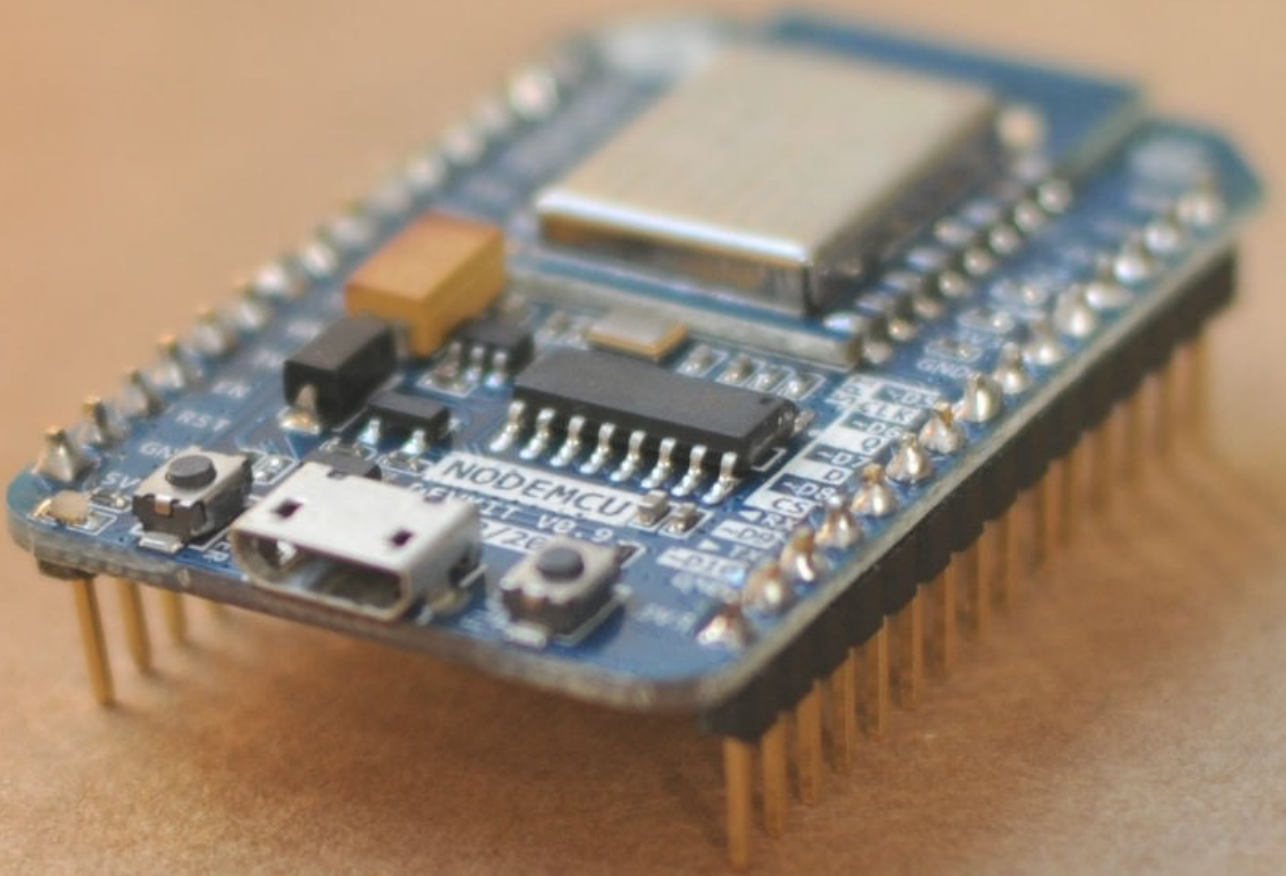


شريحة ESP8266

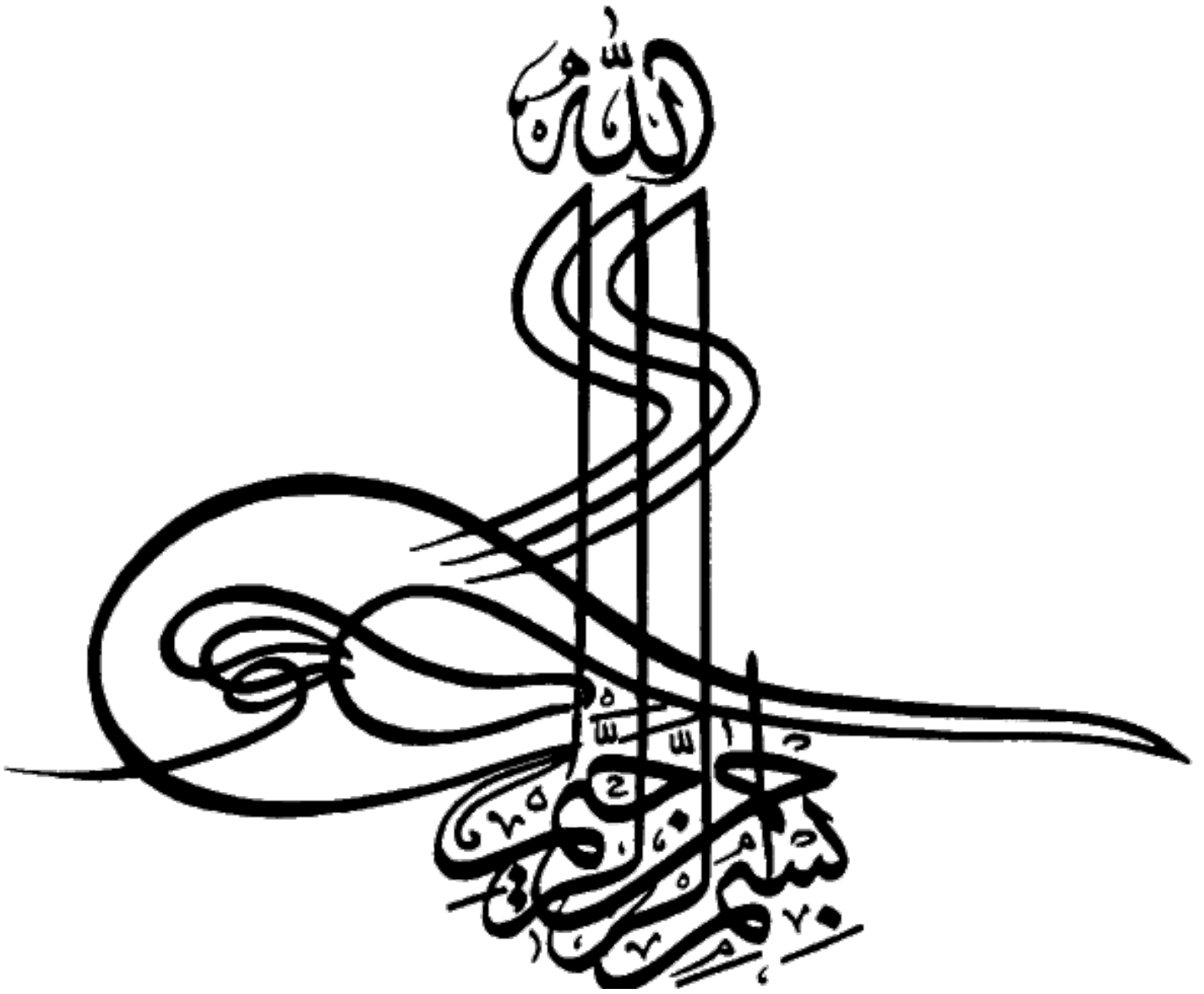
التحكم اللاسلكي من خلال
الواجهات الإلكترونية



جهاد طلعت بسيوني



ESP8266





رخصة الكتاب

هذا العمل مرخص برخصة نَسب المُصنَّف - غير تجاري - الترخيص بالمثل 4.0 دولي المشاع الإبداعي. لمشاهدة نسخة من هذه الرخصة، قم بزيارة

<http://creativecommons.org/licenses/by-nc-.sa/4.0/deed.ar>



إهداء

إلى أمتي





العلم

العلم ثروة أمة ويسارُ
العلم قد دك الجبال فهدها
بالعلم أطلعت البلاد كواكباً
بالعلم أدنى الناس شقة أرضهم
بالعلم قد طالت فأدركت المنى
العلم ينمو في المدارس دوحه
ما كان يفلح في جهاد حياته
سَيَموت رب العلم من مرض به
شتان بين الدار تبسط ظلمة
لا يرفع الوطن العزيز سوى أمرئٍ
والجهل حرمان لها وبوارُ
وأضاءَ جناح الليل فهو نهار
بالعلم صارت تنطق الأحجار
بالعلم غاصوا في البحار وطاروا
أيدٍ عن الغرض الرفيع قصار
حيناً وتقطف بعد ذاك ثمار
شعب على كسل له استمرار
وتعيش دهوراً بعده الآثار
والدار فيها تسطع الأنوار
حر على الوطن العزيز يغار



الفهرس

1	رخصة الكتاب
2	اهداء
6	الباب الأول: مقدمة عن شريحة ESP8266
7	شريحة ESP8266
9	لوحات ESP8266
11	الباب الثاني: مقدمة عن لوحة NodeMCU
12	لوحة NodeMCU
13	نظرة عامة على لوحة NodeMCU
25	الباب الثالث: الأدوات والقطع الإلكترونية
31	الباب الرابع: تثبيت نظام NodeMCU
32	الفصل الأول: لنظام ماك ولينكس
59	الفصل الثاني: لنظام ويندوز
76	الباب الخامس: البيئة التطويرية
77	الفصل الأول: برنامج ESPlorer
84	الفصل الثاني: برنامج Arduino



92	الباب السادس: الطريق إلى التحكم
94	الفصل الأول: صفحة خادم الشبكة
102	الفصل الثاني: المشاريع (برنامج ESPlorer)
103	المشروع الأول: الفلاش
116	المشروع الثاني: خاصية PWM
131	المشروع الثالث: واجهة تشغيل وإطفاء
147	المشروع الرابع: عرض بيانات حساس الضوء
163	المشروع الخامس: التحكم بالدايود متعدد الألوان
189	المشروع السادس: تنبيه البريد الإلكتروني
213	الفصل الثالث: المشاريع (برنامج Arduino)
214	المشروع الأول: واجهة تشغيل وإطفاء
231	المشروع الثاني: عرض بيانات حساس الحرارة
247	المراجع
249	أعمال أخرى للمؤلف
251	التواصل مع المؤلف



ESP8266



البيانات الأولى

مقدمة عن شريحة *ESP8266*



شريحة ESP8266:

هي عبارة عن شريحة واي فاي (wifi) و متحكم دقيق (ميكروكونترولر).
و ذات سعر منخفض جدا مقارنة بأدائها وقدراتها.
وتتوفر بعدة أنواع تختلف في مميزاتا وعدد أطرافها كما هو موضح في
الصورة التالية:



ESP-01



ESP-02



ESP-03



ESP-04



ESP-05



ESP-06



ESP-07



ESP-08



ESP-09



ESP-10



ESP-11



ESP-12



ESP-12E



ESP8266



والصورة التالية توضح الفرق بين أنواع شرائح ESP8266:

Name	Active pins	Pitch	Form factor	LEDs	Antenna	Shielded?	dimensions (mm)
ESP-01	6	0.1"	2x4 DIL	Yes	PCB trace	No	14.3 x 24.8
ESP-02	6	0.1"	2x4 castellated	No	U-FL connector	No	14.2 x 14.2
ESP-03	10	2 mm	2x7 castellated	No	Ceramic	No	17.3 x 12.1
ESP-04	10	2 mm	2x4 castellated	No	None	No	14.7 x 12.1
ESP-05	3	0.1"	1x5 SIL	No	U-FL connector	No	14.2 x 14.2
ESP-06	11	misc	4x3 dice	No	None	Yes	14.2 x 14.7
ESP-07	14	2 mm	2x8 pinhole	Yes	Ceramic + U-FL connector	Yes	20.0 x 16.0
ESP-08	10	2 mm	2x7 castellated	No	None	Yes	17.0 x 16.0
ESP-09	10	misc	4x3 dice	No	None	No	10.0 x 10.0
ESP-10	3	2 mm?	1x5 castellated	No	None	No	14.2 x 10.0
ESP-11	6	0.05"	1x8 pinhole	No	Ceramic	No	17.3 x 12.1
ESP-12	14	2 mm	2x8 castellated	Yes	PCB trace	Yes	24.0 x 16.0
ESP-12-E	20	2 mm	2x8 castellated	Yes	PCB trace	Yes	24.0 x 16.0

نلاحظ أن الشرائح ESP-06-07-08-12-12E لا يمكن استخدامها مجردة كما هي موجودة في الصورة السابقة. بل لابد من توصيلها بدوائر إلكترونية أخرى وهذا هو ما تعنيه خانة (Shielded) في الجدول أعلاه.



لوحات ESP8266:

تعتبر اللوحات الخيار الافضل لإستخدام شرائح ESP8266.
وهناك العديد من هذه اللوحات وأشهرها:



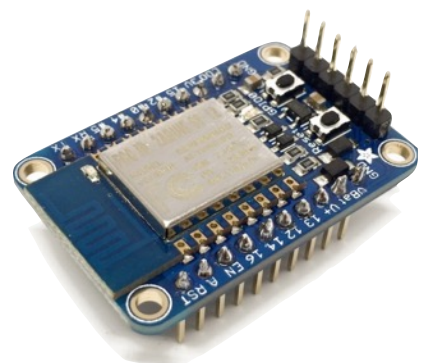
لوحة NodeMCU
وتحتوي على ESP-12



لوحة ESP-07



لوحة ESPduino
وتحتوي على ESP-13



لوحة Adafruit Huzzah
وتحتوي على ESP-12



ESP8266



والصورة التالية توضح الفرق بين بعض لوحات ESP8266:

Name	Active pins	Pitch	Form factor	LEDs	dimensions (mm)	Notes
Adafruit Huzzah ESP8266 breakout ^[15]	14	0.1"	2x10 DIL	Yes	25 x 38	Uses the ESP-12 module
ESPert ESPresso Lite V2.0 ^[22]	24	0.1"	2x10 DIL	Yes	28 x 61	Improved design and feature to ESPresso Lite.
ESPert ESPresso Lite ^[21]	16	0.1"	2x8 DIL	Yes	26.5 x 57.6	Uses the WROOM-02 module. Produced in limited quantity as beta version.
In-Circuit ESP-ADC ^[23]	18	0.1"	2x9 DIL	No	22.9 x 14.9	Uses the ESP8266EX
KNEWRON Technologies smartWIF ^[17]	12	0.1"	2x20 DIL	Yes 1 RGB	25.4 x 50.8	CP2102 USB bridge, includes battery charger, micro-USB socket for power and battery charging, 1 RGB LED and USER / Reflash button
NodeMCU DEVKIT	14	0.1"	2x15 DIL	Yes	?	Uses the ESP-12 module, includes USB serial interface
Olimex MOD-WIFI-ESP8266-DEV ^[14]	20	0.1"	2x11 DIL + castellated	Yes	?	All available GPIO pins are connected, also has pads for soldering UEXT connector (with RX/TX and SDA/SCL signals)
Olimex MOD-WIFI-ESP8266 ^[13]	2	0.1"	UEXT module	Yes	?	Only RX/TX are connected to UEXT connector
SparkFun ESP8266 Thing ^[16] WRL-13231	12	0.1"	2x10 DIL	Yes	58 x 26	FTDI serial header, Micro-USB socket for power, includes Li-ion battery charger

ملاحظات:

- في هذا الكتاب سنستخدم لوحة NodeMCU وسنشرح ونطبق عليها.
- لقد تم تجربة شريحة ESP8266-01 وهي أيضا تعمل وفق هذا الكتاب
- بالنسبة للوحات والشرائح الأخرى فإن الأرجح أن تعمل أيضا وفق هذا الكتاب. حيث أن طريقة تثبيت النظام والبرمجة واحدة لا تختلف.



NodeMCU



البيلاج التلقائي

مقدمة عن لوحة *NodeMCU*



لوحة NodeMCU:

هي عبارة عن لوحة مفتوحة المصدر قابلة للبرمجة وتوفر خاصية انترنت الاشياء (internet of things) والتي تسمح بربط الاشياء مع بعضها والتفاهم فيما بينها من خلال شبكة الانترنت. ويقصد بعبارة الاشياء انها

جميع الاجهزة الذكية مثل:

التلفاز، الجوال، الساعات،

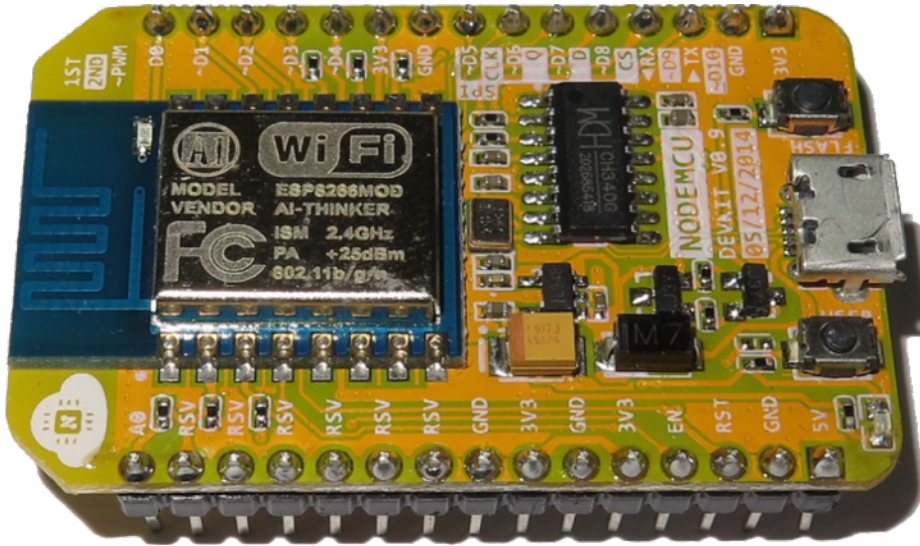
النظارات، اجهزة الانذار

والمراقبة، وغيرها

من الاجهزة التي يمكن

ربطها مع بعضها البعض

عبر شبكة الانترنت.



وتحتوي هذه اللوحة على شريحة ESP2866-12.

وتعتبر هذه اللوحة جديدة نوعا ما وخاصة في عالمنا العربي. حيث كان أول

ظهور لها في عام 2014.

وهذا رابط موقع لوحة NodeMCU

http://nodemcu.com/index_cn.html



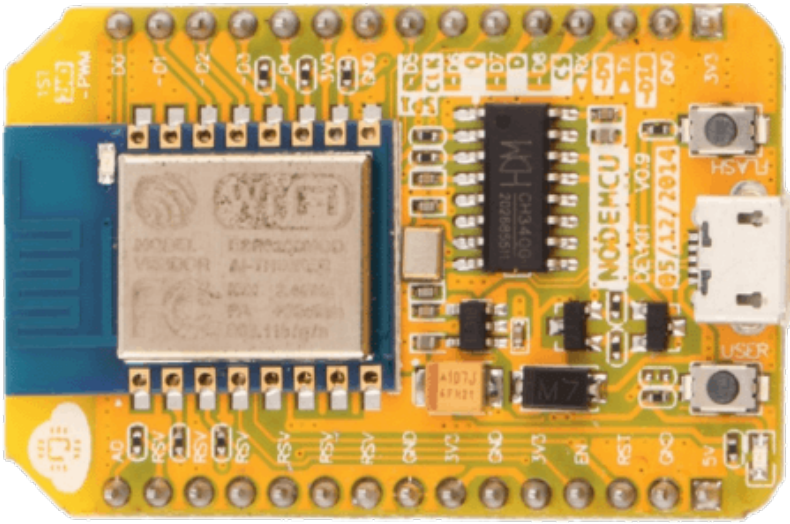
نظرة عامة على لوحة NodeMCU:

حتى الان يوجد اصدارين للوحة NodeMCU :

الإصدار الاول (V 0.9):

(أ) الأطراف:

تحتوي اللوحة على عشرة أطراف (D10 – D1) نستطيع استخدامها كدخل أو كخرج (input أو output) وتدعم خاصية PWM.



وأیضا تحتوي على طرف

(D0) لا يدعم خاصية

.PWM

وتحتوي على طرف (A0)

نستطيع استخدامها كدخل

تماثلي (analog input).

