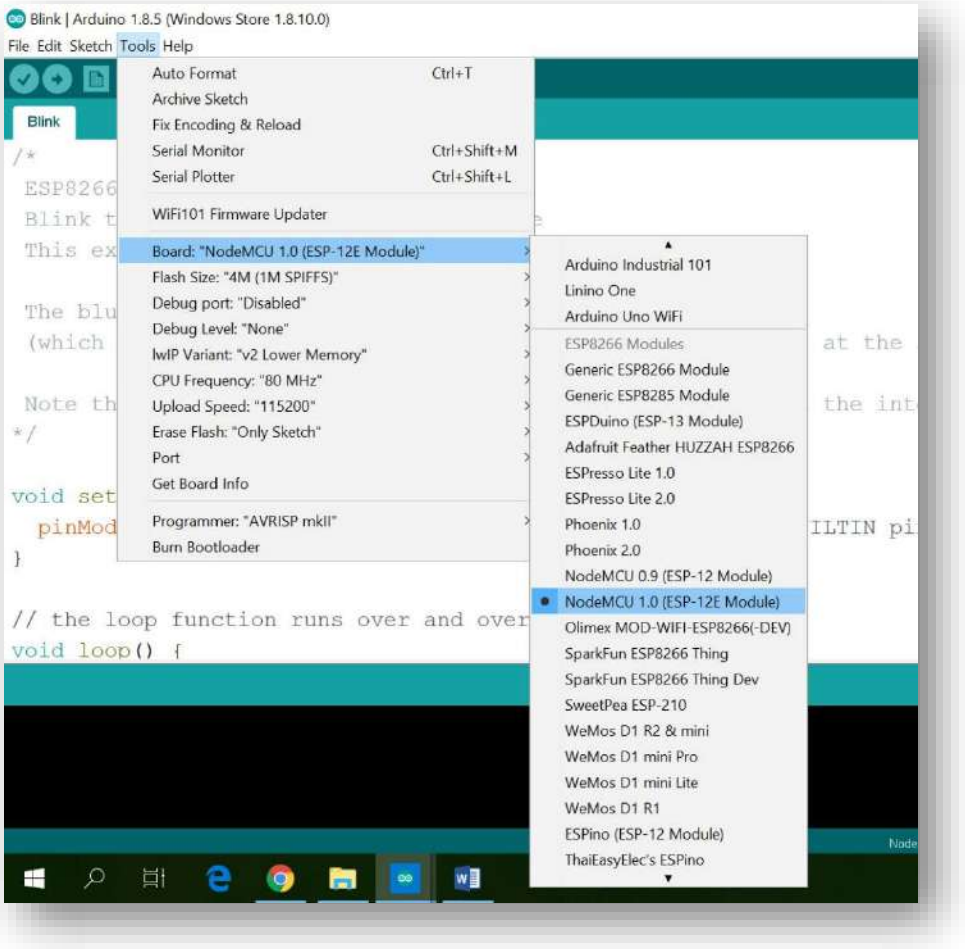
ثم نضغط على موافق، كما موضح في هذه الصورة:



بعد ذلك نذهب الى القائمة tools ومن داخل الخيار Board نختار Boards Manager لتظهر لنا نافذة نبحث فيها عن بوردة esp8266 ثم نضغط على install كما موضح في الصورة التالية:



بعد انتهاء عملية التحميل والتنصيب نعود مرة أخرى الى القائمة tools ثم من الخيار Board نختار بوردة NodeMCU كما موضح في هذه الصورة:



بعد ربط بوردة الـ Wi-Fi بالحاسوب لا تنسى تحديد الـ port المناسب من داخل برنامج الاردوينو الموجود في القائمة tools.

- لمعلومات أكثر عن طريقة تثبيت النظام على اللوحة وتفعيلها يمكن قراءة هذا الكتاب:

https://drive.google.com/open?id=1E_HVRFfaxG4_wmqfXkNaOLRiGKWuvceG

Libraries

- لمزيد من المعلومات عن المكتبات وأوامرها زر هذا الرابط

https://www.arduino.cc/en/Reference/Libraries

Servo Motor

لاستخدام اوامر هذه المكتبة يجب ان نضمنها في بداية الكود أي في المرحلة الاولى من خلال الامر #include<Servo.h> وبعدها يجب ان نعمل اوبجكت من هذه المكتبة للتعامل مع هذه الموتور من خلال كتابة كلمة Servo بعدها الاسم الذي نريد ثم فارزة منقوطة، وبنفس هذه الطريقة يمكن ان نسمي الموتور الثاني والثالث إلخ إذا كانوا عندنا، ومن خلال هذا الاسم سيمكننا ان نوجه الامر للموتور المقصود وليس لموتور اخر في نفس الدائرة.

الامر attach

يستخدم هذا الامر لنعرف على اي Pin تم ربط هذا الموتور الذي سنكتب اسمه، ويكتب هذا الامر في المرحلة الثانية أي داخل الدالة void setup، مثال:

```
#include<Servo.h>
Servo NameServo1;
const int X = 9;
void setup(){
NameServo1.attach(X);
}
void loop(){    }
```

الامر write

وهي تستخدم لتحريك الموتور الى زاوية معينة وتأخذ قيم من 0 الى 180 , ويكتب هذا الامر في المرحلة الثالثة بداخل الدالة void loop، مثال:

```
#include<Servo.h>
Servo NameServo1;
const int X = 9;
const int y = A0;
void setup(){
NameServo1.attach(X);
}
void loop(){
int val = analogRead(y);
val = map(val, 0, 1023, 0, 180);
NameServo1.write(val);
delay(15);
}
```

الامر detach

يستخدم هذا الامر عندما لا نريد توجيه اوامر كثيرة الى موتور معين فنعطيه هذا الامر بعد ان ينفذ الاوامر المطلوبة منه وذلك حفاظا على الطاقة، مثال:

```
NameServo1.detach();
```

ESP8266 Wi-Fi

تستخدم هذه المكتبة مع شريحة NodeMCU ESP8266 (شرحنا عنها سابقا)، لتضمين هذه المكتبة واستخدام اوامرها نكتب #include<ESP8266WiFi.h> في بداية البرنامج.