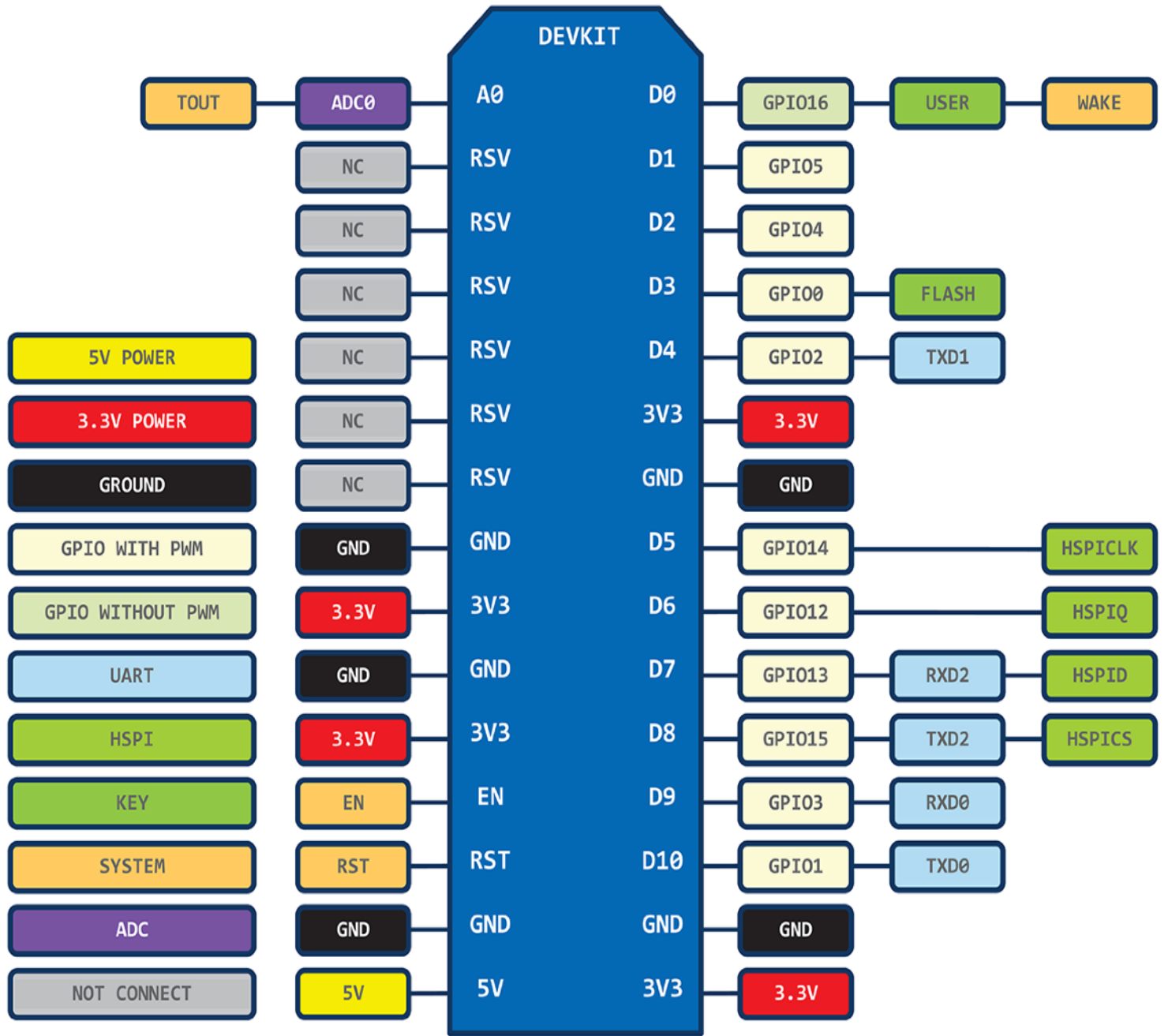
بالإضافة الى طرف 5 فولت (5v) و أربعة اطراف 3.3 فولت (3.3v) و

خمسة أطراف للأرضي (GND).

ستجدون تفاصيل الاطراف في مخطط اللوحة في الصورة التالية:



NodeMCU



نلاحظ أن هناك 6 أطراف باللون الرمادي (NC) غير متصلة أي انها غير مستخدمة.



(ب) المفاتيح:

تحتوي اللوحة على مفتاحين من النوع الضغط (push button).

* المفتاح الاول (flash button):

يستخدم عند تثبيت نظام NodeMCU

* المفتاح الثاني (user button):

يستخدم عند رفع برنامج (شيفرة برمجية)

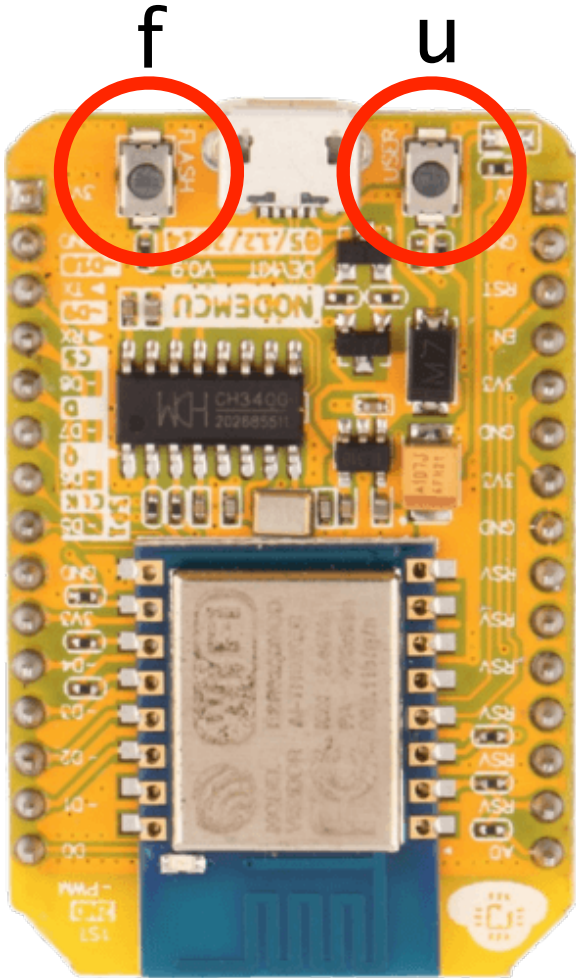
جديد على NodeMCU

ملاحظة:

إذا أردت استخدام أحد هذان المفتاحين،

يجب الضغط عليه قبل أن تمد اللوحة

بالطاقة.



(ج) ابعاد اللوحة:

تأتي اللوحة في الاصدار الاول بطول (47 ملم) وعرض (31 ملم). وهذا

يعتبر أحد عيوب هذا الاصدار. حيث لا يمكن استخدامها بسهولة مع لوحة

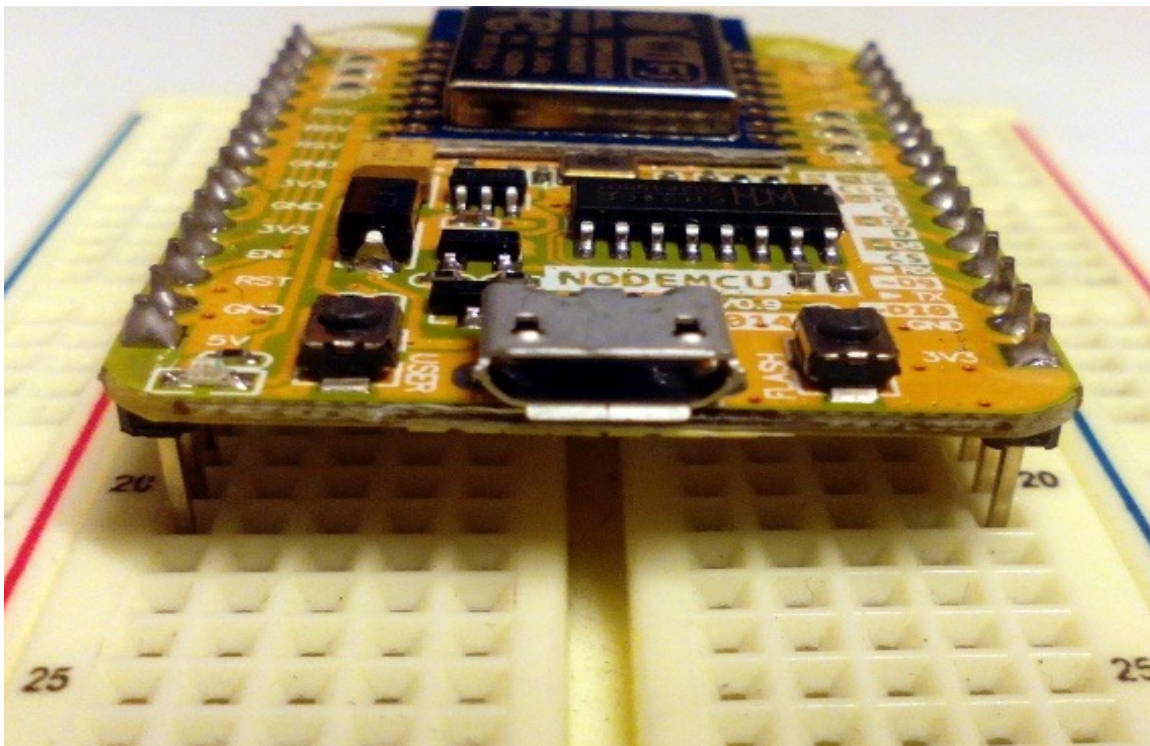
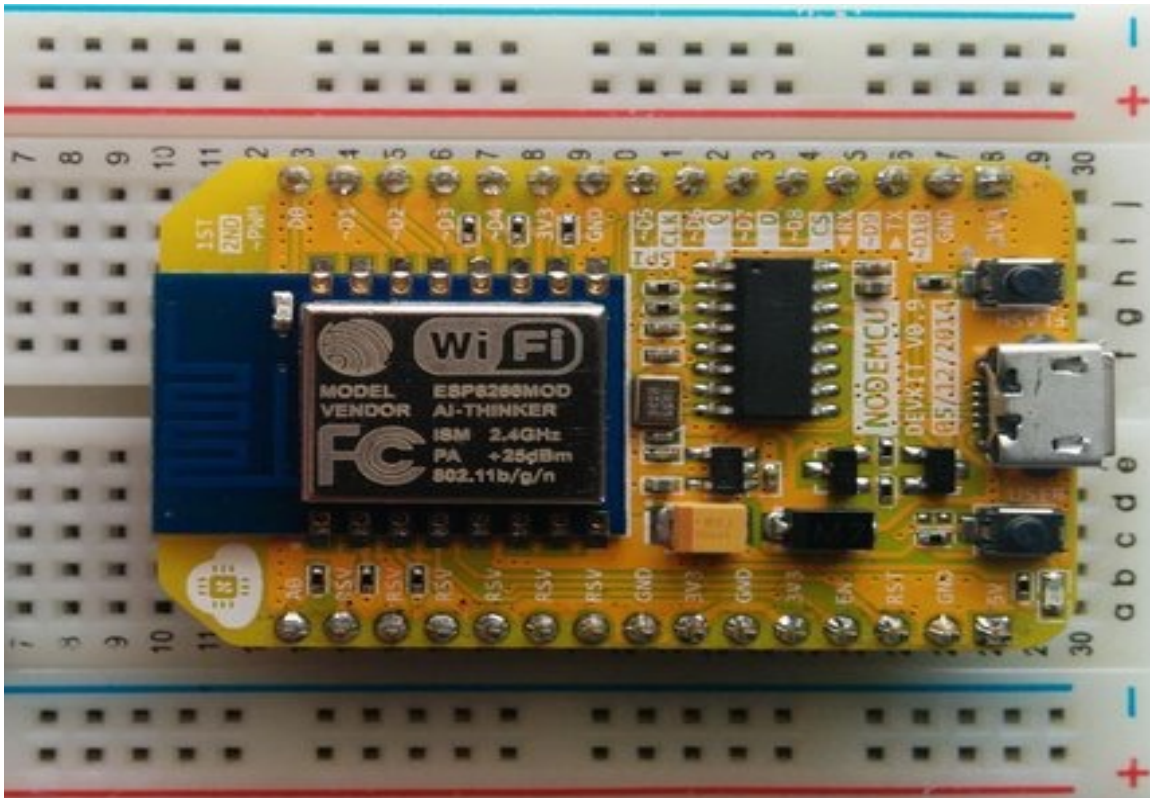
تثبيت القطع الالكترونية (Breadboard) والسبب أن لوحة NodeMCU

تقوم بتغطية جميع المداخل في لوحة تثبيت القطع كما تلاحظون في الصور

التالية:



NodeMCU





د) الطاقة:

شرائح ESP8266 تعمل على جهد 3.3 فولت. وكذلك قيمة الخرج (output) في شرائح ESP8266 هي 3.3 فولت. ولكن لوحة NodeMCU الاصدار الاول يمكن تشغيلها على جهد 3.3 أو 5 فولت والسبب أنها مزودة بمنظم للجهد (regulator) لتخفيض الجهد من 5 الى 3.3 فولت.

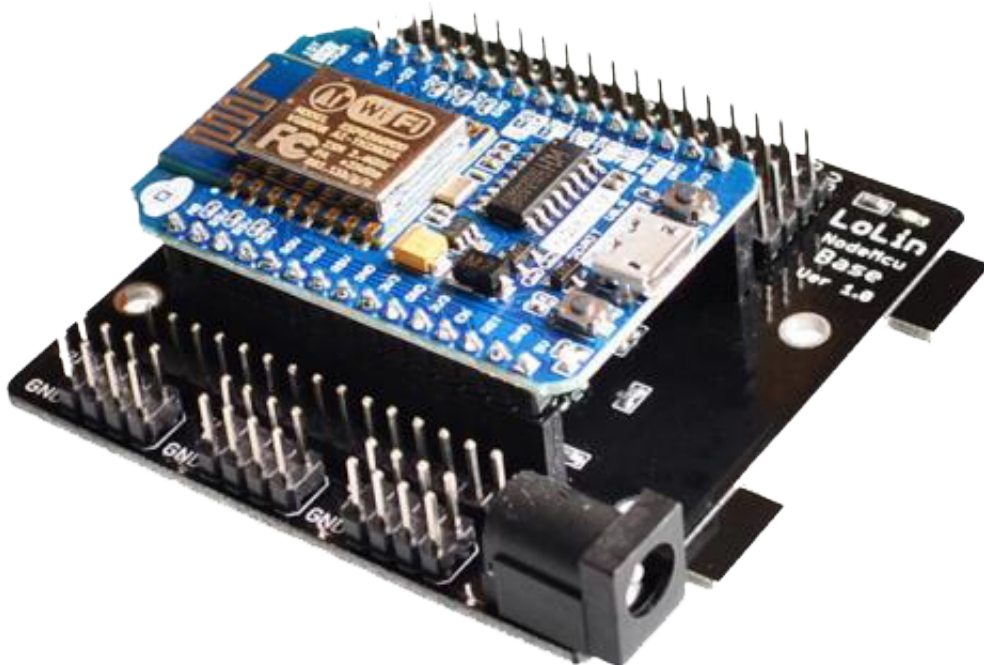
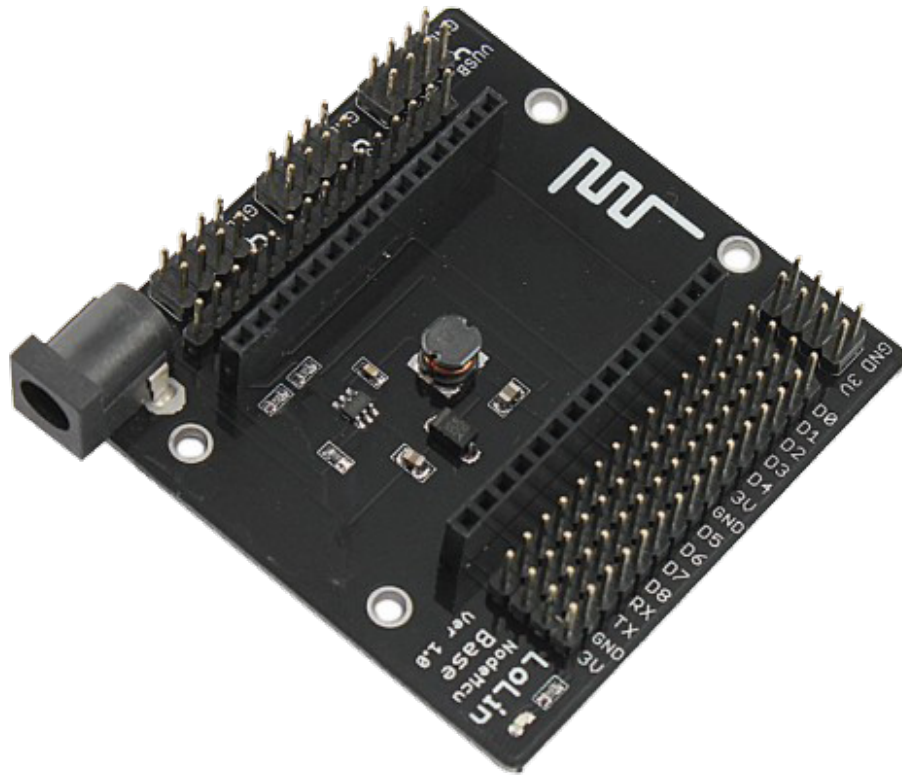
هـ) حامل لوحة NodeMCU الاصدار الاول:

عبارة عن قاعدة تثبت عليها لوحة NodeMCU و له عدة مزايا:

- حل مشكلة أبعاد اللوحة التي ذكرناها سابق والاسْتغناء عن لوحة تثبيت القطع الالكترونية.
 - يمكننا من استخدام مصدر تغذية خارجي (بطاريات) بقيمة جهد تتراوح بين 6 الى 24 فولت.
 - يحتوي على منفذ مخصص لتوصيل مصادر التغذية (البطاريات).
- لذلك، فإن من أسهل الحلول للتعامل مع لوحة NodeMCU هو استخدام هذا الحامل كما هو موضح في الصور التالية:



NodeMCU





الإصدار الثاني (V 1):

أ) الأطراف:

تحتوي اللوحة على عشرة أطراف (D10 - D1) نستطيع استخدامها كدخل أو كخرج (output أو input) وتدعم خاصية PWM.

وأیضا تحتوي على طرف

(D0) لا يدعم خاصية

.PWM

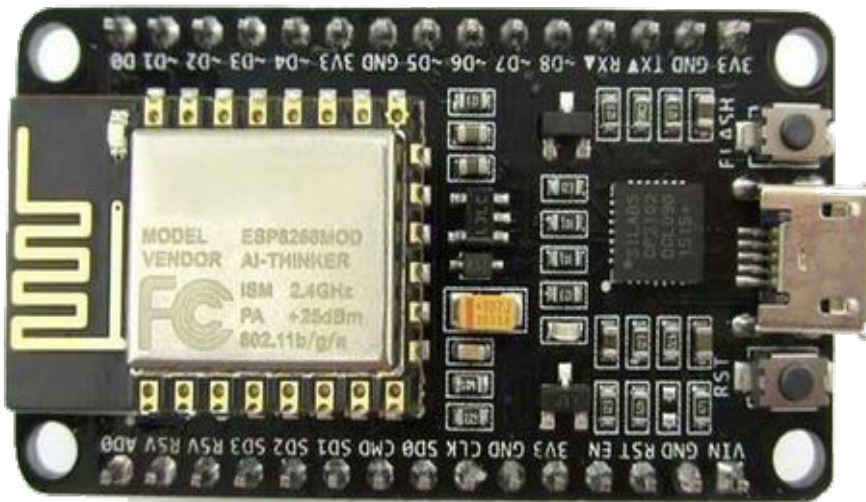
وتحتوي على طرف (A0)

نستطيع استخدامه كدخل

تماثلي (analog input).

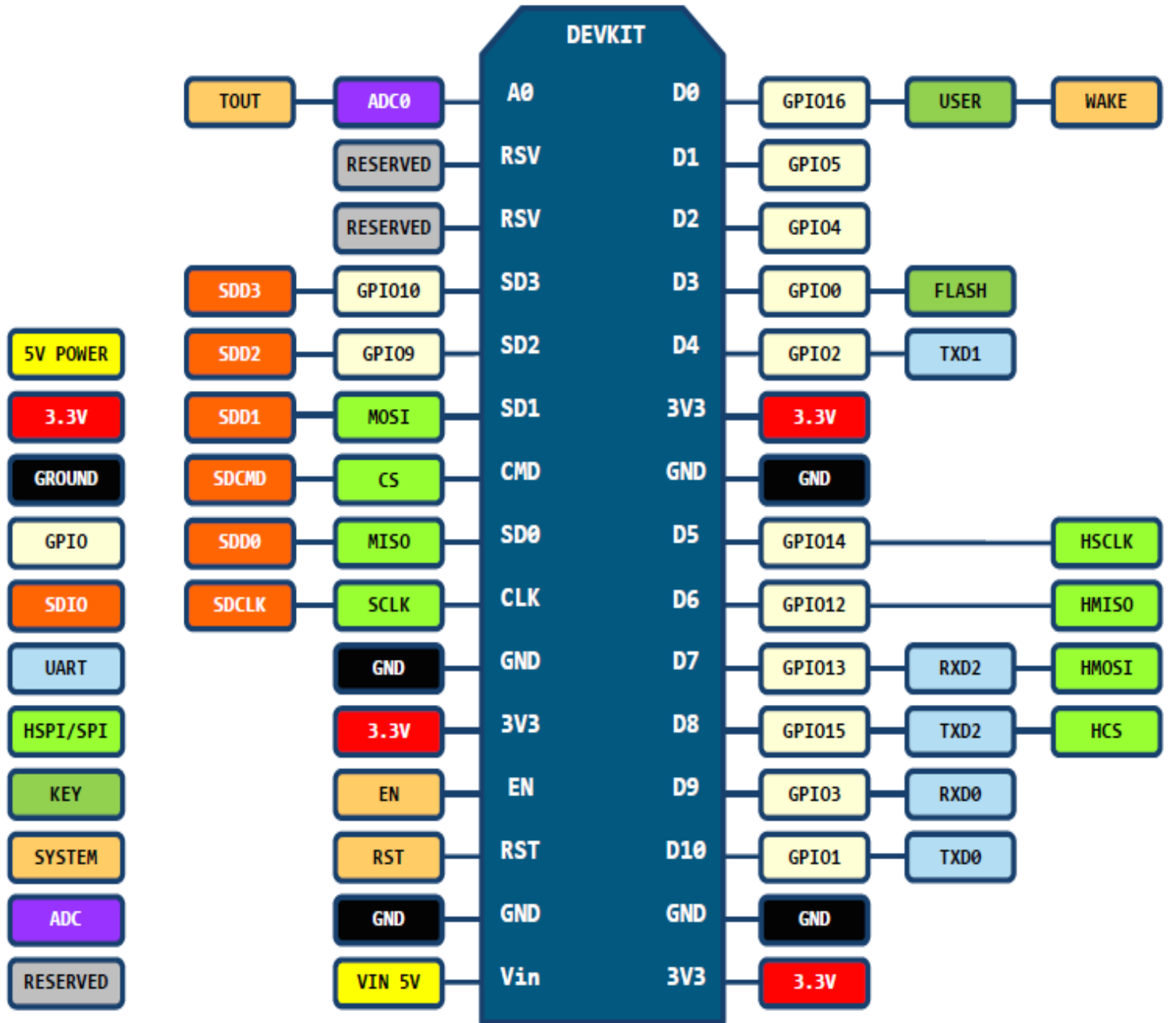
بالإضافة الى طرف V_{in} (3.3v - 10v) وثلاثة أطراف 3.3 فولت (3.3v)
و أربعة أطراف للأرضي (GND).

ستجدون تفاصيل الاطراف في مخطط اللوحة في الصورة التالية:





NodeMCU



نلاحظ أن هناك طرفين باللون الرمادي (NC) غير متصلة أي انها غير مستخدمة.



ب) المفاتيح:

تحتوي اللوحة على مفتاحين من النوع الضغط (push button).

* المفتاح الاول (flash button):

يستخدم عند تثبيت نظام NodeMCU

* المفتاح الثاني (rst button):

يستخدم عند رفع برنامج (شيفرة برمجية)

جديد على NodeMCU

ملاحظة:

إذا أردت استخدام أحد هذان المفتاحين،

يجب الضغط عليه قبل أن تمد اللوحة

بالطاقة.



ج) ابعاد اللوحة:

تأتي اللوحة في الاصدار الثاني بطول (47 ملم) وعرض (26 ملم). وهذا

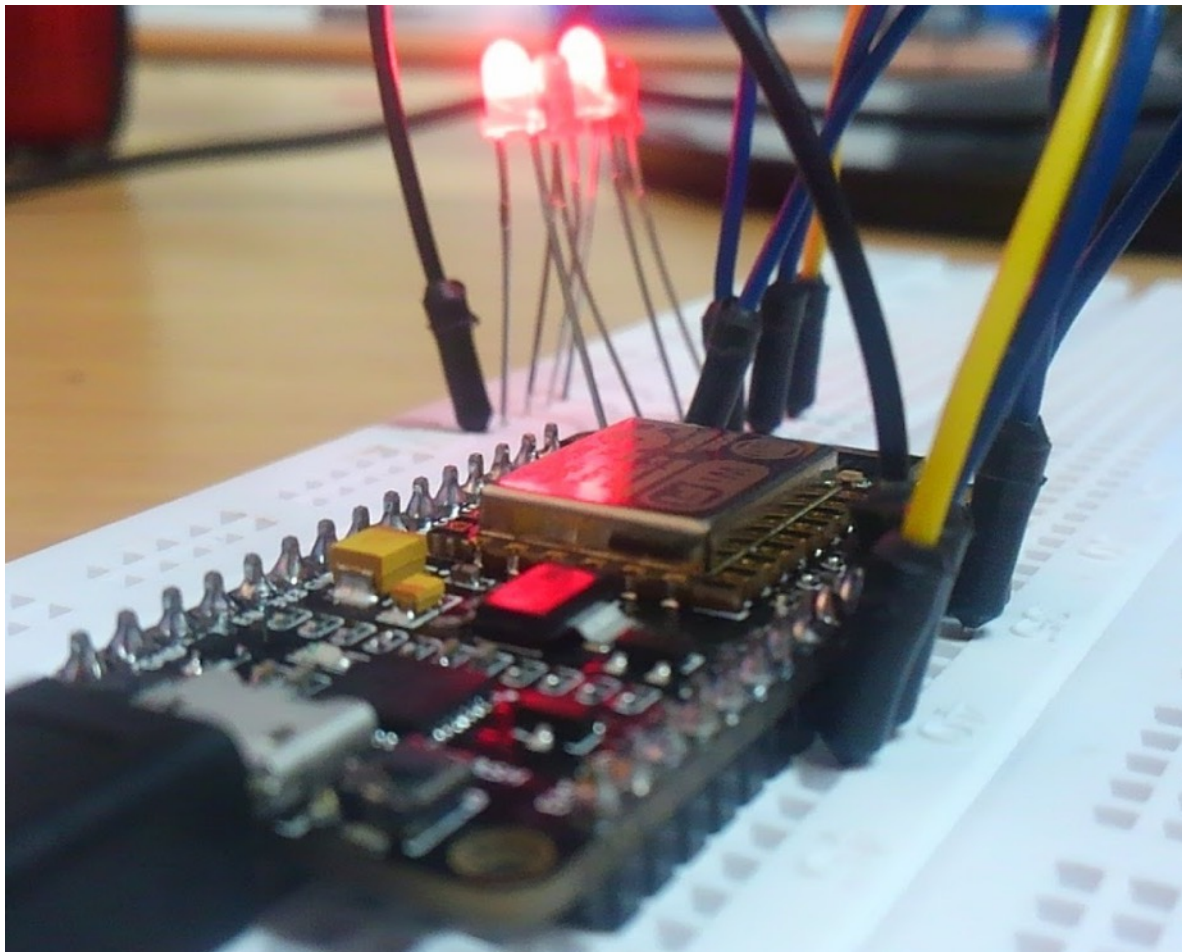
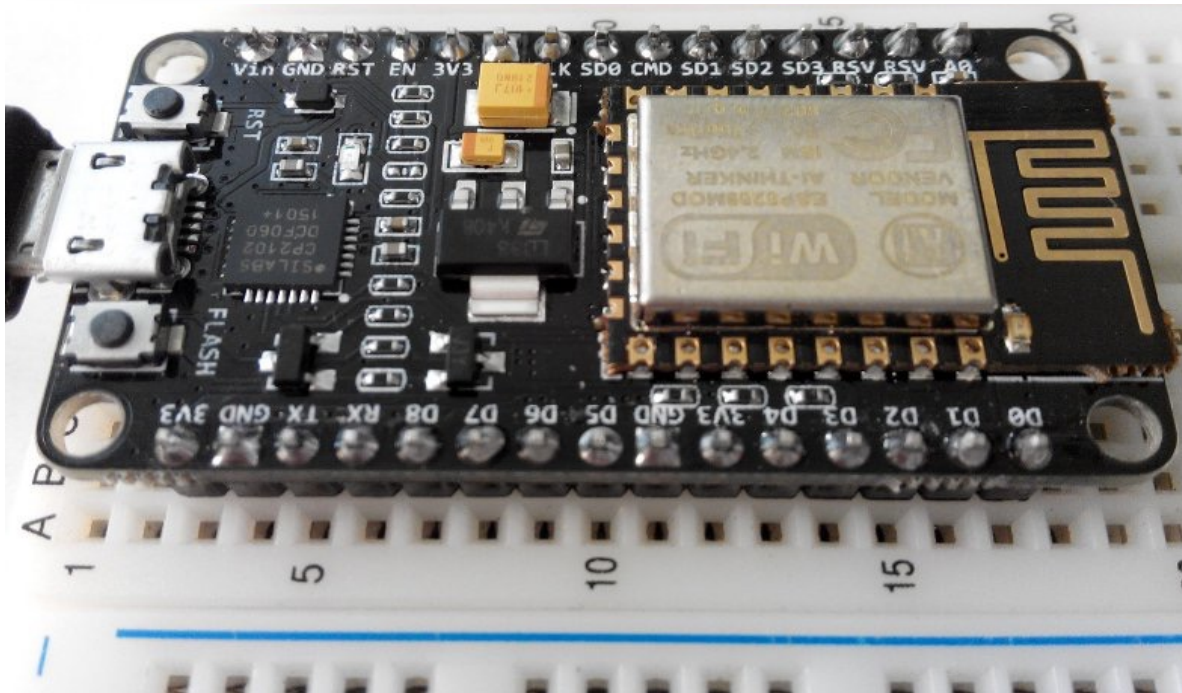
أحد المميزات التي يتفوق فيها الاصدار الثاني على الاول. حيث اننا نستطيع

استخدامها بسهولة مع لوحة تثبيت القطع الالكترونية (Breadboard)

كما تلاحظون في الصور التالية:



NodeMCU





د) الطاقة:

شرائح ESP8266 تعمل على جهد 3.3 فولت. وكذلك قيمة الخرج (output) في شرائح ESP8266 هي 3.3 فولت. ولكن لوحة NodeMCU الاصدار الثاني يمكن تشغيلها على جهد يتراوح بين 3.3 الى 10 فولت والسبب أنها مزودة بمنظم للجهد (regulator) لتخفيض الجهد الى 3.3 فولت

و يعتبر 10 فولت اعلى جهد يمكن استخدامه ويفضل استخدام 5 فولت.

هـ) حامل لوحة NodeMCU الاصدار الثاني:

عبارة عن قاعدة تثبت عليها لوحة NodeMCU و له عدة مزايا:

- يمكننا من استخدام مصدر تغذية خارجي (بطاريات) بقيمة جهد تتراوح بين 6v الى 24v.

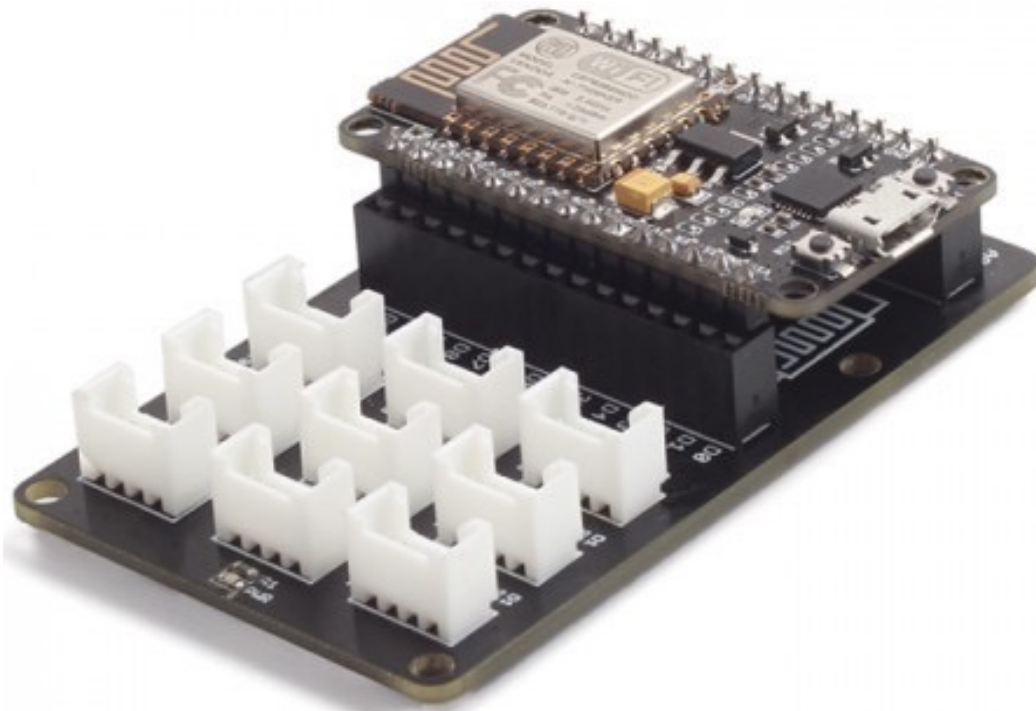
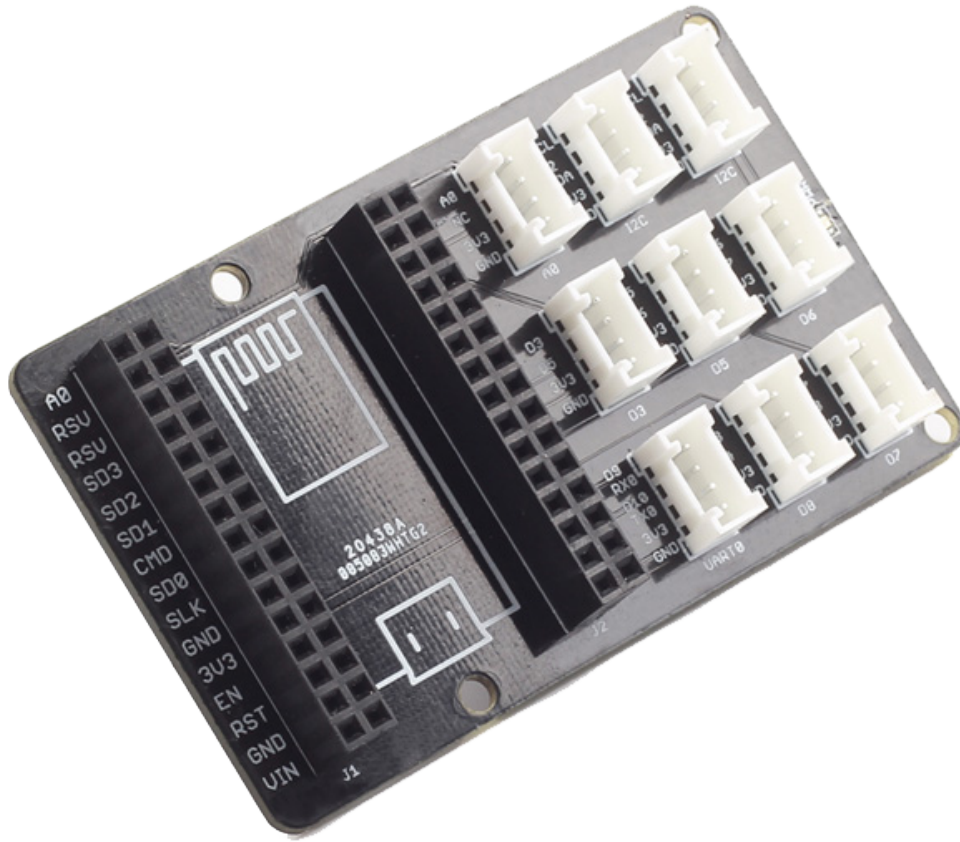
- يحتوي على منفذ مخصص لتوصيل مصادر التغذية (البطاريات).

لذلك، فإن من أسهل الحلول للتعامل مع لوحة NodeMCU هو استخدام

هذا الحامل كما هو موضح في الصور التالية:



NodeMCU





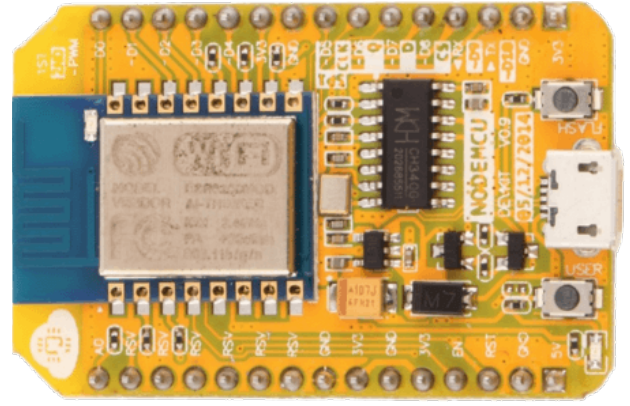
الباب الثالث

الادوات والقطع الالكترونية



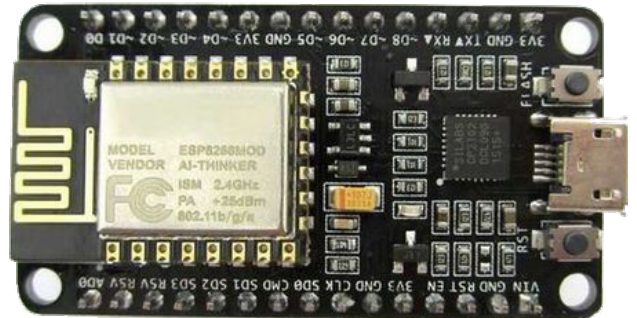
الأدوات والقطع المستخدمة:

إما الإصدار الأول من لوحة
NodeMCU



أو

الإصدار الثاني من لوحة
NodeMCU

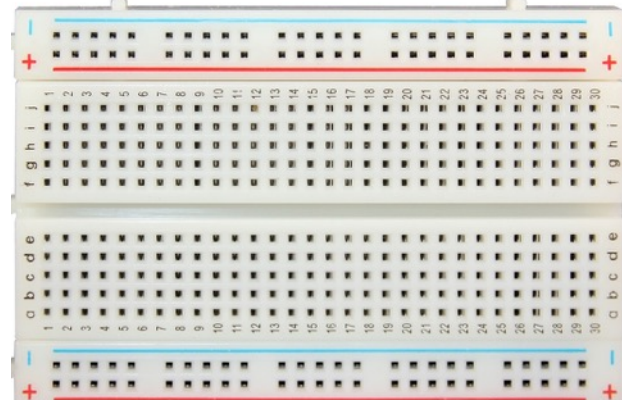


سلك (كابل)



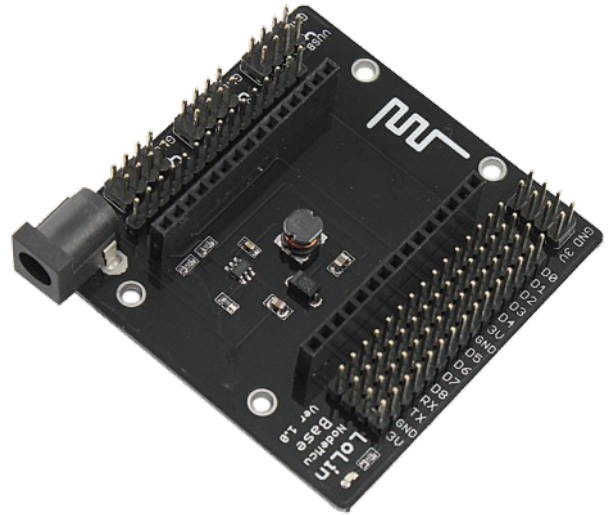


لوحة تجارب (Breadboard)



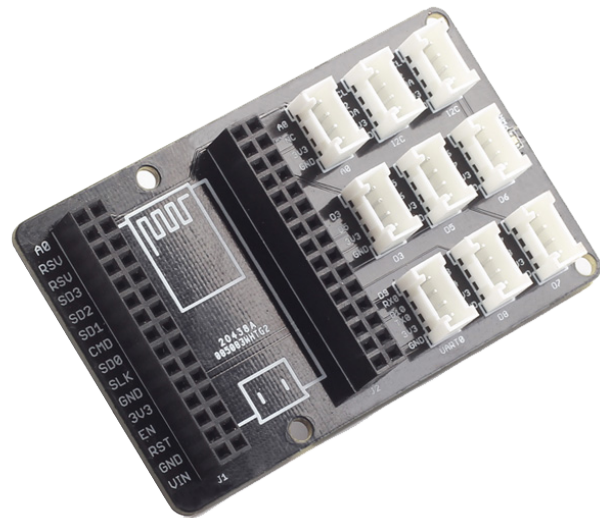
قاعدة الاصدار الاول من لوحة
NodeMCU

(يفضل شراءها عند استخدام
الاصدار الاول)