- هذه المكتبة مشابهه الى حد كبير الى المكتبة Wi-Fi والتي يمكن ان نجد كل تفاصيلها على هذا الرابط:

https://www.arduino.cc/en/Reference/WiFi

تحتوي هذه المكتبة على مجموعة من الاوبجكتس والكلاسات والاوامر والتي سنشرحها بعد قليل.

WiFi

يحتوي هذا الاوبجكت على مجموعة من الأوامر المهمة لاتصال الـ Wi-Fi منها:

begin

يستخدم هذا الامر لفتح اتصال Wi-Fi مع الشبكة ويأخذ بين قوسيه اسم الشبكة (اسم الراوتر) وكلمة السر الخاصة به، مثال:

```
WiFi.begin(“TP-LINK”, “ 12345678”);
```

config

تستخدم هذه الدالة لوضع عنوان الـ ip الخاص بالاتصال، وهي تكتب قبل الدالة begin فعندما تنفذ الدالة begin سوف تستخدم هذا العنوان للاتصال اما إذا لم نستخدم هذه الدالة config او وضعناها بعد الدالة begin فانه سيفتح اتصال بعنوان ip عشوائي.

الدالة config تأخذ بين قوسيها 4 بارامترات لكنها اختيارية وواحد منها فقط اجباري وهو يمثل عنوان الـ ip، الصيغة العامة لهذه الدالة هي:

```
WiFi.config(ip, dns, gateway, subnet);
```

مثال:

```
IPAddress ip(192, 168, 0, 177);
```

```
IPAddress gateway(192, 168, 0, 1);
IPAddress subnet(255, 255, 255, 0);
void setup() {
 WiFi.config(ip, gateway, subnet);
 WiFi.begin("TP-LINK", "12345678");
 while (WiFi.status() != WL_CONNECTED) {
 delay(500);
  }
}
```

في هذا المثال اخترنا الـ ip 192.168.0.177 ليكون هو منفذ الاتصال.

status

يستخدم هذ الامر لمعرفة حالة الاتصال بالشبكة هل تمت بنجاح ام لا، إذا حدث الاتصال بنجاح ستعيد القيمة WL_CONNECTED، مثال:

```
if( WiFi.status() != WL_CONNECTED ) {
        // do something
}
```

localIP

تستخدم هذه الدالة لتعيد عنوان الـ ip الذي تم فتحه، مثال:

```
IPAddress ip;
ip = WiFi.localIP();
```

RSSI

تستخدم هذه الدالة لتعيد قوة إشارة الاتصال بالراوتر، تعيد قيمة من النوع long، مثال:

```
long rssi = WiFi.RSSI();
```

SSID

تستخدم هذه الدالة لتعيد اسم الشبكة (الراوتر) المتصلين به حاليا.

scanNetworks

تستخدم هذه الدالة للبحث عن شبكات الاتصال المتاحة، وتعيد رقم يمثل عددها، هذا العدد يكون من النوع البياني byte، مثال:

```
byte numSsid = WiFi.scanNetworks();
```

WiFiServer

هذا الكلاس ينشأ خادم من خلاله يمكن التحكم بعملية الاتصال من ارسال واستلام البيانات الى العميل المتصل (العميل المتصل هو البرنامج الذي يعمل على الجهاز في الطرف الاخر من عملية الاتصال)، ينشأ خادم للإصغاء الى عمليات الاتصال القادمة على منفذ port محدد، مثال:

```
WiFiServer server(80);

void setup() { }

void loop() { }
```

begin

تخبر الخادم ان يبدأ الاصغاء إلى عمليات الاتصال القادمة، مثال:

```
WiFiServer server(80);

void setup() {

WiFi.begin(“TP-LINK”, “12345678”);

if ( WiFi.status() != WL_CONNECTED ) {

   while(true);

} else {
```

```
  server.begin();
}
void loop() { }
```

available

تستخدم هذه الدالة لمعرفة فيما إذا كان الخادم العميل متاح للقراءة وهي تعيد اوبجكت من النوع Client اما إذا لم تجده فأنها تعيد false، مثال:

```
WiFiServer server(80);
void setup() {
WiFi.begin(“TP-LINK”, “12345678”);
if ( WiFi.status() != WL_CONNECTED ) {
  while(true);
} else {
  server.begin();
}
void loop() {
WiFiClient client = server.available();
if (!client) {
return;
}
…
}
```

write

تستخدم هذه الدالة للكتابة على كل العملاء المتصلين بالخادم، الصيغة العامة لها هي:

```
server.write(data)
```

حيث ان data تمثل أي بيانات من النوع char او byte، يمكن ان تعيد هذه الدالة عدد البايتات التي تمت كتابتها لكنه ليس من الضروري قراءة هذه البايتات.

print

تستخدم هذه الدالة للكتابة على كل العملاء المتصلين بالخادم، وهي ترسل البيانات على شكل ارقام متسلسلة من الـ ASCII كود (مثلا الرقم 123 سيتم ارساله على شكل ثلاثة حروف '3' '2' '1')، الصيغة العامة لها هي:

```
server.print(data)
```

حيث ان البارامیتر data يمكن ان يمثل أي بيانات من النوع (char, byte, int, long, or string).

println

هي تماما نفس الدالة السابقة print لكن الفرق الوحيد هو ان هذه الدالة بعد انتهائها ستنزل سطر.