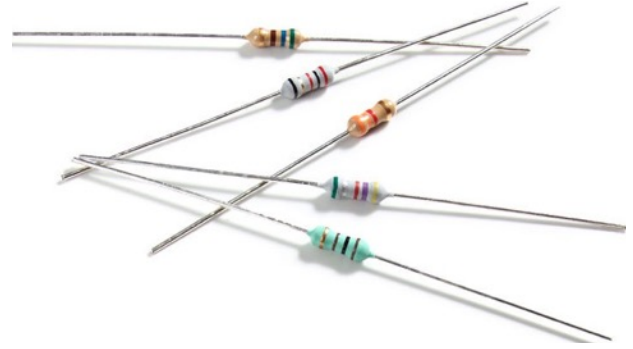
قاعدة الاصدار الثاني من لوحة
NodeMCU

(ليس بالضرورة شراءها عند
استخدام الاصدار الثاني)





مقاومات (Resistors) بقيم
مختلفة. سأذكر قيم المقاومات
وعدها في الامثلة



دايود باعث للضوء (LED)



دايود ضوئي ثلاثي (RGB LED)



مقاومة ضوئية (LDR)

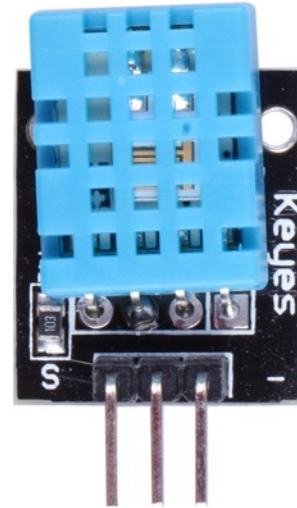




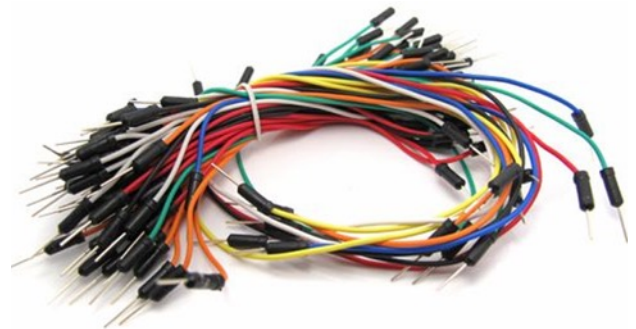
حساس كاشف حركة الأجسام (PIR)



حساس الحرارة والرطوبة
(DHT11)



اسلاك توصيل (jumper wire)





أسعار الأدوات والقطع:

أنصح بالشراء من المواقع الصينية مثل (aliexpress ، DealeXtreme) لأن المواقع الصينية تبيع بسعر أرخص وكميات أكثر. وتوفر خدمة التوصيل المجاني (قد تطول مدة التوصيل المجاني من 20 الى 60 يوم)

اسم القطعة	السعر
لوحة NodeMCU	يتراوح سعرها من 3 إلى 6 دولار
حامل لوحة NodeMCU	تقريبا 2.5 دولار
لوحة تثبيت القطع الإلكترونية	تقريبا 1.3 دولار
مقاومات	550 حبة بسعر 5 دولار
دايود باعث للضوء	100 حبة بسعر 4 دولار
دايود ضوئي ثلاثي	50 حبة بسعر 2.3 دولار
مقاومة ضوئية	10 حبات بسعر 0.5 دولار
حساس كاشف حركة الأجسام	تقريبا 1 دولار
حساس الحرارة والرطوبة	تقريبا 1 دولار
أسلاك توصيل	100 حبة بسعر 2 دولار

ملاحظة: هذه الأسعار بالنسبة لموقع aliexpress



الجزء الرابع

تنصيب نظام NodeMCU



الفصل الأول

نظام ماك و لينكس



في هذا الفصل سنقوم بتثبيت نظام NodeMCU خطوة خطوة باستخدام نظام ماك و لينكس. حيث أن طريقة التثبيت هي نفسها في كلا النظامين.

بدايتاً سنقوم بتحميل و تثبيت بعض البرامج قبل تثبيت النظام:

(1) برنامج **ch341 driver**:

هذا البرنامج خاص لمستخدمي نظام (ماك). أما بالنسبة لمستخدمي نظام (لينكس) فلا يحتاجون له. البرنامج موجود على الرابط التالي:

http://raysfiles.com/drivers/ch341ser_mac.zip

و بعد إنتهاء التحميل وفك الضغط سنقوم بتثبيتهما بطريقة تقليدية

(2) برنامج **silabs driver**:

هذا البرنامج يستخدم مع نظام (ماك) و نظام (لينكس) سنجده على الرابط التالي:

<https://www.silabs.com/products/mcu/Pages/USBtoUARTBridgeVCPDrivers.aspx>

بعد الدخول على الرابط سنختار منه النظام (ماك) أو (لينكس) وبعدها سيتم تحميل البرنامج. وبعد ذلك سنقوم بتثبيته بطريقة تقليدية.



3) برنامج Git:

هذا البرنامج يستخدم مع نظام (ماك) و نظام (لينكس) سنجده على الرابط التالي:

<http://git-scm.com/downloads>

بعد الدخول على الرابط سنختار منه النظام (ماك) أو (لينكس) وبعدها سيتم تحميل البرنامج. وبعد ذلك سنقوم بتثبيته بطريقة تقليدية.

4) برنامج python:

سنقوم بتحميل الاصدار **python 2.7** من خلال الرابط التالي:

<https://www.python.org/downloads/>

و بعد إنتهاء التحميل سنقوم بتثبيته بطريقة تقليدية.



5) برنامج pyserial:

نستطيع الحصول على هذا البرنامج من خلال الرابط التالي:

<https://pypi.python.org/pypi/pyserial>

بعد الدخول على الرابط سنجد ملفين كما في الصورة التالية:

File	Type
pyserial-3.0.tar.gz (md5)	Source
pyserial-3.0.win32.exe (md5)	MS Windows installer

الملف الأول لمستخدمي نظام (ماك) و نظام (لينكس) . أما الملف الثاني فهو لمستخدمي الويندوز. سنقوم بالضغط على الملف الأول وسيبدأ التحميل مباشرة ثم بعد ذلك سنقوم بفك الضغط.

ملاحظة: فقط نحتاج لتحميل هذا البرنامج ولا يتطلب تثبيت.



6) نظام NodeMCU:

الآن سنقوم بتحميل نظام NodeMCU من خلال الرابط التالي:

https://github.com/nodemcu/nodemcu-firmware/releases/tag/0.9.6-dev_20150704

بعد الدخول على الرابط سنلاحظ وجود 4 ملفات وسنقوم بتحميل الملف الذي يدعى (**nodemcu_integer_0.9.6-dev_20150704.bin**)

بعد ذلك سنقوم بفتح برنامج سطر الأوامر مثل:

برنامج الوحدة الطرفية (**Terminal**) على نظام الماك.

وبرنامج **Terminal** أو **LXTerminal** على نظام لينكس.

أو أي برنامج (سطر أوامر) آخر.



برنامج الوحدة الطرفية (Terminal) لنظام ماك:

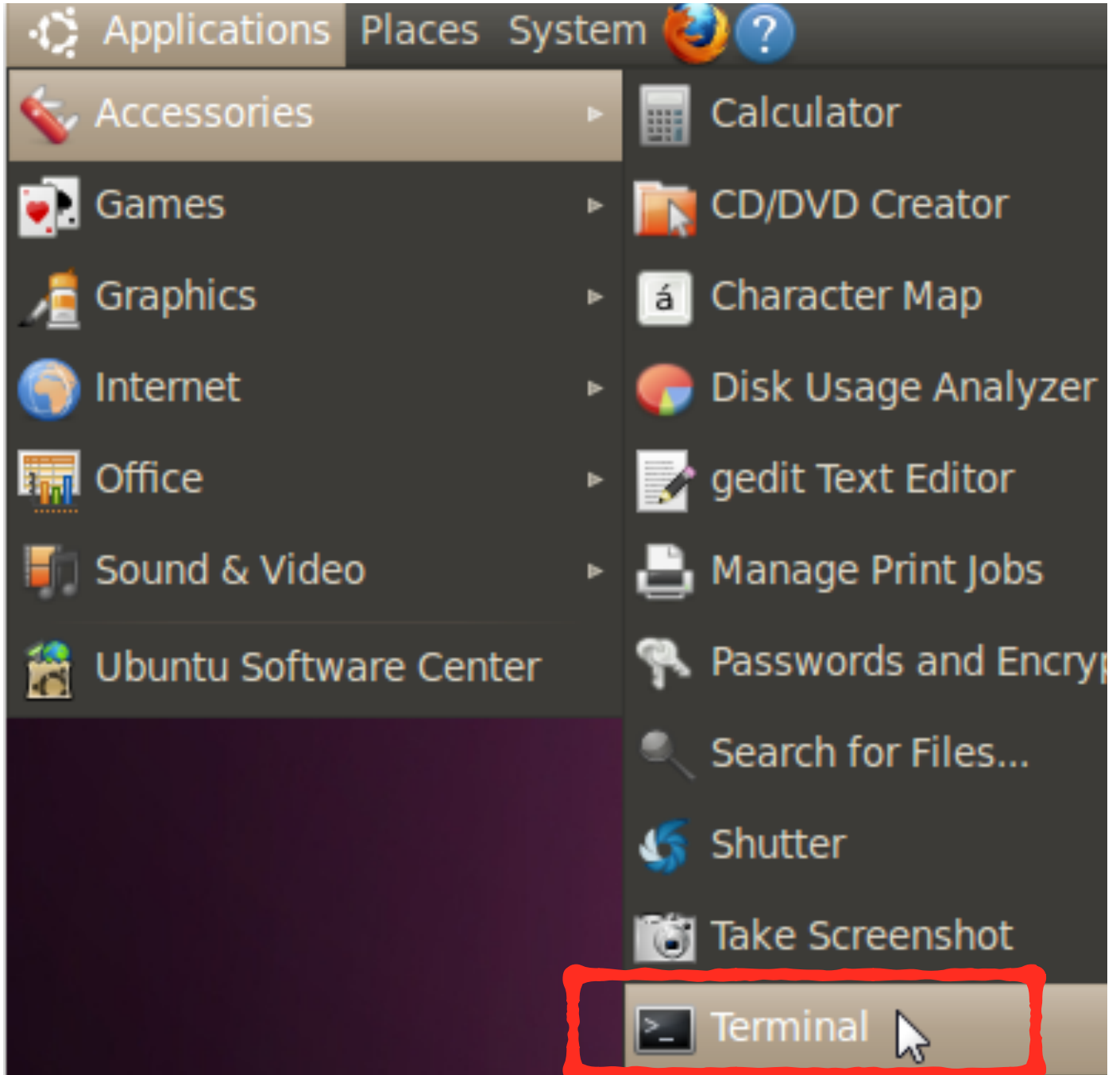




NodeMCU



برنامج Terminal لنظام لينكس:





NodeMCU



سنفتح برنامج الوحدة الطرفية (Terminal) وستظهر لنا الصفحة التالية:

```
inventor — -bash — 80x24
Last login: Wed Dec 23 03:27:20 on ttys000
MacBook-Pro-alkhas-b-jihad:~ inventor$
```

بما أنني استخدم نظام ماك فسأكمل بقية الشرح عليه. وبالنسبة لمستخدمي لينكس فسيتبعون معي بقية الخطوات لأن طريقة تثبيت نظام NodeMCU بإستخدام ماك أو لينكس ستكون بطريقة واحدة.



```
inventor — -bash
Last login: Wed Dec 23 03:27:20 on ttys000
MacBook-Pro-alkhas-b-jihad:~ inventor$
```

نلاحظ وجود صورة منزل وهذا يشير الى مجلد المنزل (Home) الموجود في Finder. وبالنسبة لي فقد سميت هذا المجلد باسم inventor. ملاحظة: يختلف اسم هذا المجلد من شخص لشخص.

حيث أن أي تحميلات سنقوم بها من خلال برنامج Terminal ستحفظ وتخزن داخل مجلد المنزل (Home) وبالنسبة لي فهو مجلد inventor.

بعد ذلك سنقوم بتحميل وتثبيت عدة أدوات من خلال Terminal وهي كالاتي:

- (1) أداة pip.
- (2) أداة pyserial.
- (3) أداة esptool.
- (4) تثبيت نظام NodeMCU.



(1) أداة pip:

غالباً بعد تثبيت برنامج بايثون (python) سيتم تثبيت هذه الأداة تلقائياً وللتحقق من ذلك سنقوم بكتابة الأمر التالي:

```
pip --version
```

```
inventor — -bash — 64x24
Last login: Thu Dec 24 16:51:02 on ttys000
[MacBook-Pro-alkhas-b-jihad:~ inventor$ pip --version
pip 7.1.2 from /Library/Frameworks/Python.framework/Versions/2.7
/lib/python2.7/site-packages (python 2.7)
MacBook-Pro-alkhas-b-jihad:~ inventor$
```

فإن ظهرت لنا نتيجة ان هناك اصدار pip 7.1.2 او غيره (غالباً ستظهر لنا هذه النتيجة) فستجاوز طريقة تثبيت pip و تنتقل الى أداة pyserial . ولكن إن لم تظهر نتيجة ان هناك اصدار لـ pip وظهرت لنا الرسالة التالية:

```
pip: command not found
```

فهذا يعنى انه يجب تثبيت أداة pip يدوياً بالطريقة التالية:



تثبيت pip يدوياً:

سنقوم بكتابة الامر التالي لتثبيت أداة pip:

```
sudo easy_install pip
```

```
inventor — -bash — 62x19
Last login: Thu Dec 24 00:17:55 on console
MacBook-Pro-alkhas-b-jihad:~ inventor$ sudo easy_install pip
```

ثم بعد ذلك نضغط على زر **Enter** وسيطلب منا ادخال الرقم السري لجهاز الماك (هذا الرقم هو الذي يستخدم لتسجيل دخول المستخدم عند بداية تشغيل الحاسب) كما هو موضح في الصورة التالية:



```
inventor — sudo — 62x19
Last login: Thu Dec 24 00:17:55 on console
[MacBook-Pro-alkhas-b-jihad:~ inventor$ sudo easy_install pip
Password: ?
```

ملاحظة: إذا كان جهاز الحاسب لا يتطلب رقم سري لتسجيل الدخول، فهذا يعني انه تلقائياً سيبدأ عملية تثبيت أداة pip.

ملاحظة: عند ادخال الرقم السري فإنه لا يظهر اي شيء على صفحة Terminal يدل على انه تم إدخال الرقم السري.

بعد إدخال الرقم السري سنضغط على زر **Enter** وبعدها سيتم تثبيت أداة pip كما هو موضح في الصورة التالية:



```
inventor — -bash — 80x24
warning: no previously-included files found matching '.travis.yml'
warning: no previously-included files found matching 'pip/_vendor/Makefile'
warning: no previously-included files found matching 'tox.ini'
warning: no previously-included files found matching 'dev-requirements.txt'
no previously-included directories found matching '.travis'
no previously-included directories found matching 'docs/_build'
no previously-included directories found matching 'contrib'
no previously-included directories found matching 'tasks'
no previously-included directories found matching 'tests'
creating /Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/site-packages/pip-7.1.2-py2.7.egg
Extracting pip-7.1.2-py2.7.egg to /Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/site-packages
Adding pip 7.1.2 to easy-install.pth file
Installing pip script to /Library/Frameworks/Python.framework/Versions/2.7/bin
Installing pip2.7 script to /Library/Frameworks/Python.framework/Versions/2.7/bin
Installing pip2 script to /Library/Frameworks/Python.framework/Versions/2.7/bin

Installed /Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/site-packages/pip-7.1.2-py2.7.egg
Processing dependencies for pip
Finished processing dependencies for pip
MacBook-Pro-alkhas-b-jihad:~ inventor$
```

وللتحقق من أنه تم التثبيت بنجاح سنقوم بكتابة الأمر الذي استخدمناه سابقاً لإظهار رقم الاصدار الذي تم تثبيته.

```
pip --version
```



بعض الأوامر الهامة لأداة **pip**:

أ) أمر تثبيت **pip**:

```
sudo easy_install pip
```

ب) أمر لمعرفة إصدار **pip**:

```
pip --version
```

ج) أمر لتحديث **pip**:

```
pip install -U pip
```

د) أمر لحذف **pip**:

```
sudo pip uninstall pip
```



(2) أداة pyserial:

لتحميل هذه الأداة سنقوم بكتابة الأمر التالي:

```
sudo pip install pyserial
```

```
inventor — sudo — 80x24
Last login: Sat Jan 2 05:57:20 on ttys000
MacBook-Pro-alkhas-b-jihad:~ inventor$ sudo pip install pyserial
Password: ?
```

بعد ذلك سيطلب منا ادخال الرقم السري لجهاز الماك (هذا الرقم هو الذي يستخدم لتسجيل دخول المستخدم عند بداية تشغيل الحاسب) كما هو موضح في الصورة السابقة:



NodeMCU



بعد إدخال الرقم السري سنضغط على زر **Enter** وبعدها سيتم تثبيت أداة **pyserial** كما هو موضح في الصورة التالية:

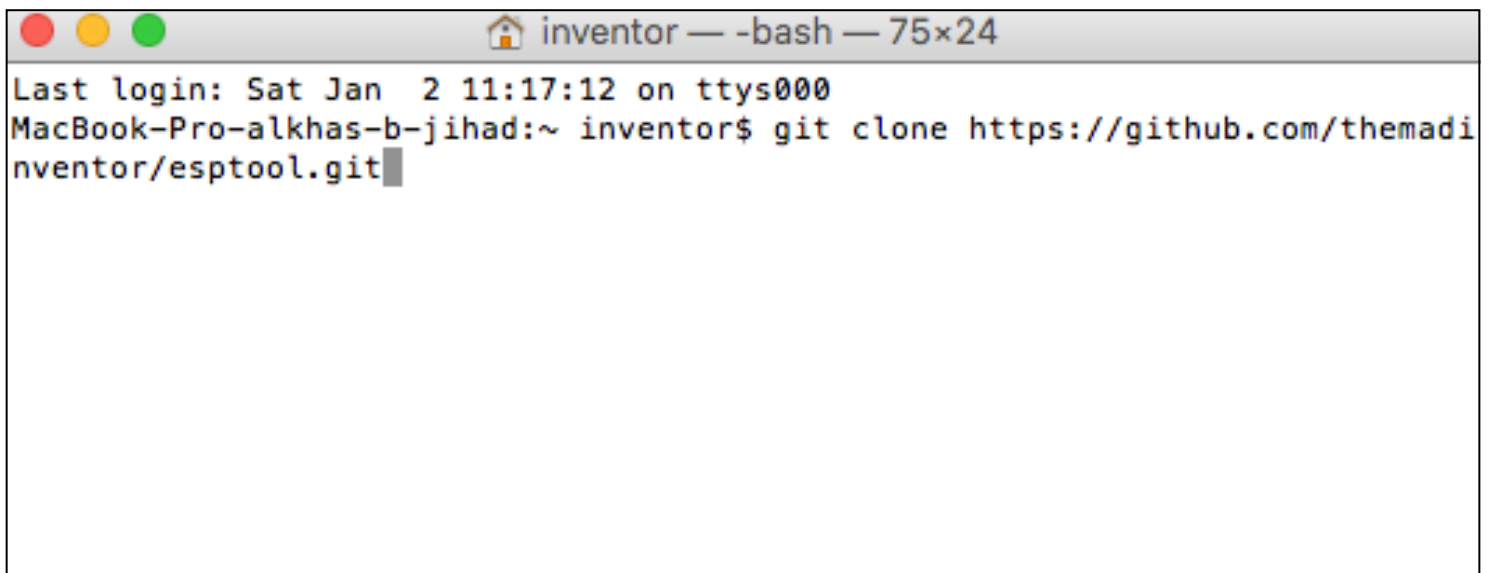
```
inventor — -bash — 80x24
Last login: Sat Jan  2 05:57:20 on ttys000
[MacBook-Pro-alkhas-b-jihad:~ inventor$ sudo pip install pyserial
>Password:
The directory '/Users/inventor/Library/Caches/pip/http' or its parent directory
is not owned by the current user and the cache has been disabled. Please check t
he permissions and owner of that directory. If executing pip with sudo, you may
want sudo's -H flag.
The directory '/Users/inventor/Library/Caches/pip' or its parent directory is no
t owned by the current user and caching wheels has been disabled. check the perm
issions and owner of that directory. If executing pip with sudo, you may want su
do's -H flag.
Collecting pyserial
  Downloading pyserial-3.0.tar.gz (133kB)
    100% |████████████████████████████████████████| 135kB 3.2MB/s
Installing collected packages: pyserial
  Running setup.py install for pyserial
Successfully installed pyserial-3.0
MacBook-Pro-alkhas-b-jihad:~ inventor$
```