WiFiClient

كلاس من خلاله يتم انشاء عميل يستطيع الاتصال بموقع انترنت محدد، يتم تحديده من خلال عنوان الـ ip والمنفذ port وذلك باستخدام الدالة connect (سنشرح عنها بعد قليل).

```
WiFiClient client;
```

connected

تعيد القيمة true إذا حدث اتصال مع العميل وتعيد القيمة false إذا لم يحدث الاتصال، لاحظ انه يعتبر الاتصال مع العميل موجود إذا اغلق الاتصال لكن هناك بيانات لم تقرئ بعد، مثال:

```
client.connected()
```

connect

تستخدم هذه الدالة لإجراء اتصال مع عنوان ip عبر منفذ port معين، لاحظ انه يمكن وضع عنوان الموقع (اسم الـ domain) مثلا google.com بدلا من عنوان الـ ip، ونفس هذه الدالة تعيد القيمة true إذا نجح في الاتصال وتعيد القيمة false إذا فشل، مثال:

```
if (client.connect(servername, 80)) {

//do something

}
```

write

تستخدم هذه الدالة للكتابة على الخادم المتصل به العميل، وتأخذ بين قوسيها باراميتر واحد يمثل البيانات التي سترسل وهي تكون اما من النوع char او من النوع byte، وهي تعيد عدد الحروف التي تم ارسالها ولاحظ انه ليس من الضروري قراءة هذه الحروف، مثال:

```
client.write(data)
```

print

تستخدم هذه الدالة للكتابة على الخادم المتصل به العميل، وهي ترسل البيانات على شكل ارقام متسلسلة من الـ ASCII كود (مثلا الرقم 123 سيتم ارساله على شكل ثلاثة حروف ’3‘ ’2‘ ’1‘)، الصيغة العامة لها هي:

```
client.print(data)
```

حيث ان الباراميتر data يمكن ان يمثل أي بيانات من النوع (char, byte, int, long, or string).

println

هي تماما نفس الدالة السابقة print لكن الفرق الوحيد هو ان هذه الدالة بعد انتهائها ستنزل سطر.

available

تعيد عدد البايتات المتاحة للقراءة التي تم ارسالها من قبل الخادم للعميل، مثال:

```
if (client.available()) {

  // there are incoming bytes available from the server

}
```

read

تقرأ البايت التالي القادم من الخادم المتصل به العميل، البايت التالي نقصد به البايت الذي يلي اخر بايت قرأته هذه الدالة في اخر استدعاء لها، هذه الدالة تعيد البايت التالي او الحرف او انها ستعيد الرقم واحد إذا لم يكن هناك حرف تالي، مثال:

```
client.read()
```

flush

تستخدم لحذف الذاكرة المؤقتة buffer وهي تستدعى في الاخر بعد الانتهاء من الكتابة على العميل، مثال:

```
client.flush()
```

stop

تقطع اتصال العميل بالخادم، مثال:

```
client.stop()
```

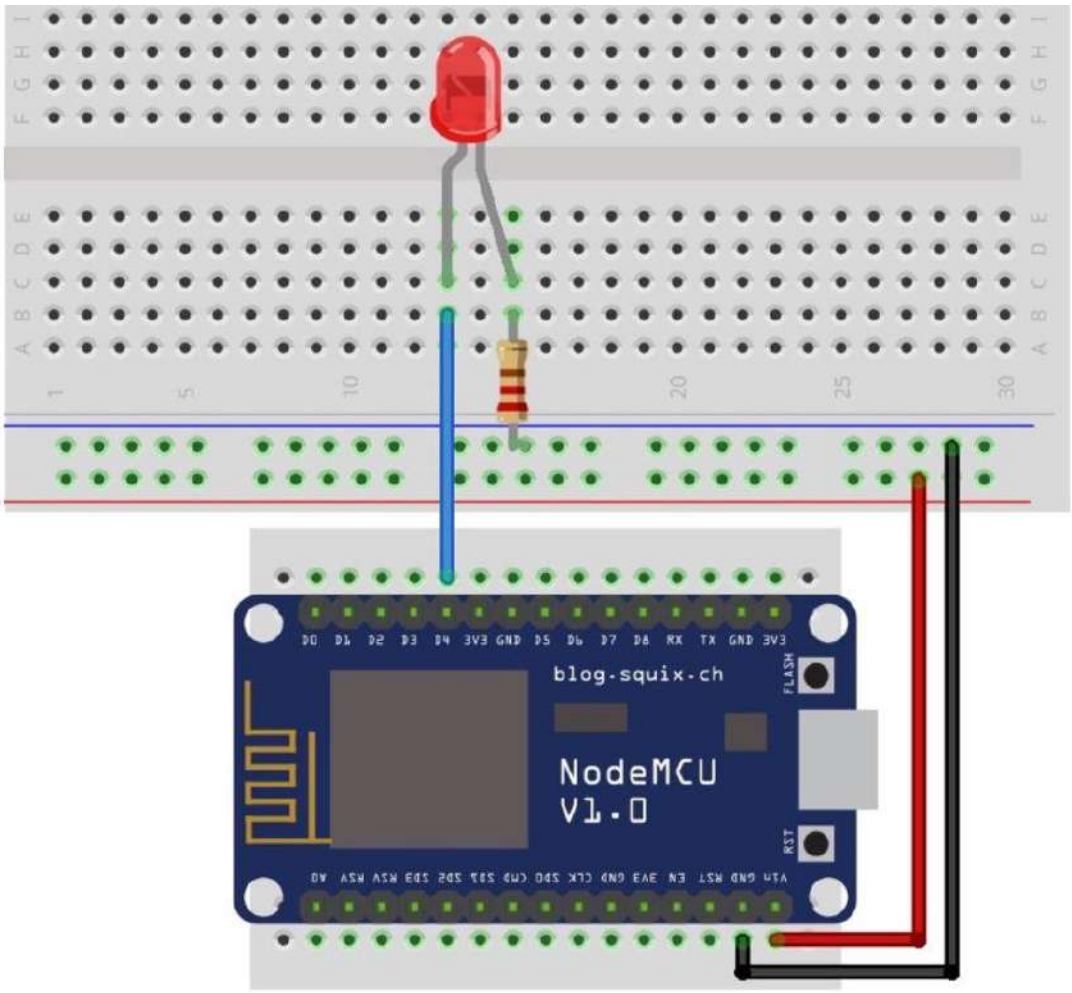
readStringUntil

يقوم باستلام البيانات المرسلة الى العميل وتخزينها في متغير نصي، مثال:

```
String request = client.readStringUntil('\r');
```