(3) أداة esptool:

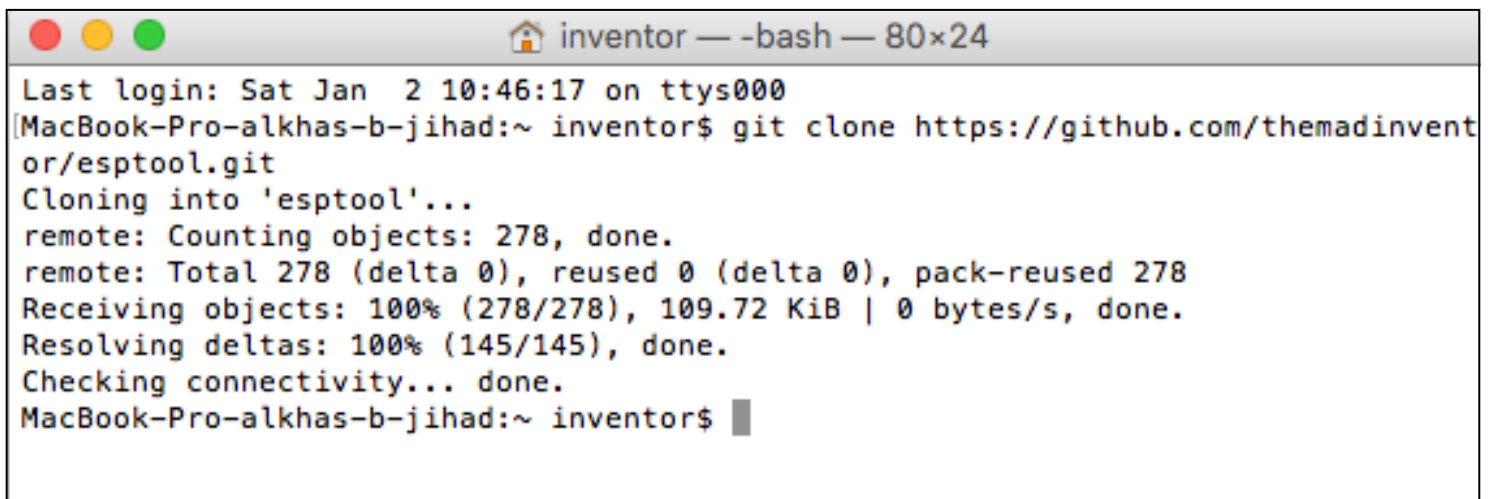
لتحميل هذه الأداة سنقوم بكتابة الأمر التالي:

```
git clone https://github.com/themadinventor/esptool.git
```



```
inventor — -bash — 75x24
Last login: Sat Jan  2 11:17:12 on ttys000
MacBook-Pro-alkhas-b-jihad:~ inventor$ git clone https://github.com/themadinventor/esptool.git
```

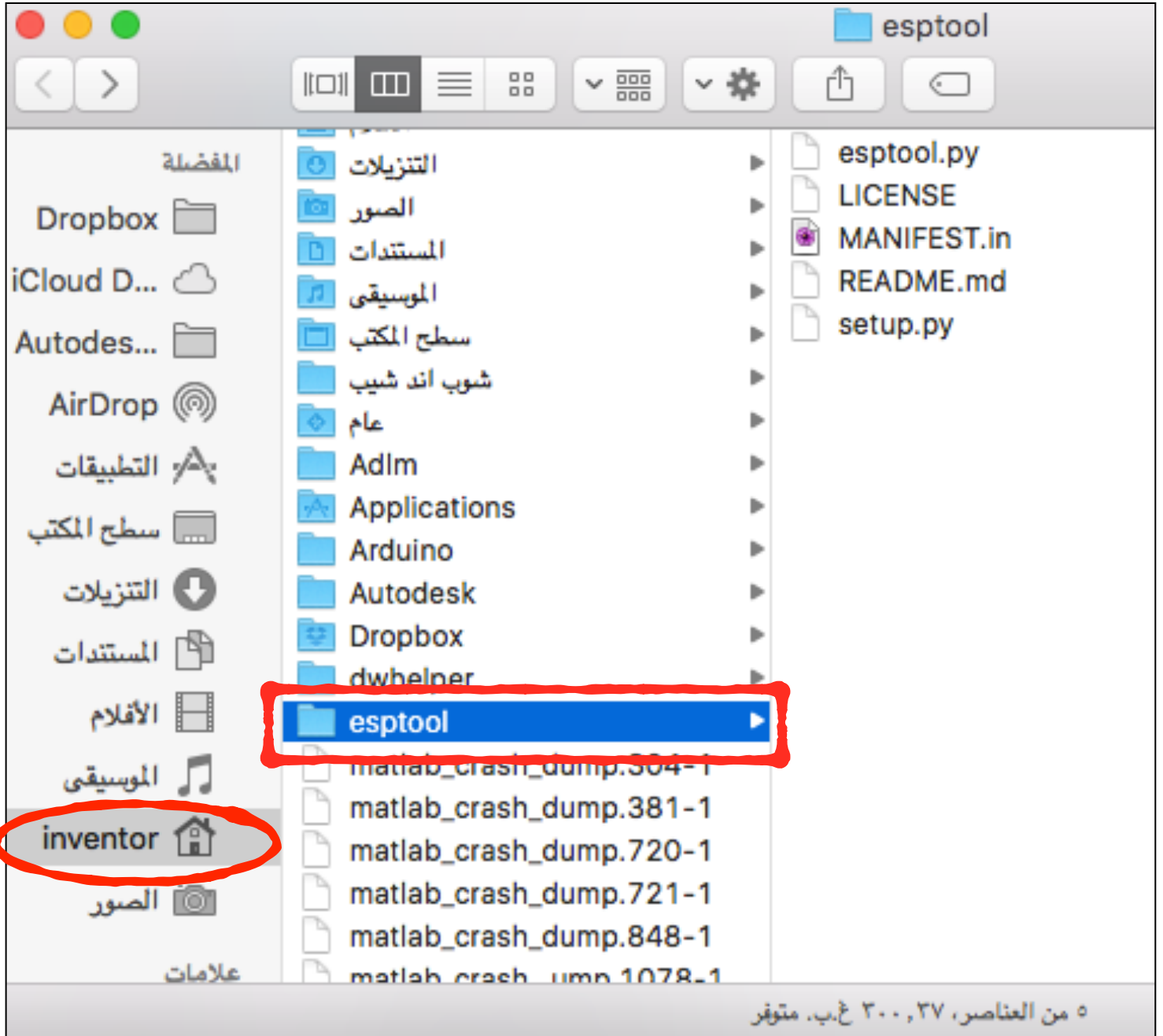
ثم بعد الضغط على زر **Enter** سيبدأ التحميل كما هو موضح في الصورة التالية:



```
inventor — -bash — 80x24
Last login: Sat Jan  2 10:46:17 on ttys000
MacBook-Pro-alkhas-b-jihad:~ inventor$ git clone https://github.com/themadinventor/esptool.git
Cloning into 'esptool'...
remote: Counting objects: 278, done.
remote: Total 278 (delta 0), reused 0 (delta 0), pack-reused 278
Receiving objects: 100% (278/278), 109.72 KiB | 0 bytes/s, done.
Resolving deltas: 100% (145/145), done.
Checking connectivity... done.
MacBook-Pro-alkhas-b-jihad:~ inventor$
```




بعد تحميل أداة esptool. سنجد مجلد بإسم esptool داخل مجلد المنزل (Home).



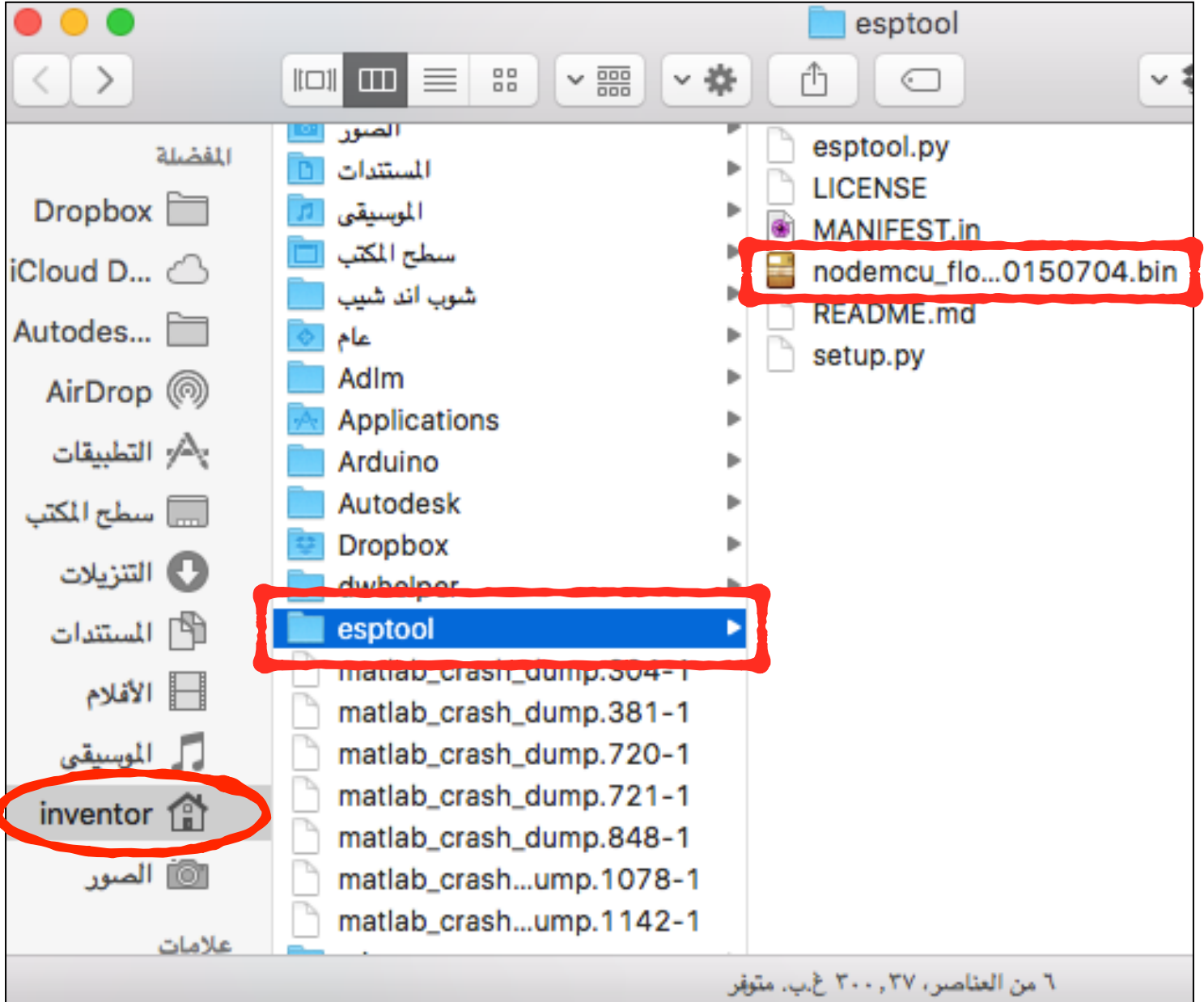
ملاحظة: بالنسبة لمستخدمي نظام (ماك) فإن مجلد المنزل يختلف تسميته من شخص لشخص. فبالنسبة لي فإن مجلد المنزل يدعى inventor



NodeMCU



بعد ذلك سنضع ملف نظام NodeMCU الذي قمنا بتحميله داخل مجلد esptool الموجود في مجلد المنزل كما هو موضح في الصورة التالية:





4) تثبيت نظام NodeMCU:

في الخطوات السابقة قمنا بوضع ملف نظام NodeMCU داخل مجلد esptool الموجود في مجلد المنزل (Home). وهذا يعني أن النظام يجب أن يثبت على مجلد esptool. لذلك سنقوم بتغيير مجلد Terminal من مجلد المنزل الى مجلد esptool عن طريق كتابة الأمر التالي:

```
cd esptool
```

```
inventor — bash — 80x24
Last login: Sat Jan  9 19:04:31 on ttys000
MacBook-Pro-alkhas-b-jihad:~ inventor$ cd esptool
```

ثم بعد الضغط على زر Enter سيتم تغيير مجلد المنزل (inventor) الى مجلد esptool كما هو موضح في الصورة التالية:



NodeMCU



```
esptool - bash - 80x24
Last login: Sat Jan 9 19:28:53 on ttys000
MacBook-Pro-alkhas-b-jihad:~ inventor$ cd esptool
MacBook-Pro-alkhas-b-jihad:esptool inventor$
```

وبعد ذلك سنقوم بتوصيل لوحة NodeMCU بالحاسب عن طريق منفذ USB. و سنكتب الأمر التالي لإظهار اسم منفذ USB المستخدم.

```
ls /dev/tty.*
```

```
esptool - bash - 80x24
Last login: Sat Jan 9 19:29:15 on ttys000
MacBook-Pro-alkhas-b-jihad:~ inventor$ cd esptool
MacBook-Pro-alkhas-b-jihad:esptool inventor$ ls /dev/tty.*
```

ثم بعد الضغط على زر Enter سيظهر لنا أسماء منافذ USB المستخدمة كما هو موضح في الصورة التالية:



```
esptool — bash — 80x24
Last login: Sat Jan 9 19:29:15 on ttys000
MacBook-Pro-alkhas-b-jihad:~ inventor$ cd esptool
MacBook-Pro-alkhas-b-jihad:esptool inventor$ ls /dev/tty.*
/dev/tty.Bluetooth-Incoming-Port      /dev/tty.wchusbserial1410
/dev/tty.Bluetooth-Modem
MacBook-Pro-alkhas-b-jihad:esptool inventor$ █
```

هذا هو اسم المنفذ (/dev/tty.wchusbserial1410) الذي تتصل به لوحة NodeMCU.

ملاحظة: يختلف اسم المنفذ من جهاز لآخر.

بعد ذلك سنقوم بالخطوة الاخيرة وسنكتب أمر تثبيت النظام على لوحة NodeMCU

```
sudo python esptool.py --port /dev/ttyUSB0 write_flash
0x00000 The_Path_To_The_NodeMCU_Firmware.bin
```

نستبدل /dev/ttyUSB0 بإسم منفذ USB الذي تتصل به لوحة NodeMCU

نستبدل The_Path_To_The_NodeMCU_Firmware.bin بإسم

ملف نظام NodeMCU الذي قمنا بتحميله سابقا.



NodeMCU



وبالنسبة لي سيصبح الأمر كالتالي:

```
sudo python esptool.py --port /dev/tty.wchusbserial1410  
write_flash 0x00000 nodemcu_float_0.9.6-dev_20150704.bin
```

```
esptool — bash — 80x24  
Last login: Sun Jan 10 01:58:39 on ttys000  
MacBook-Pro-alkhas-b-jihad:~ inventor$ cd esptool  
MacBook-Pro-alkhas-b-jihad:esptool inventor$ ls /dev/tty.*  
/dev/tty.Bluetooth-Incoming-Port      /dev/tty.wchusbserial1410  
/dev/tty.Bluetooth-Modem  
MacBook-Pro-alkhas-b-jihad:esptool inventor$ sudo python esptool.py --port /dev/  
tty.wchusbserial1410 write_flash 0x00000 nodemcu_float_0.9.6-dev_20150704.bin
```

بعد الضغط على زر Enter سيطلب منا إدخال الرقم السري لجهاز الماك (هذا الرقم هو الذي يستخدم لتسجيل دخول المستخدم عند بداية تشغيل الحاسب) كما هو موضح في الصورة التالية:

```
esptool — sudo — 80x24  
Last login: Sun Jan 10 01:58:39 on ttys000  
MacBook-Pro-alkhas-b-jihad:~ inventor$ cd esptool  
MacBook-Pro-alkhas-b-jihad:esptool inventor$ ls /dev/tty.*  
/dev/tty.Bluetooth-Incoming-Port      /dev/tty.wchusbserial1410  
/dev/tty.Bluetooth-Modem  
MacBook-Pro-alkhas-b-jihad:esptool inventor$ sudo python esptool.py --port /dev/  
tty.wchusbserial1410 write_flash 0x00000 nodemcu_float_0.9.6-dev_20150704.bin  
Password:
```



بعد إدخال الرقم السري والضغط على زر Enter سيظهر لنا أحد هاتين الحالتين:

الحالة الأولى:

سيبدأ عملية تثبيت نظام NodeMCU على اللوحة مباشرة كما هو موضح

في الصورة التالية:

```
esptool - Python - 80x24
Last login: Sun Jan 10 01:58:39 on ttys000
MacBook-Pro-alkhas-b-jihad:~ inventor$ cd esptool
MacBook-Pro-alkhas-b-jihad:esptool inventor$ ls /dev/tty.*
/dev/tty.Bluetooth-Incoming-Port      /dev/tty.wchusbserial1410
/dev/tty.Bluetooth-Modem
MacBook-Pro-alkhas-b-jihad:esptool inventor$ sudo python esptool.py --port /dev/
tty.wchusbserial1410 write_flash 0x00000 nodemcu_float_0.9.6-dev_20150704.bin
Password:
Connecting...
Erasing flash...
Took 1.66s to erase flash block
Writing at 0x0001ac00... (23 %)
```

سننتظر حتى ينتهي التثبيت. وبعدها ستظهر لنا رسالة تفيد بأنه تم الانتهاء

من التثبيت كما هو موضح في الصورة التالية:



```
esptool — bash — 80x24
Last login: Sun Jan 10 01:58:39 on ttys000
MacBook-Pro-alkhas-b-jihad:~ inventor$ cd esptool
MacBook-Pro-alkhas-b-jihad:esptool inventor$ ls /dev/tty.*
/dev/tty.Bluetooth-Incoming-Port      /dev/tty.wchusbserial1410
/dev/tty.Bluetooth-Modem
MacBook-Pro-alkhas-b-jihad:esptool inventor$ sudo python esptool.py --port /dev/
tty.wchusbserial1410 write_flash 0x000000 nodemcu_float_0.9.6-dev_20150704.bin
Password:
Connecting...
Erasing flash...
Took 1.66s to erase flash block
Wrote 462848 bytes at 0x00000000 in 55.2 seconds (67.1 kbit/s)...
Leaving...
MacBook-Pro-alkhas-b-jihad:esptool inventor$
```

نستطيع بعد ذلك إطفاء Terminal وفصل لوحة NodeMCU من الحاسب.