مثال متكامل:

لتنفيذ هذا المثال نربط المنفذ D4 لبوردة NodeMCU ESP8266 بالطرف الموجب لـ LED والطرف الاخر الساب للـ LED نربطه بمقاومة 220 اوم وطرف المقاومة الاخر نربطه بالأرضي G للبوردة كما موضح في هذه الصورة:



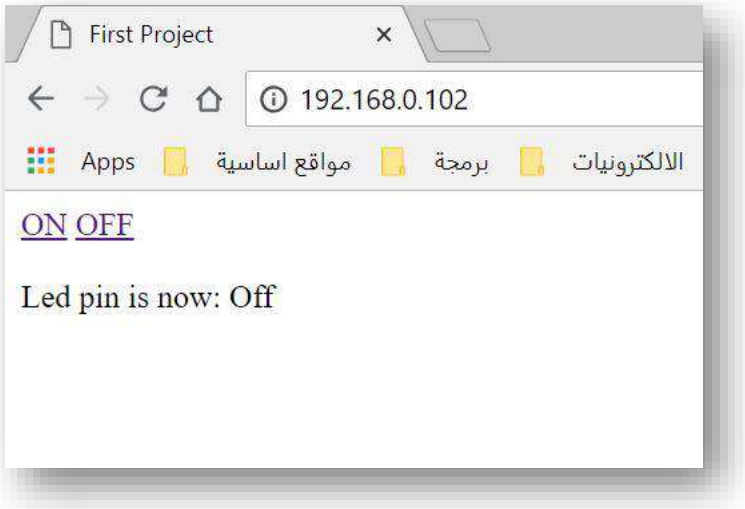
ثم نربط البوردة بالحاسب وعبر برنامج الاردوينو ننزل عليها الكود التالي:

```
#include <ESP8266WiFi.h>

WiFiServer server(80);

int value = LOW;

void setup() {

  Serial.begin(115200);

  pinMode(D4, OUTPUT);

  WiFi.begin("TP-LINK", "55555555");

  while (WiFi.status() != WL_CONNECTED) {
```

```
    delay(100);
  }
  server.begin();
  Serial.print(WiFi.localIP());
}
void loop() {
  WiFiClient client = server.available();
  if (!client) {
    return;
  }
  client.println("<!DOCTYPE HTML>");
  client.println("<html>");
  client.println("<head>");
  client.println("<title>First Project</title>");
  client.println("</head>");
  client.println("<body>");
  client.println("<a href='/LED=ON'>ON</a>");
  client.println("<a href='/LED=OFF'>OFF</a>");
  client.print("<br><br>");
  client.println("Led pin is now: ");
  if(value == HIGH) {
```

```
  client.print("On");
  } else {
  client.print("Off");
  }
  client.println("</body>");
  client.println("</html>");
  while(!client.available()){
    delay(1);
  }
  String request = client.readStringUntil('\r');
  client.flush();
  if (request.indexOf('/LED=ON') != -1) {
    digitalWrite(D4, HIGH);
    value = HIGH;
  }
  if (request.indexOf('/LED=OFF') != -1) {
    digitalWrite(D4, LOW);
    value = LOW;
  }
}
```

نفتح شاشة الـ Serial Monitor سنجد عنوان ip شبيه بهذا 192.168.0.102 (إذا لم يظهر نضغط على الزر RST الموجود على لوحة الـ Wi-Fi، ثم نأخذ هذا الـ ip وندخله في المتصفح وستظهر لنا نافذه بهذا الشكل:



عند الضغط على ON سيشتغل الـ LED وعند الضغط على OFF سينطفئ.

- إذا أردنا تحديث صفحة الـ HTML تلقائيا لتحديث عرض بيانات معينة كل عدد معين من الثواني نكتب هذا الكود في داخل الوسم <head> </head>

```
client.println("<meta http-equiv='refresh' content='3'>");
```

هذا الكود سيحدث الصفحة كل 3 ثانية.

مثال متكامل 2:

هذا المثال يستقبل بيانات نصية من برنامج اندرويد (برنامج الاندرويد موجود في كتاب الاندرويد الذي كتبته انا) ويعرضها على شاشة الـ Serial Monitor وكذلك يعيد نص الى برنامج الاندرويد.