الحالة الثانية:

ستظهر لنا رسالة بأن هناك خطأ كما هو موضح في الصورة التالية:

```
esptool — bash — 80x24
Last login: Sun Jan 10 01:58:39 on ttys000
MacBook-Pro-alkhas-b-jihad:~ inventor$ cd esptool
MacBook-Pro-alkhas-b-jihad:esptool inventor$ ls /dev/tty.*
/dev/tty.Bluetooth-Incoming-Port      /dev/tty.wchusbserial1410
/dev/tty.Bluetooth-Modem
MacBook-Pro-alkhas-b-jihad:esptool inventor$ sudo python esptool.py --port /dev/
tty.wchusbserial1410 write_flash 0x000000 nodemcu_float_0.9.6-dev_20150704.bin
Password:
Connecting...
A fatal error occurred: Failed to connect to ESP8266
```

ولحل هذه المشكلة سنقوم بعمل الخطوات التالية:



- فصل لوحة NodeMCU من الحاسب.
- الضغط المستمر المطول على مفتاح الفلاش (Flash button) الموجود على لوحة NodeMCU ثم سنقوم بتوصيل اللوحة بالحاسب مع الإستمرار بالضغط (إستمر بالضغط أثناء توصيل اللوحة بالحاسب).
- بعد توصيل لوحة NodeMCU بالحاسب سنتوقف عن الضغط على مفتاح الفلاش (Flash button).
- سنقوم بكتابة أمر تثبيت النظام مرة أخرى:

```
sudo python esptool.py --port /dev/tty.wchusbserial1410  
write_flash 0x00000 nodemcu_float_0.9.6-dev_20150704.bin
```

```
esptool — sudo — 80x24  
Last login: Sun Jan 10 01:58:39 on ttys000  
MacBook-Pro-alkhas-b-jihad:~ inventor$ cd esptool  
MacBook-Pro-alkhas-b-jihad:esptool inventor$ ls /dev/tty.*  
/dev/tty.Bluetooth-Incoming-Port      /dev/tty.wchusbserial1410  
/dev/tty.Bluetooth-Modem  
MacBook-Pro-alkhas-b-jihad:esptool inventor$ sudo python esptool.py --port /dev/  
tty.wchusbserial1410 write_flash 0x00000 nodemcu_float_0.9.6-dev_20150704.bin  
Password:
```

بعد إدخال الرقم السري والضغط على زر Enter سيبدأ عملية تثبيت نظام NodeMCU على اللوحة كما هو موضح في الصورة التالية:



NodeMCU



```
esptool — Python — 80x24
Last login: Sun Jan 10 01:58:39 on ttys000
MacBook-Pro-alkhas-b-jihad:~ inventor$ cd esptool
MacBook-Pro-alkhas-b-jihad:esptool inventor$ ls /dev/tty.*
/dev/tty.Bluetooth-Incoming-Port      /dev/tty.wchusbserial1410
/dev/tty.Bluetooth-Modem
MacBook-Pro-alkhas-b-jihad:esptool inventor$ sudo python esptool.py --port /dev/
tty.wchusbserial1410 write_flash 0x00000 nodemcu_float_0.9.6-dev_20150704.bin
Password:
Connecting...
Erasing flash...
Took 1.66s to erase flash block
Writing at 0x0001ac00... (23 %)█
```

سننتظر حتى ينتهي التثبيت. وبعدها ستظهر لنا الرسالة تفيد بأنه تم الانتهاء من التثبيت كما هو موضح في الصورة التالية:

```
esptool — bash — 80x24
Last login: Sun Jan 10 01:58:39 on ttys000
MacBook-Pro-alkhas-b-jihad:~ inventor$ cd esptool
MacBook-Pro-alkhas-b-jihad:esptool inventor$ ls /dev/tty.*
/dev/tty.Bluetooth-Incoming-Port      /dev/tty.wchusbserial1410
/dev/tty.Bluetooth-Modem
MacBook-Pro-alkhas-b-jihad:esptool inventor$ sudo python esptool.py --port /dev/
tty.wchusbserial1410 write_flash 0x00000 nodemcu_float_0.9.6-dev_20150704.bin
Password:
Connecting...
Erasing flash...
Took 1.66s to erase flash block
Wrote 462848 bytes at 0x00000000 in 55.2 seconds (67.1 kbit/s)...
Leaving...
MacBook-Pro-alkhas-b-jihad:esptool inventor$ █
```

نستطيع بعد ذلك إطفاء Terminal وفصل لوحة NodeMCU من الحاسب



الفصل الثلثي

لنظام ويندوز



في هذا الفصل سنقوم بتثبيت نظام NodeMCU خطوة خطوة باستخدام نظام (ويندوز).

بدايتاً سنقوم بتحميل و تثبيت بعض البرامج قبل تثبيت النظام:

1) برنامج **silabs driver**:

سنجد هذا البرنامج على الرابط التالي:

<https://www.silabs.com/products/mcu/Pages/USBtoUARTBridgeVCPDrivers.aspx>

بعد الدخول على الرابط سنختار منه النظام (ويندوز) وبعدها سيتم تحميل البرنامج. وبعد ذلك سنقوم بتثبيته بطريقة تقليدية.

2) برنامج **CH340 driver**:

سنجد هذا البرنامج على الرابط التالي:

<http://www.14core.com/drivers/>

بعد الدخول على الرابط سنلاحظ وجود العديد من الملفات وسنقوم بتحميل الملف الذي يدعى (**CH340 Serial Communication Driver**) كما هو موضح في الصورة التالية:



WCH 沁恒

NodeMCU Serial Communication Driver

CH340 Serial Communication Driver for Mac | [Zip](#)

CH340 Serial Communication Driver for Windows 32/Vista/Server 2008/Win7/Win8/Win8.1 | [Zip](#)

CH340 Serial Communication Driver for Linux | [Zip](#)

CH340G Serial Communication Driver for Windows | [Zip](#)

CH340G Serial Communication Driver for Mac | [Zip](#)

CH340G Serial Communication Driver for Mac Old | [Zip](#)

CH340G Serial Communication Driver for Linux (Not Required)

وبعد التحميل وفك الضغط سنقوم بفتح الملف الذي يدعى (SETUP.EXE)
لتثبيت البرنامج كما هو موضح في الصورة التالية:

الاسم	الحجم	المحزوم	النوع	التعديل	CRC32
..			مجلد ملفات		
DRVSETUP64			مجلد ملفات	٢٠١٦/١٠/٢٣ م	
CH341PT.DLL	٦,٧١٢	٢,٣٧٦	ملحق التطبيق	١٢:٠٠ ٢٤/٠٦/٢٦ ...	V٣E535E2
CH341S64.SYS	٥٩,٩٠٤	٢٩,٠٤٦	ملف نظام	١٢:٠٠ ٠٦/٠٤/٢٦ ...	١DB90F64
CH341S98.SYS	١٩,٦٨٠	١١,٥٥٤	ملف نظام	١٢:٠٠ ٢٦/٠٥/٢٨ ...	٢D6CF845
ch341SER.CAT	١٠,٤٦٦	٥,٧٥٤	كتالوج الأمان	٠٦:٤٣ ١٧/٠٤/٢٦ ...	٨٤F3124D
CH341SER.INF	٦,٦٧٨	١,٧٩٥	معلومات الإعداد	١٢:٠٠ ١٢/١٠/٢٥ ...	F5890558
CH341SER.SYS	٤١,٤٧٢	٢٢,٧٧٥	ملف نظام	١٢:٠٠ ٠٦/٠٤/٢٦ ...	٩VECDC17
CH341SER.VXD	٢٠,٠٨٩	٩,٩٩٦	Virtual device driver	١٢:٠٠ ٢٠/١٢/٢٩ ...	٩VBC4AC4
SETUP.EXE	٨٩,٧٨٤	٢٦,٩٩٧	التطبيق	٠٥:٤١ ٢٣/٠٤/٢٦ م	E44A2091



(3) نظام NodeMCU:

الآن سنقوم بتحميل نظام NodeMCU من خلال الرابط التالي:

https://github.com/nodemcu/nodemcu-firmware/releases/tag/0.9.6-dev_20150704

بعد الدخول على الرابط سنلاحظ وجود 4 ملفات وسنقوم بتحميل الملف الذي يدعى (`nodemcu_integer_0.9.6-dev_20150704.bin`)

(4) برنامج `nodemcu flasher`:

هذا البرنامج يستخدم لتثبيت نظام NodeMCU سنجده على الرابط التالي:

<https://github.com/nodemcu/nodemcu-flasher>



(5) برنامج putty:

سنجد هذا البرنامج على الرابط التالي:

<http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>

بعد الدخول على الرابط سنلاحظ وجود العديد من الملفات وسنقوم بتحميل الملف الذي يدعى (**putty.exe**) كما هو موضح في الصورة التالية:

For Windows on Intel x86

PuTTY:	putty.exe
PuTTYtel:	puttytel.exe
PSCP:	pscp.exe
PSFTP:	psftp.exe



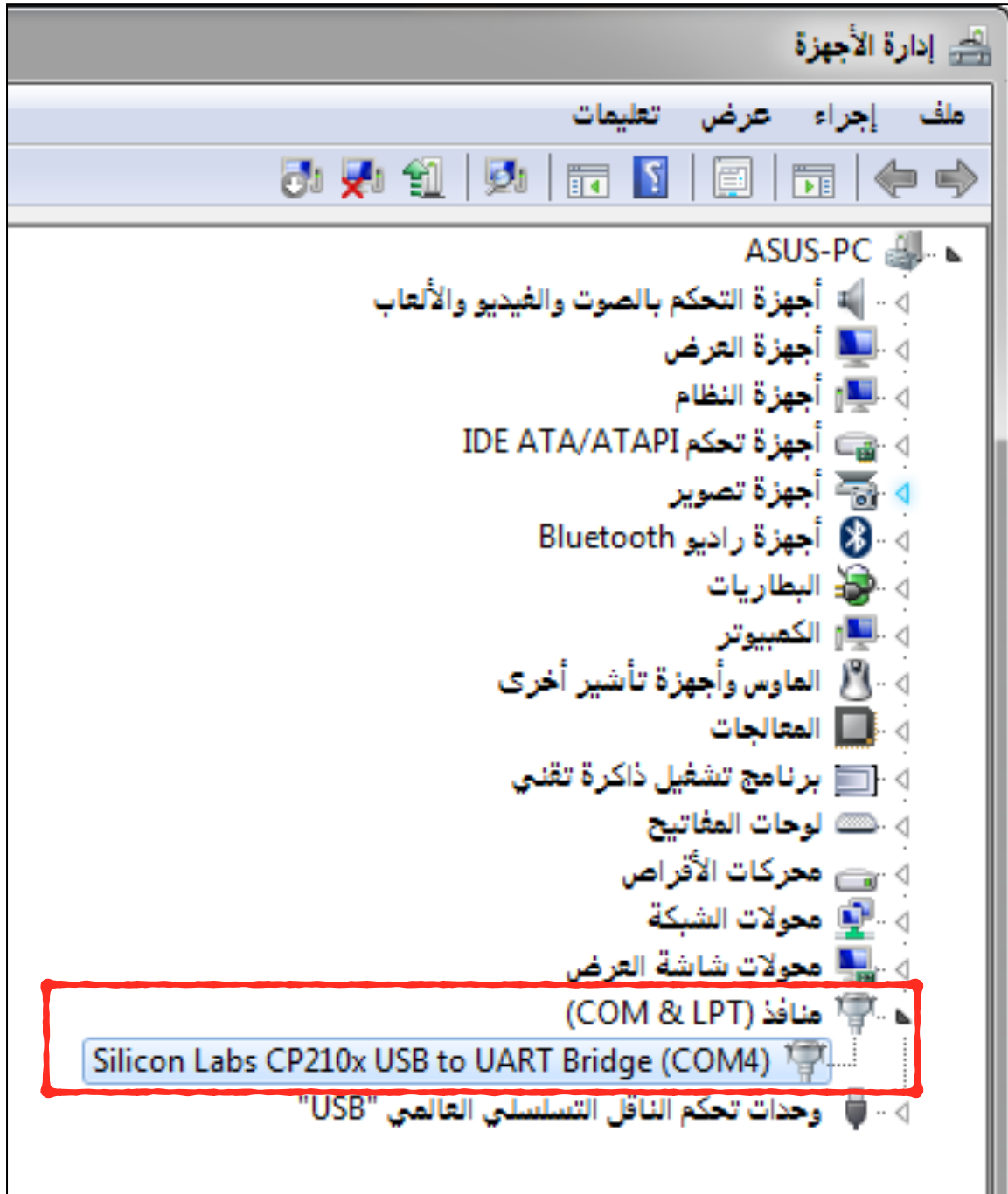
بعد ذلك سنقوم بعمل عدة خطوات لنتمكن من تثبيت نظام NodeMCU

1) معرفة اسم المنفذ الذي تتصل به لوحة NodeMCU:

سنقوم بتوصيل لوحة NodeMCU بالحاسب. وبعد ذلك سنذهب الى لوحة التحكم ثم الى إدارة الأجهزة كما هو موضح في الصورة التالية:



وبعد الضغط على ادارة الاجهزة سيظهر لنا اسم المنفذ الذي تتصل به لوحة NodeMCU كما هو موضح في الصورة التالية:



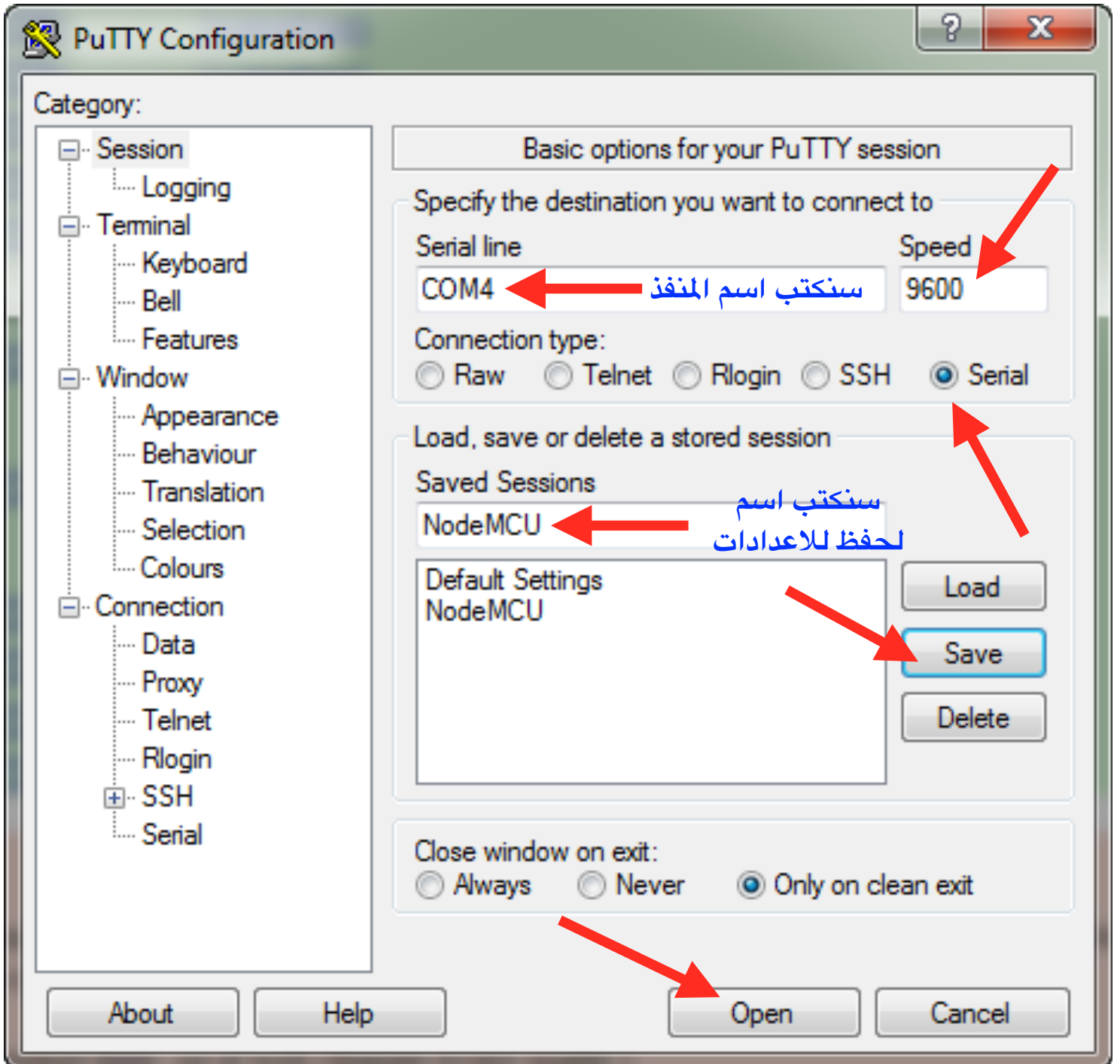
بالنسبة لي فإن اسم او رقم المنفذ هو (COM4) الذي تتصل به لوحة NodeMCU.

ملاحظة: يختلف اسم المنفذ من جهاز لآخر.



(2) استخدام برنامج Putty:

بعد أن نفتح برنامج Putty سنقوم ببعض التعديلات كما هو موضح في الصورة التالية:





NodeMCU



ثم بعد الضغط على زر **Open** ستظهر لنا الشاشة التالية:

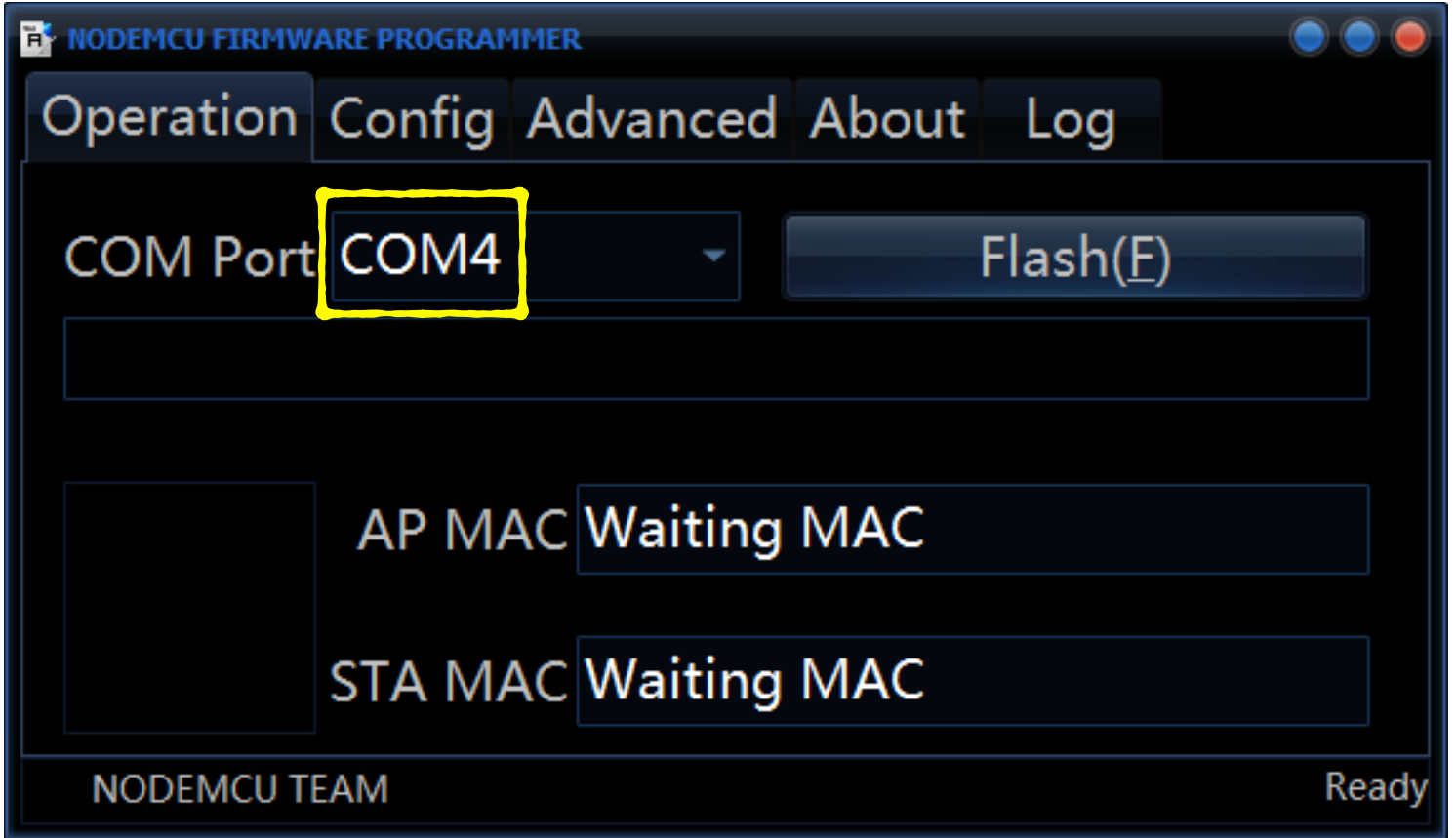


ليس بالمهم ما سيظهر لنا على الشاشة من كتابات بل و قد لا يظهر أي شيء.
فسنقوم بعد ذلك بإغلاق هذه الشاشة والخروج منها.

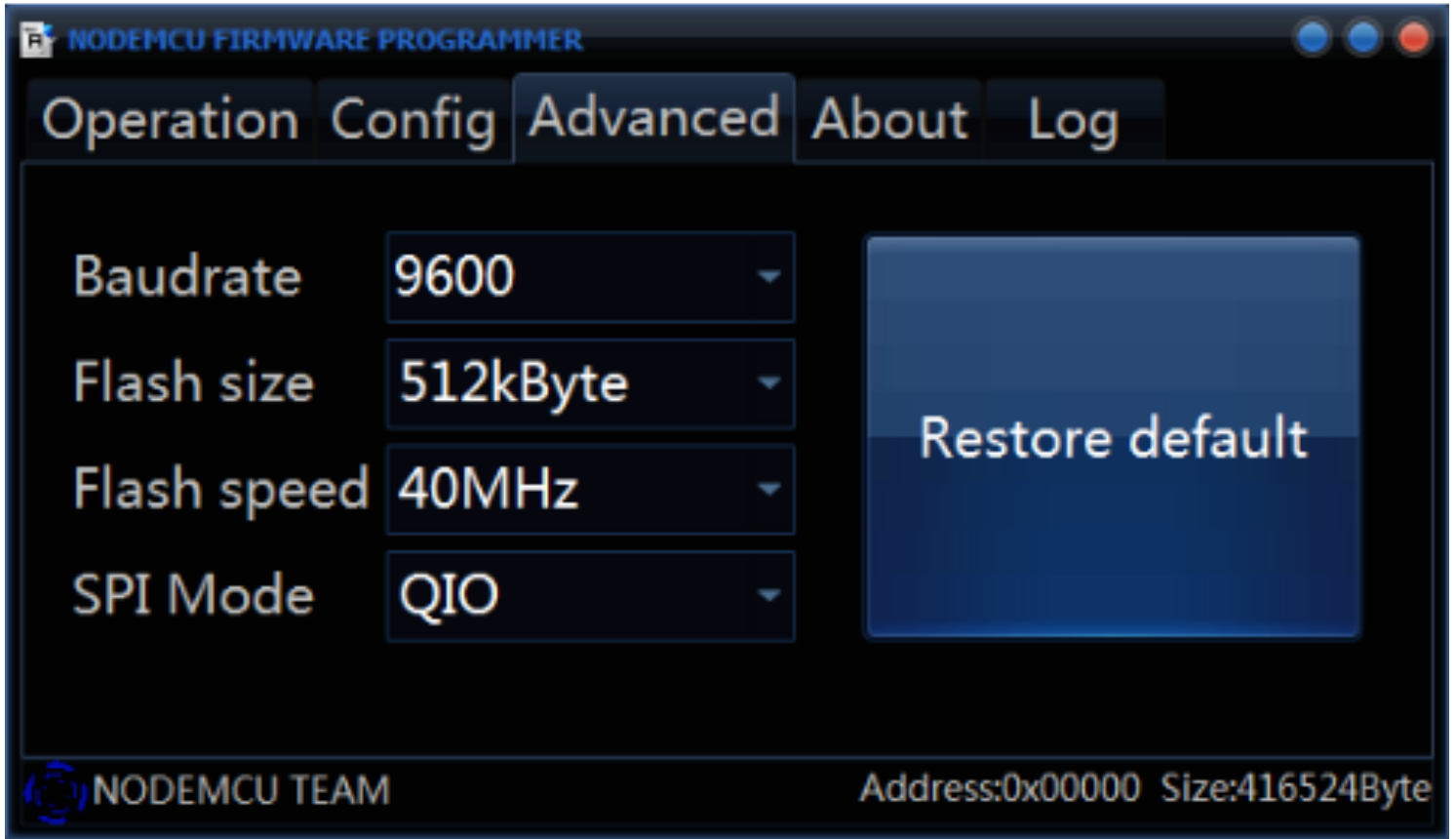


3) تثبيت نظام NodeMCU:

سنقوم بفتح برنامج (nodemcu flasher) الذي قمنا بتحميله سابقاً
وستظهر لنا الشاشة التالية:



نتأكد من أن اسم المنفذ هو المنفذ الصحيح. ثم بعد ذلك سنضغط على خانة
(**Advanced**) وستظهر لنا الشاشة التالية:



نتأكد من أن البيانات هي كالتالي:

• 9600 = Baudrate

• 512kByte = Flash size

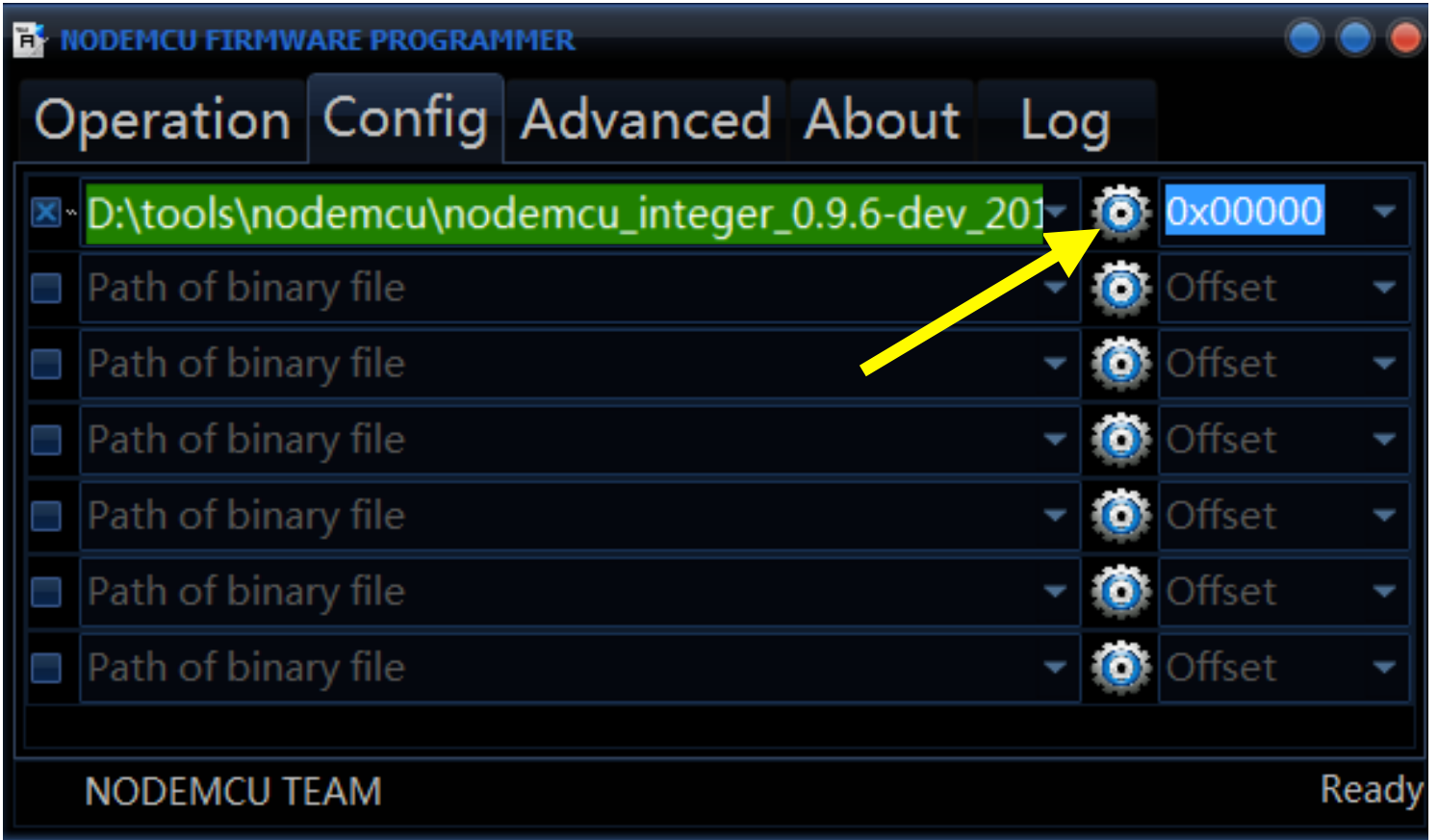
• 40MHz = Flash speed

• QIO = SPI Mode

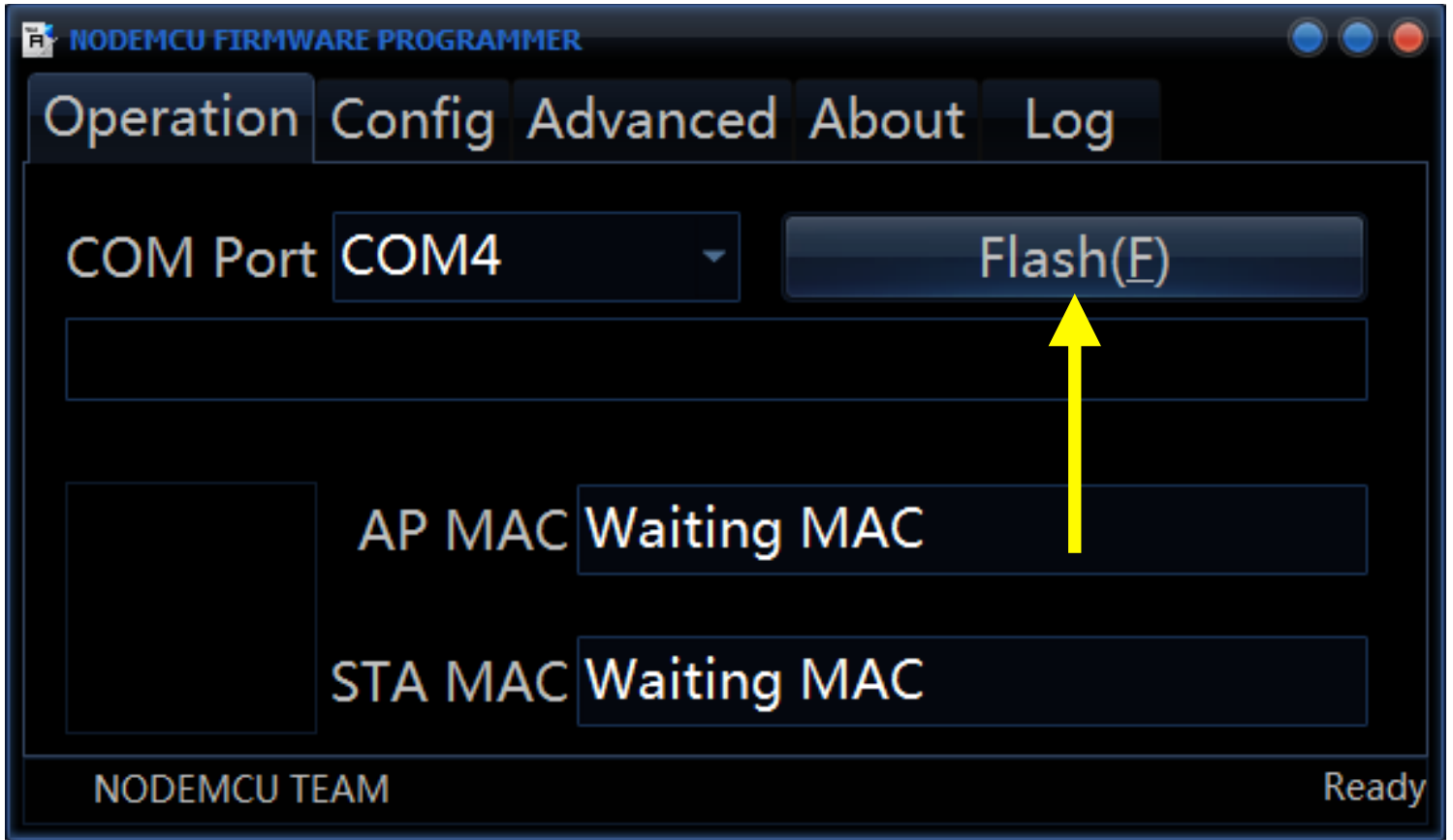
ثم بعد ذلك سنضغط على خانة (**Config**) وستظهر لنا الشاشة التالية:



NodeMCU



من هذه الصفحة سنقوم بإختيار ملف النظام NodeMCU الذي قمنا بتحميله سابقا . ثم بعد ذلك سنضغط على خانة (**Operation**) وستظهر لنا الشاشة التالية:



من هذه الصفحة سنضغط على زر **Flash** وسيبدأ تثبيت النظام كما هو موضح في الصورة التالية:



NodeMCU





NODEMCU FIRMWARE PROGRAMMER

Operation Config Advanced About Log

COM Port **COM4** Stop(S)

connect.world()


 AP MAC 1A.....3
STA MAC 1E.....


 NODEMCU TEAM Address:0x00000 Size:450072Byte

NODEMCU FIRMWARE PROGRAMMER

Operation Config Advanced About Log

COM Port **COM4** Flash(F)

 AP MAC 1A.....3
STA MAC 1E.....

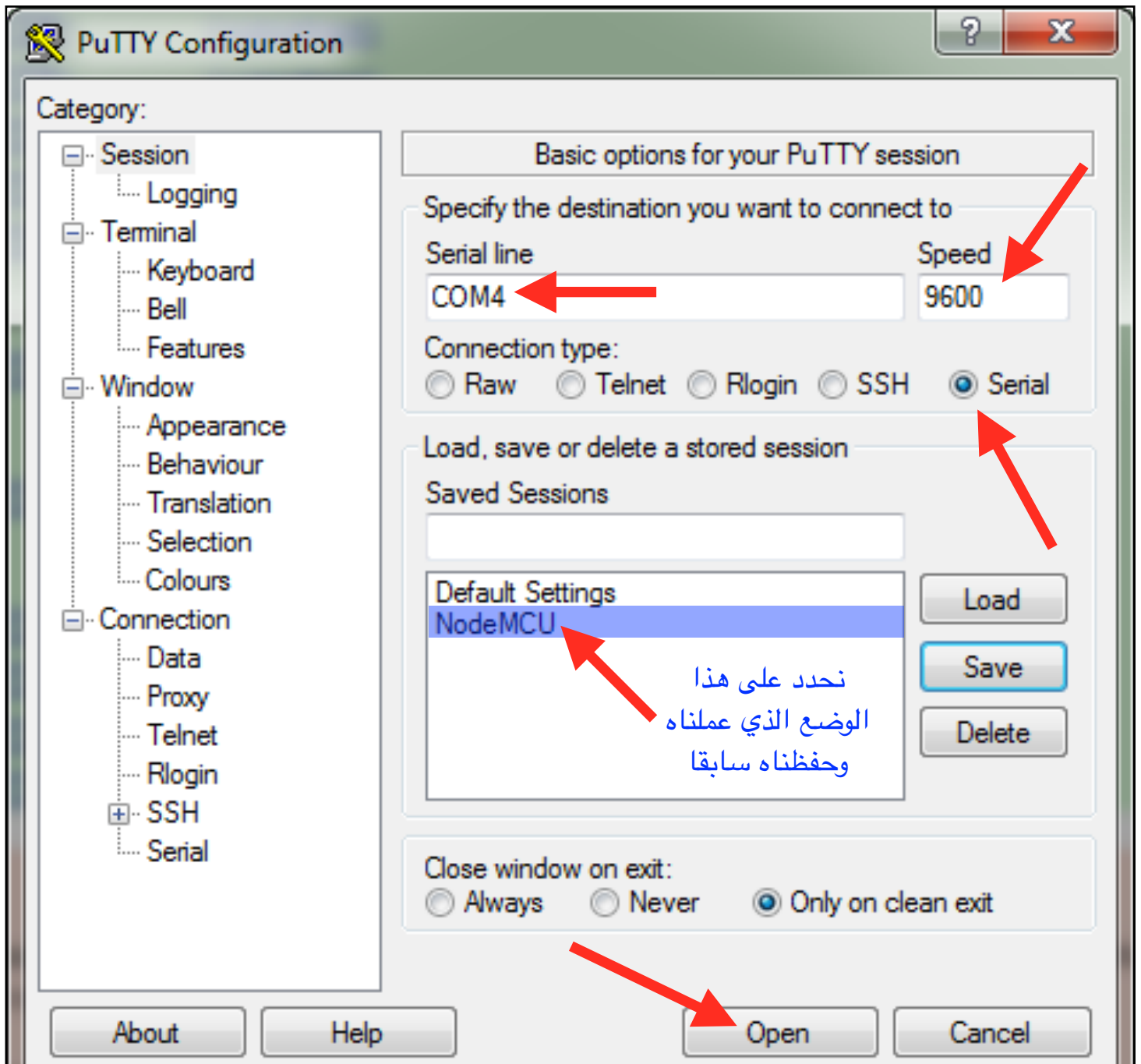
 NODEMCU TEAM Ready



4) التأكد من صحة تثبيت نظام NodeMCU:

سنقوم في هذه الخطوة بالتأكد من أن النظام تم تثبيته بالشكل الصحيح. سوف نفتح برنامج Putty وبعدها سنحدد الوضع الذي حفظناه سابقا بإسم

NodeMCU وثم سنضغط على زر **open**





NodeMCU



وبعد الضغط على زر **Open** ستظهر لنا الشاشة التالية:

```
COM4 - PuTTY
Restore init data.

NodeMCU 0.9.6 build 20150704 powered by Lua 5.1.4
lua: cannot open init.lua
>
```

في حال ظهرت لنا الشاشة خالية أو بها كلمات غير مفهومة فسنحتاج للضغط على مفتاح (user او rst) الموجود في لوحة NodeMCU وبعدها ستظهر لنا الشاشة كما هي موجودة في الصورة أعلاه.

بعد ذلك سنقوم بكتابة الامر التالي ثم الضغط على زر **Enter**:

```
print ("Hello NodeMCU")
```

وسنلاحظ بأن عبارة " Hello NodeMCU " قد طبعت على الشاشة كما هو موضح في الصورة التالية:



NodeMCU



```
COM4 - PuTTY
NodeMCU 0.9.6 build 20150704 powered by Lua 5.1.4
lua: cannot open init.lua
> print("Hello NodeMCU")
Hello NodeMCU
> █
```

بذلك نكون قد انهينا عملية تثبيت نظام NodeMCU ونستطيع بعد ذلك إقفال برنامج Putty وفصل لوحة NodeMCU من الحاسب.



البرنامج الخامس

البيئة التطويرية



الفصل الأول

برنامج *ESPlorer*



البيئة التطويرية هي البرنامج الذي سنكتب فيه الاسطر والاوامر البرمجية للتحكم بلوحة NodeMCU.

وفي هذا الفصل سنقوم بتحميل وتثبيت برنامج ESPlorer من خلال الخطوات التالية:

1) تحميل برنامج ESPlorer:

<http://esp8266.ru/esplorer/>

بعد الدخول على الرابط سنجد عبارة Download مكتوبة بخط كبير وسنضغط على هذه العبارة وبعدها سيبدأ التحميل كما هو موضح في الصورة التالية:



ملاحظة: برنامج ESPlorer يعمل على جميع أنظمة التشغيل



(2) تحميل الجافا (Java):

<http://java.com/en/download/>

بعد الدخول على الرابط سنضغط على Free Java Download وسيبدأ التحميل كما هو موضح في الصورة التالي:

Free Java Download

Download Java for your desktop computer now!