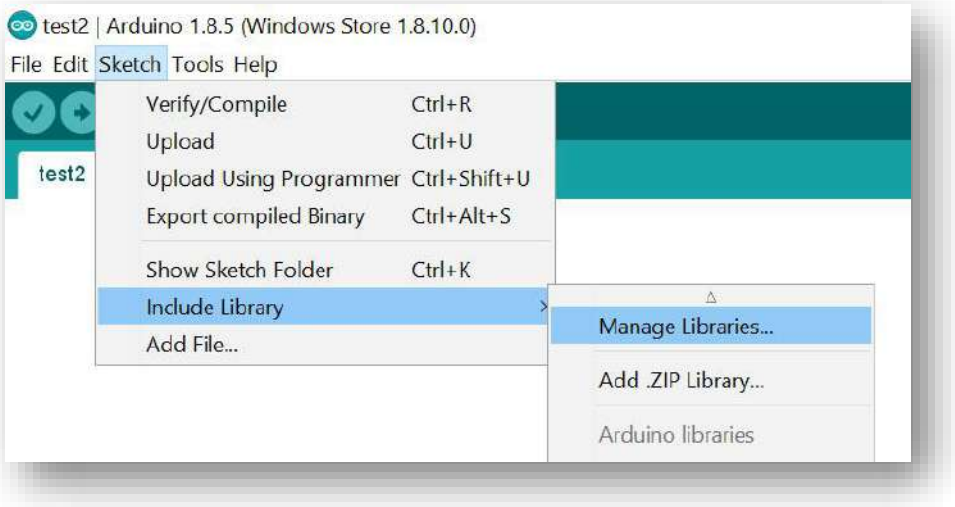
```
#include <ESP8266WiFi.h>
```

```
WiFiServer server(8090);
IPAddress ip(192, 168, 0, 117);
IPAddress gateway(192, 168, 0, 1);
IPAddress subnet(255, 255, 255, 0);

void setup() {
  Serial.begin(115200);
  WiFi.config(ip, gateway, subnet);
  WiFi.begin("TP-LINK", "55555555");
  while (WiFi.status() != WL_CONNECTED) {
    delay(100);
  }
  server.begin();
  Serial.println("Connected");
}

void loop() {
  WiFiClient client = server.available();
  if (!client) {
    return;
  }
```

```
  Serial.println(client.readStringUntil('$'));
  client.print("Hello from esp8266");
  delay(100);
  client.flush();
}
```

Simple DHT

تستخدم هذه المكتبة للتعامل مع متحسس الحرارة والرطوبة DHT (شرحنا عنه في هذا الكتاب في قسم المتحسسات) لنتعامل مع هذه المكتبة علينا في البداية اضافتها الى برنامج الاردوينو من خلال الذهاب الى Sketch ثم الى include library ثم نختار Manage libraries كما موضح في هذه الصورة:



ستظهر لنا نافذة نبحث فيها عن المكتبة SimpleDHT ثم نضغط على install كما في هذه الصورة:

بعد اكتمال تنصيبها سيكون بمقدورنا استخدامها في برامجنا التالية، ولتضمين هذه المكتبة في أي برنامج نستخدم هذه العبارة #include <SimpleDHT.h>

SimpleDHT11 && SimpleDHT22

من خلال أحد هذه الكلاسات نأخذ اوبجكت مرجع للمتحسس، ويستخدم أحد هذه الكلاسات بالاعتماد على نوع المتحسس المستخدم فإذا كنا قد استخدمنا المتحسس DHT11 نستخدم معه الكلاس SimpleDHT11 اما إذا استخدمنا المتحسس DHT22 فسنستخدم الكلاس SimpleDHT22، مثال:

```
#include <SimpleDHT.h>

SimpleDHT11 ss;

void setup() { }

void loop() { }
```

في هذا الكلاس أصبح عندنا الاوبجكت ss ومنه يمكن التعامل مع دوال هذا الكلاس.

read

هذه الدالة تابعه للكلاسين SimpleDHT11 && SimpleDHT22 من خلالها يتم قراءة درجة الحرارة والرطوبة، تأخذ هذه الدالة 4 بارامترات الأول هو عبارة عن رقم الـ pin المربوط عليه منفذ المتحسس DATA، الباراميتر الثاني هو يمثل المتغير الذي ستخزن فيه درجة الحرارة لذا يمرر هذا الباراميتر كمرجع (أي يسبق بالعلامة &) الباراميتر الثالث هو عبارة عن متغير ستخزن فيه درجة الرطوبة لذا يمرر كمرجع أيضا (أي يسبق بالعلامة &)، الباراميتر الرابع يوضع عادتا NULL، مثال:

```
#include <SimpleDHT.h>

SimpleDHT11 s;

void setup() {

  Serial.begin(115200);

}

void loop() {

  byte t = 0;

  byte h = 0;

  s.read(2, &t, &h, NULL);

  Serial.print(t); Serial.print(" *C, ");

  Serial.print(h); Serial.println(" H");

  delay(1500);

}
```

Wire

هذه المكتبة مخصصة للتعامل مع البروتوكول I2C، هذا البروتوكول مسؤول عن نقل البيانات من متحسسات متعددة الى الاردوينو فقط من خلال سلكين وسيكون الاردوينو بمثابة السيد master والمتحسسات بمثابة الخادم slave، لتضمين هذه المكتبة نكتب في بداية البرنامج #include <Wire.h>

begin

تستخدم هذه الدالة لتهيئة المكتبة وهي تستدعى مرة واحدة (أي في داخل الدالة setup)، الصيغة العامة لها هي:

Wire.begin(address)

حيث ان البارامتر الممرر لها هو اختياري حيث إذا وضعناه فهذا يعني انه سننضم الى خط البيانات كخادم slave اما إذا لم نضعه فهذا يعني اننا سننضم كسيد master، هذا البارامتر يمثل عنوان الـخادم وهو مكون من 7 bit، كل جهاز او متحسس نربطه له عنوان خاص به يمكن ان نجده في الـ datasheet الخاصة بهذا المتحسس.

requestFrom

تستخدم هذه الدالة من قبل السيد master لطلب البيانات من الخادم، وهذه البيانات يتم استرجاعها (اخذها) من الخادم باستخدام الدوال available و read، الصيغة العامة لها هي:

Wire.requestFrom(address, quantity, stop)

حيث ان البارامتر الأول يمثل عنوان الخادم الذي سنطلب منه البيانات (يمكن ان يكون اكثر من خادم مربوط على نفس الخط) وهو مكون من 7 bit، كل جهاز او متحسس نربطه له عنوان خاص به يمكن ان نجده في الـ datasheet الخاصة بهذا المتحسس، البارامتر الثاني يمثل عدد البايتات التي سنطلبها، البارامتر الثالث هو اختياري اما يكون true وهذا يعني ان نوقف عملية الاتصال بهذا الخادم بعد اخذ الطلب لتحرير الخط الناقل لاستقبال بيانات من الخوادم الأخرى (وهذه القيمة هي الافتراضية) اما القيمة false فتعني انه بعد هذا الطلب لدينا طلب اخر من نفس الخادم لتستمر العملية طلب بعد طلب، هذه الدالة تعيد عدد البايتات القادمة من الخادم.

available

تعيد هذه الدالة عدد البايتات المتاحة للقراءة، هذه الدالة يجب ان تستدعى في جهاز السيد master بعد الدالة requestFrom او في جهاز الخادم slave بداخل الدالة onReceive، الصيغة العامة لهذه الدالة هي:

```
Wire.available()
```

read

تستخدم هذه الدالة لقراءة البيانات القادمة من الخادم وذلك بعد استدعاء الدالة requestFrom، الصيغة العامة لهذه الدالة هي:

```
Wire.read()
```

هذه الدالة تعيد البايت التالي المستلم من الخادم.

مثال:

```
#include <Wire.h>
void setup(){
  Wire.begin();        // join i2c bus (address optional for master)
  Serial.begin(9600);  // start serial for output
}
void loop(){
  Wire.requestFrom(2, 6);    // request 6 bytes from slave device #2
                             // device address is specified in datasheet
  while(Wire.available()){    // slave may send less than requested
    char c = Wire.read();    // receive a byte as character
```

```
    Serial.print(c);        // print the character
  }
  delay(500);
}
```

beginTransmission

تستخدم هذه الدالة للانضمام الى الخط الناقل I2C والتهيئة لبدا نقل البيانات الى الجهاز الخادم الذي نضع عنوانه بين قوسي هذه الدالة (العنوان هو عبارة عن 7 bit، كل جهاز او متحسس نربطه له عنوان خاص به يمكن ان نجده في الـ datasheet الخاصة بهذا المتحسس)، الصيغة العامة لهذه الدالة هي:

```
Wire.beginTransmission(address)
```

write

بعد ان هيئنا عملية نقل البيانات بالدالة beginTransmission علينا الان ان نجهز البيانات المطلوب ارسالها بهذه الدالة write، لهذه الدالة أكثر من صيغة عامة يمكن استخدام أي واحدة منها وهي:

```
Wire.write(value)

Wire.write(string)

Wire.write(data, length)
```

- value قيمة يتم ارسالها كبايت مفرد.
- string بيانات نصية يتم ارسالها على شكل سلسلة من البايتات.
- data مصفوفة من البيانات يتم ارسالها على شكل بايتات.
- length عدد البايتات التي ستنقل.

endTransmission

بعد تهيئة عملية النقل باستخدام الدالة beginTransmission ومن ثم تجهيز البيانات باستخدام الدالة write حان الوقت الان لإكمال عملية الارسال وانهائها باستخدام هذه الدالة endTransmission، الصيغة العامة لهذه الدالة هي:

Wire.endTransmission(stop)

حيث ان البارامتر stop هو اختياري (يمكن عدم وضعه) وهو اما يكون true وهذا يعني ان نوقف عملية الاتصال بهذا الخادم بعد ان تم ارسال البيانات لتحرير الخط الناقل لاستقبال بيانات من الخوادم الأخرى (وهذه القيمة هي الافتراضية) اما القيمة false فتعني انه بعد هذه العملية سنبقي عملية الاتصال قائمة مع نفس الخادم للاستمرار بعمليات أخرى، هذه الدالة تعيد رقم بايت يمثل حالة عملية النقل:

- 0 العملية تمت بنجاح.
- 1 البيانات طويلة جدا ولا تتناسب مع الذاكرة المؤقتة buffer (المسؤولة عن التخزين قبل الارسال).
- 2 استلم NACK على عنوان النقل.
- 3 استلم NACK على البيانات المرسلة.
- 4 أخطاء أخرى.

مثال:

```
#include <Wire.h>

byte val = 0;

void setup(){

  Wire.begin(); // join i2c bus

}

void loop(){

  Wire.beginTransmission(44); // transmit to device #44 (0x2c)

                    // device address is specified in datasheet

  Wire.write(val);          // sends value byte
```

```
  Wire.endTransmission();     // stop transmitting
  val++;        // increment value
  if(val == 64){ // if reached 64th position (max)
    val = 0;    // start over from lowest value
  }
  delay(500);
}
```

setClock

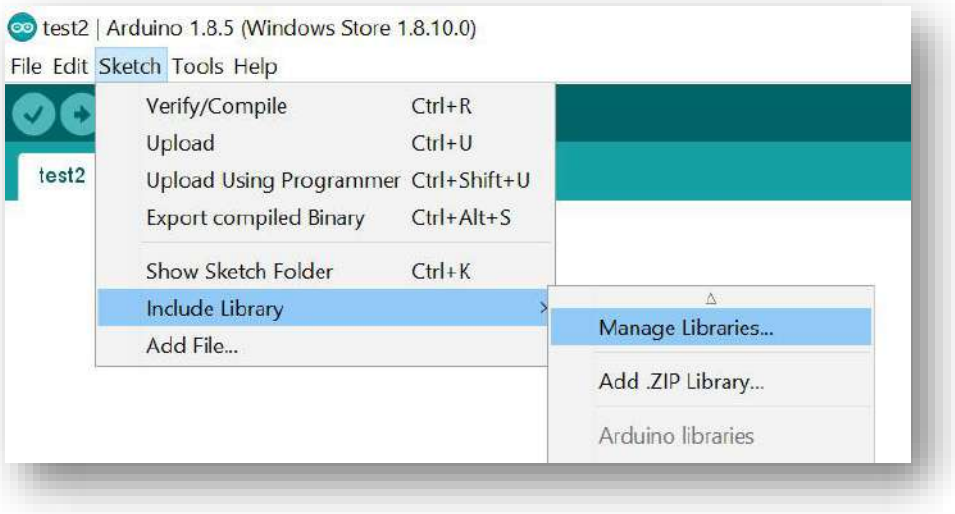
هذه الدالة تعدل تردد الساعة للاتصال I2C، الجهاز الخادم في الغالب ليس له تردد ساعة محدد لكن غالبا ما يستخدم التردد 100 كيلوهرتز للساعة، الصيغة العامة لهذه الدالة هي:

```
Wire.setClock(clockFrequency)
```

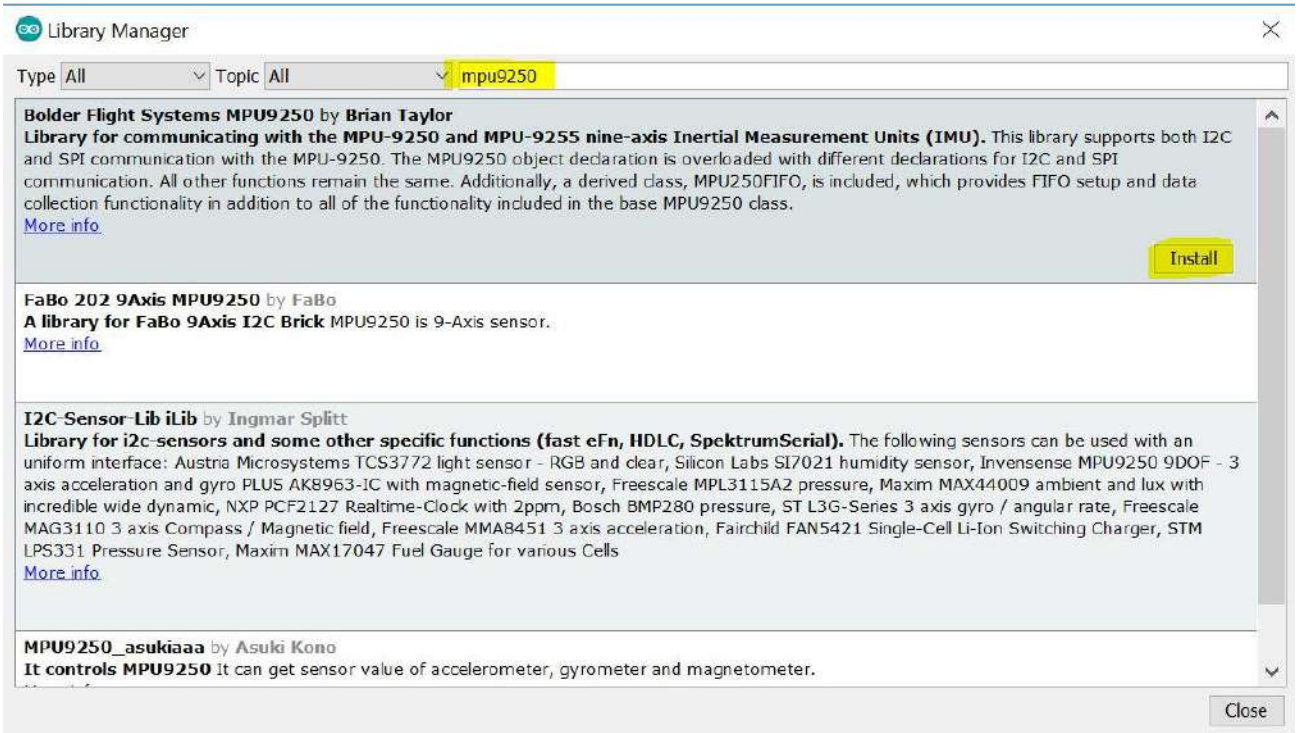
البارامیتر الممرر لها هو عبارة عن رقم يمثل تردد الساعة بالهرتز، حيث ان الرقم 100000 يعتبر هو القيمة الافتراضية ويمكن ان نقلل او نزيد هذا الرقم.

MPU9250

هذه المكتبة خاصة للتعامل مع المتحسس MPU-9250 (شرحنا عن هذا المتحسس في قسم المتحسسات من هذا الكتاب)، لإضافة هذه المكتبة الى برنامج الاردوينو نذهب الى القائمة Sketch ثم الى include library ثم نختار Manage libraries كما موضح في هذه الصورة:



ستظهر لنا نافذة نبحث فيها عن المكتبة MPU9250 ثم نضغط على install كما موضح في هذه الصورة:



بعد اكتمال تثبيتها يمكن الذهاب الى القائمة file ونختار examples ونبحث عن الأمثلة الخاصة بـ MPU9250 لنستعرضها.

يوجد شرح كامل لهذه المكتبة على هذا الرابط:

https://github.com/bolderflight/MPU9250

هذه المكتبة تدعم البروتوكول I2C والبروتوكول SPI، بدايتا علينا استدعاء المكتبة بهذا الشكل:

```
#include <MPU9250.h>
```

الكلاس MPU9250

في بداية البرنامج وحتى قبل الدالة setup علينا انشاء اوبجكت من هذا الكلاس لان كل دوال هذه المكتبة ستعتمد عليه، ويتم انشاء اوبجكت من هذا الكلاس للتعامل مع البروتوكول I2S بهذا الشكل:

```
MPU9250 IMU(Wire,0x68);
```

حيث ان IMU هو اسم الاوبجكت و0x68 يمثل العنوان للمتحسس MPU9250 (العنوان address الافتراضي لهذا المتحسس هو 0x68 ويمكن تغييره الى 0x69 إذا ربطنا المنفذ AD0/SD0 بـ 5 فولت).

begin

هذه الدالة اجبارية وهي تكتب عادتا في داخل الدالة setup وهي تستخدم لتهيئة الاتصال مع المتحسس MPU9250، هذه الدالة تعيد قيمة موجبة في حال نجح الاتصال مع المتحسس او تعيد قيمة سالبة إذا لم ينجح الاتصال (أي علينا التأكد من الربط مع المتحسس)، مثال:

```
int status;

status = IMU.begin();
```

setAccelRange

هذه الدالة تستخدم لتحديد مدى القياس للـ accelerometer وتمرر لها أحد هذه القيم كبارامتر:

Accelerometer Name	Accelerometer Full Scale Range
ACCEL_RANGE_2G	+/- 2 (g)
ACCEL_RANGE_4G	+/- 4 (g)
ACCEL_RANGE_8G	+/- 8 (g)
ACCEL_RANGE_16G	+/- 16 (g)

بشكل افتراضي في حال عدم كتابة هذه الدالة سيكون مدى القياس هو (g) 16 -/+ ، هذه الدالة تعيد قيمة من النوع int موجبة إذا تمت العملية بنجاح او تعيد قيمة سالبة في حالة الفشل، مثال لجعل المدى هو (g) 8 -/+

```
int status;

status = IMU.setAccelRange(MPU9250::ACCEL_RANGE_8G);
```

setGyroRange

هذه الدالة تستخدم لتحديد مدى القياس للـ gyroscope وتمرر لها أحد هذه القيم كبارامتر:

Gyroscope Name	Gyroscope Full Scale Range
GYRO_RANGE_250DPS	+/- 250 (deg/s)
GYRO_RANGE_500DPS	+/- 500 (deg/s)
GYRO_RANGE_1000DPS	+/- 1000 (deg/s)
GYRO_RANGE_2000DPS	+/- 2000 (deg/s)

بشكل افتراضي في حال عدم كتابة هذه الدالة سيكون مدى القياس هو (deg/s) 2000 -/+ ، هذه الدالة تعيد قيمة من النوع int موجبة إذا تمت العملية بنجاح او تعيد قيمة سالبة في حالة الفشل، مثال لجعل المدى هو +/- 250 (deg/s)

```
int status;

status = IMU.setGyroRange(MPU9250::GYRO_RANGE_250DPS);
```

setDlpfBandwidth

هذه الدالة هي اختيارية وهي تستخدم لتحديد الـ Digital Low Pass Filter (DLPF) bandwidth ، بشكل افتراضي في حال عدم استخدام هذه الدالة تكون القيمة 184 Hz ، ويمكن تغيير القيمة الافتراضية واستخدام أحد هذه القيم:

Bandwidth Name	DLPF Bandwidth	Gyroscope Delay	Accelerometer Delay	Temperature Bandwidth	Temperature Delay
DLPF_BANDWIDTH_184HZ	184 Hz	2.9 ms	5.8 ms	188 Hz	1.9 ms
DLPF_BANDWIDTH_92HZ	92 Hz	3.9 ms	7.8 ms	98 Hz	2.8 ms

DLPF_BANDWIDTH_41HZ	41 Hz	5.9 ms	11.8 ms	42 Hz	4.8 ms
DLPF_BANDWIDTH_20HZ	20 Hz	9.9 ms	19.8 ms	20 Hz	8.3 ms
DLPF_BANDWIDTH_10HZ	10 Hz	17.85 ms	35.7 ms	10 Hz	13.4 ms
DLPF_BANDWIDTH_5HZ	5 Hz	33.48 ms	66.96 ms	5 Hz	18.6 ms

هذه الدالة تعيد قيمة من النوع int موجبة إذا تمت العملية بنجاح او تعيد قيمة سالبة في حالة الفشل، مثال لجعل القيمة DLPF bandwidth of 20 Hz:

```
int status;

status = IMU.setDlpfBandwidth(MPU9250::DLPF_BANDWIDTH_20HZ);
```

setSrd

هذه الدالة تحدد نسبة البيانات المخرجة، حيث ان نسبة البيانات المخرجة تحدد بواسطة المعادلة التالية:

```
Data Output Rate = 1000 / (1 + SRD)
```

بشكل افتراضي في حال عدم استخدام هذه الدالة فإن قيمة الـ SDR تكون 0، وهذه القيمة تسمح للمتحسسات gyroscopes, accelerometers, temperature بان تكون بين 3.9 Hz الى 1000 Hz، مثال:

```
status = IMU.setSrd(9);
```

هذه الدالة تعيد قيمة موجبة في حال نجاحها وقيمة سالبة في حالة فشلها.

ملاحظات عامة

- لكتابة تعليقات في داخل البرنامج (وتستخدم التعليقات فقط للتوضيح للمبرمج وهي لن تنفذ في البرنامج ويمكن كتابتها في اي مكان من البرنامج) نستخدم العلامة // وهذه تستخدم لكتابة سطر واحد اما اذا اردنا ان نكتب بأكثر من سطر سنستخدم العلامة /* هنا التعليق */

- العمليات الحسابية

المعنى	العملية
عملية مساواة	=
عملية جمع	+
عملية طرح	-
عملية ضرب	*
عملية قسمة لكن تعطي الناتج كعدد صحيح فقط، مثال: 2 == 2 / 5	/
باقي القسمة، مثال: 1 == 2 / 5	%

- المنافذ في بوردة الاردوينو المرقمة A0 الى A5 هي في الحقيقة تكملة للمنافذ الاخرى حيث ان A0 يمثل المنفذ 14 وA1 تمثل المنفذ رقم 15 وهكذا، ويمكن استخدامها كمنافذ ادخال او اخراج Digital وليس فقط كمنافذ ادخال Analog كما هو متعارف عليه، ولكن انتبه انه لا يمكن استخدام المنافذ الاخرى (والتي ليست من ضمن A0 الى A5) كمنافذ ادخال Analog .

- لمعرفة اي شيء عن لغة اردوينو زر هذا الرابط

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

ولمزيد من المعلومات حول مكتبة لوحة المفاتيح زر هذا الرابط

http://playground.arduino.cc/Code/Keypad