Version 8 Update 73

Release Date February 5, 2016

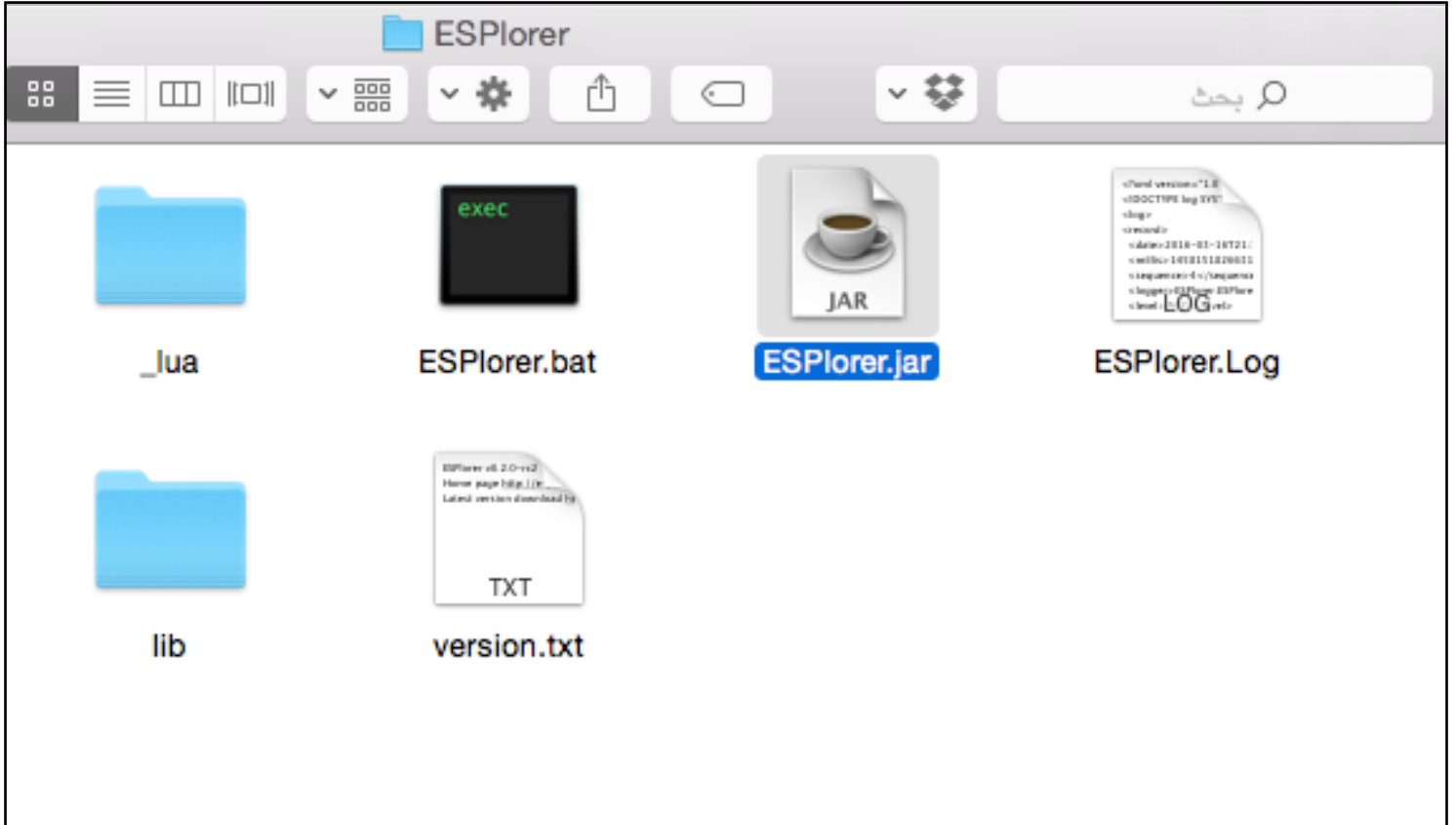
Free Java Download

وبعدها سنقوم بتثبيت البرنامج بطريقة تقليدية



(3) تشغيل برنامج ESPlorer:

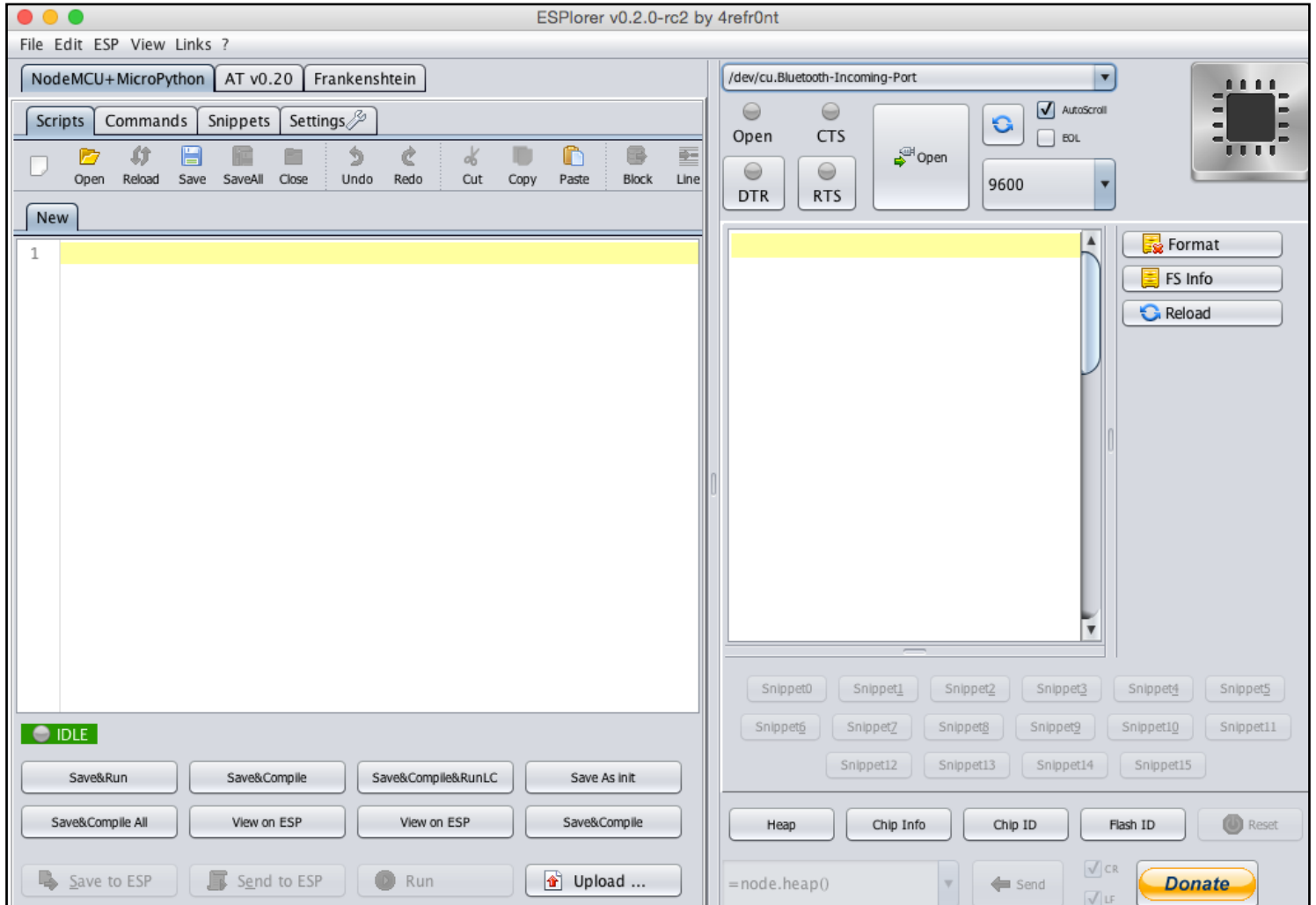
بعد الدخول على مجلد البرنامج سنجد عدة ملفات وسنضغط على الملف ESPlorer.jar كما هو موضح في الصورة التالي:



وبعد ذلك ستظهر لنا صفحة واجهة البرنامج كما هي موضحة في الصورة التالية:



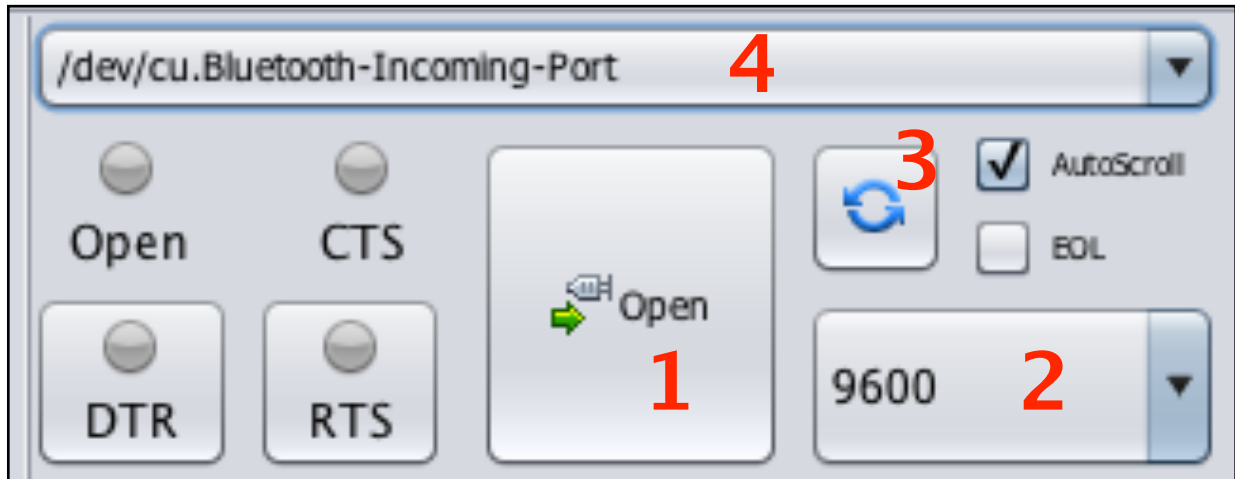
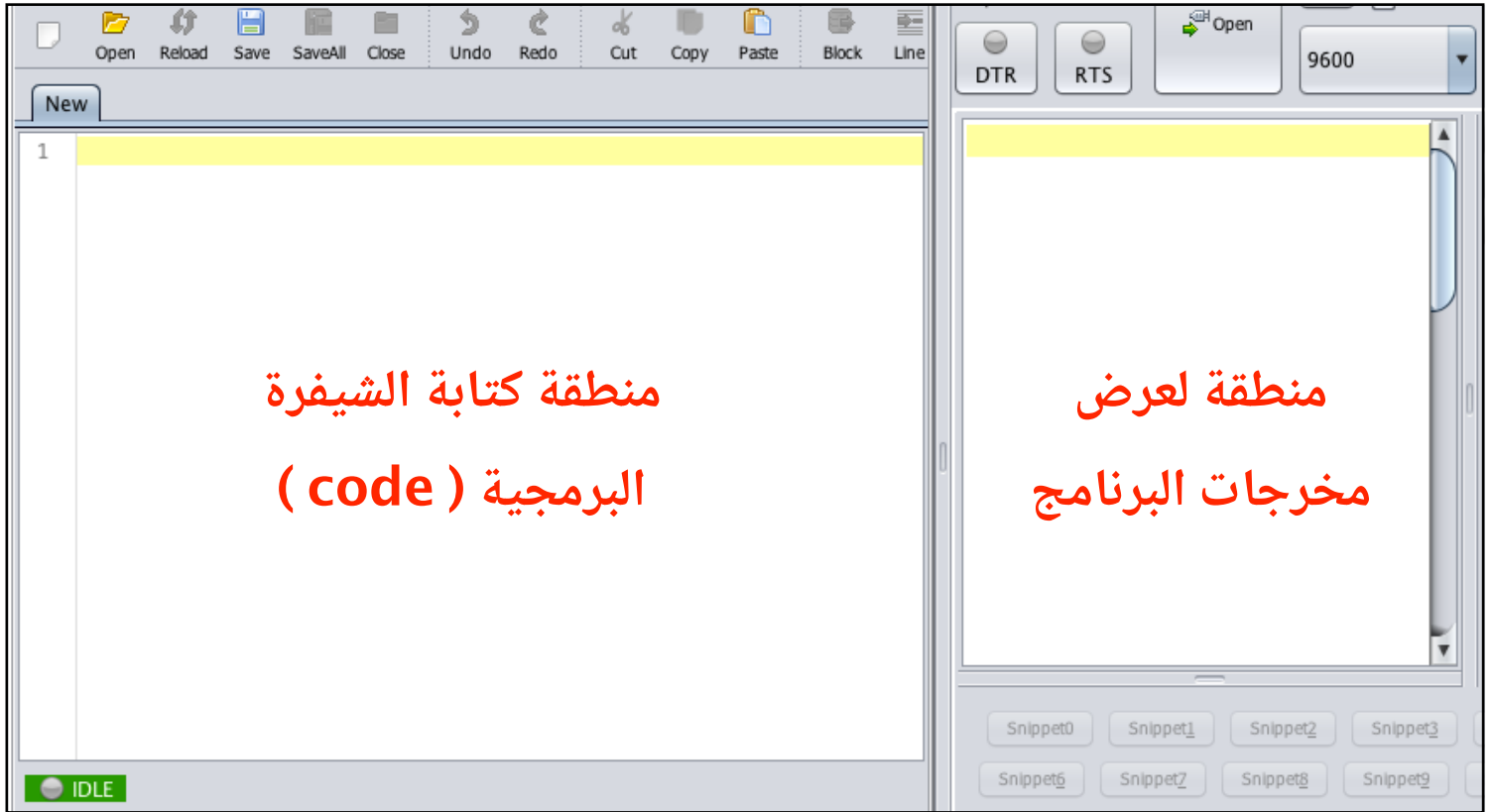
NodeMCU



شرح واجهة البرنامج موضح في الصور التالية:



NodeMCU



- 1) لتفعيل و قطع الاتصال بين الحاسب و لوحة NodeMCU.
- 2) لتحديد سرعة نقل البيانات بين الحاسب و لوحة NodeMCU.
- 3) لتحديث أو لإعادة البحث عن منافذ USB المستخدمة.
- 4) لإختيار اسم المنفذ المتصل بلوحة NodeMCU.



4) اللغة البرمجية المستخدمة في برنامج ESPlorer:

سنستخدم لغة لوا (Lua) لبرمجة لوحة
.NodeMCU

وهذا رابط الصفحة الرسمية للغة لوا
(Lua):

<http://www.lua.org/home.html>

وأيضاً، رابط آخر يحتوي على مجموعة كبيرة من
الأوامر البرمجية بلغة لوا (Lua) مع الأمثلة:

<http://www.nodemcu.com/docs/index>





الفصل الثلثي

برنامج Arduino



NodeMCU



البيئة التطويرية هي البرنامج الذي سنكتب فيه الاسطر والوامر البرمجية للتحكم بلوحة NodeMCU.

وفي هذا الفصل سنقوم بتحميل وتثبيت برنامج الأردوينو (Arduino IDE) وإضافة حزمة التشغيل الخاصة بـ شريحة ESP8266 من خلال الخطوات التالية:

1) تحميل برنامج الأردوينو:

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

بعد الدخول على الرابط سنختار نظام التشغيل الذي نعمل عليه وبعدها سننتقل مباشرة على صفحة التحميل وسنضغط على just download

TIMES. IMPRESSIVE! THIS IDE IS NO LONGER JUST FOR ARDUINO AND GENUINO BOARDS. HUNDREDS OF COMPANIES AROUND THE WORLD ARE USING IT TO PROGRAM THEIR DEVICES, INCLUDING COMPATIBLES, CLONES, AND EVEN COUNTERFEIT. YOU CAN HELP ACCELERATE THE DEVELOPMENT OF THE ARDUINO IDE BY CONTRIBUTING TOWARDS THE EFFORT OF MAKING IT BETTER.

\$5 \$10 \$25 \$50 OTHER

JUST DOWNLOAD CONTRIBUTE & DOWNLOAD



2) اضافة حزمة ESP8266 لبرنامج الاردوينو:

سنفتح برنامج الاردوينو وسنتقل الى المسار التالي:

للماك : Arduino ← Preferences (التفضيلات)

للويندوز : File (ملف) ← Preferences (التفضيلات)

للينكس : File (ملف) ← Preferences (التفضيلات)

وستظهر لنا صفحة التفضيلات كما هو موضح في الصورة التالية:

مكان كتاب الشيفرة البرمجية "سكتش بوك"
/Users/inventor/Documents/Arduino استعرض

لغة المحرر: الاعدادات الافتراضية (يتطلب اعادة تشغيل للاردوينو)

حجم خط المحرر: 15 (يتطلب اعادة تشغيل للاردوينو)

رفع ترجمة عرض المخرجات خلال:

Compiler warnings: None

عرض أرقام السطور
 التأكد من الكود بعد الرفع
 استعمال محرر خارجي
 افحص التحديثات عند التشغيل
 تحديث ملفات الشيفرة البرمجية الى امتداد جديد عند الحفظ (.ino. -> pde)
 Save when verifying or uploading

Proxy Settings

Server (HTTP): Port (HTTP): 8080

Server: (HTTPS) Port (HTTPS): 8443

Username: كلمة المرور

Additional Boards Manager URLs:

يمكن تعديل خصائص اكثر في الملف يتشكل مباشر
/Users/inventor/Library/Arduino15/preferences.txt
(لا يمكن التحرير والاردوينو تعمل)

موافق الغاء



بعد ذلك سنتجه لخانة (additional boards manager urls) المشار إليها في الصورة السابقة وسنضع في هذه الخانة الرابط التالي:

http://arduino.esp8266.com/versions/2.1.0/package_esp8266com_index.json

وبعد ذلك سنضغط على موافق كما هو موضح في الصورة التالية:

مكان كتاب الشيفرة البرمجية "سكتش بوك"

/Users/inventor/Documents/Arduino استعرض

لغة المحرر: (يتطلب اعادة تشغيل للأردوينو)

حجم خط المحرر: (يتطلب اعادة تشغيل للأردوينو)

رفع ترجمة عرض المخرجات خلال:

Compiler warnings:

عرض أرقام السطور

التأكد من الكود بعد الرفع

استعمال محرر خارجي

افحص التحديثات عند التشغيل

تحديث ملفات الشيفرة البرمجية الى امتداد جديد عند الحفظ (.ino -> .pde)

Save when verifying or uploading

Proxy Settings

Server (HTTP): Port (HTTP):

Server: (HTTPS) Port (HTTPS):

Username: كلمة المرور

Additional Boards Manager URLs:

يمكن تعديل خصائص أكثر في الملف بشكل مباشر
/Users/inventor/Library/Arduino15/preferences.txt
(لا يمكن التحرير والأردوينو تعمل)

موافق الغاء



NodeMCU

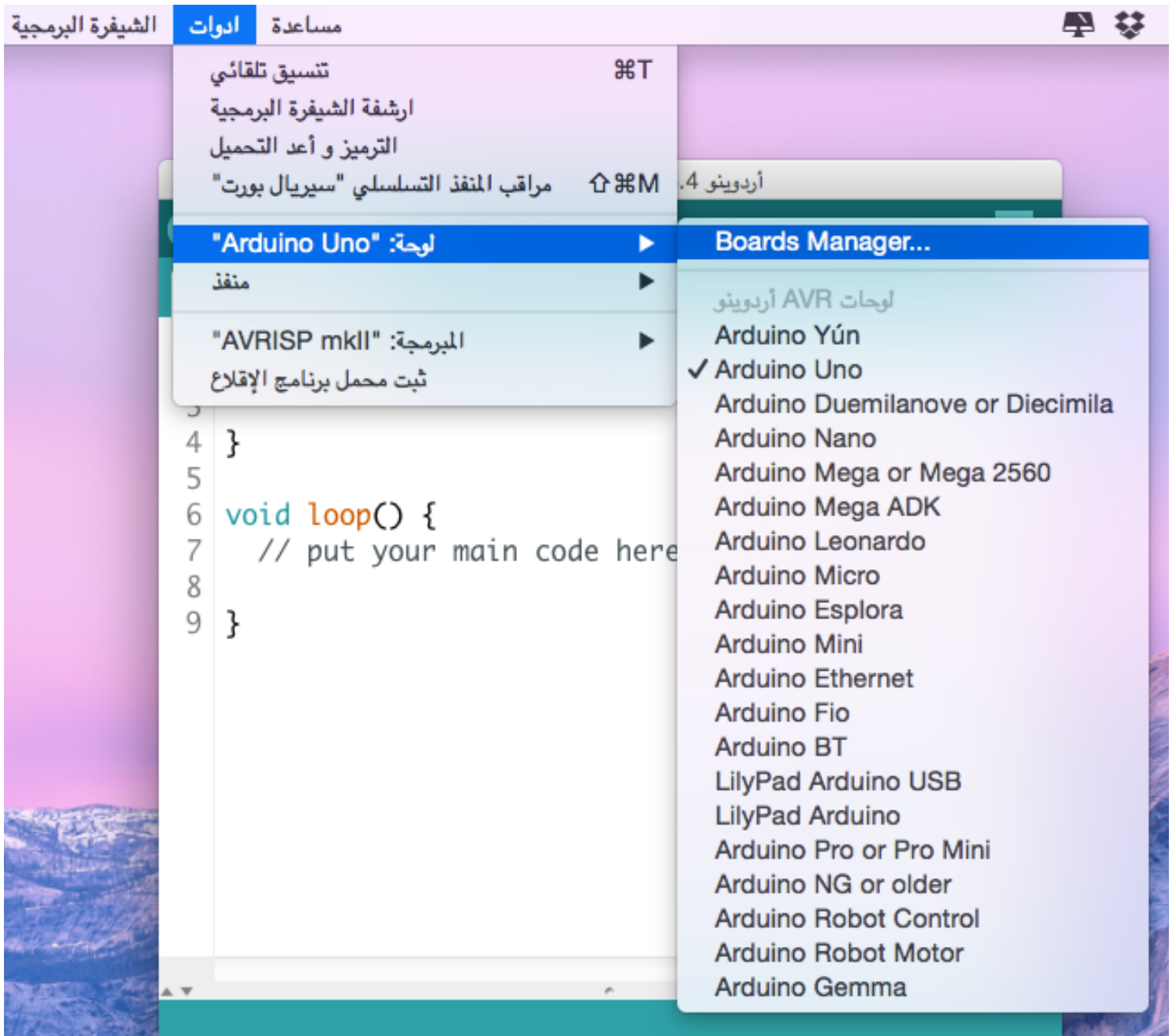


بعد ذلك سننتقل الى المسار التالي:

أدوات ← لوحة ← إدارة اللوحات

Boards Manager ← Board ← Tools

كما هو موضح في الصورة التالية:

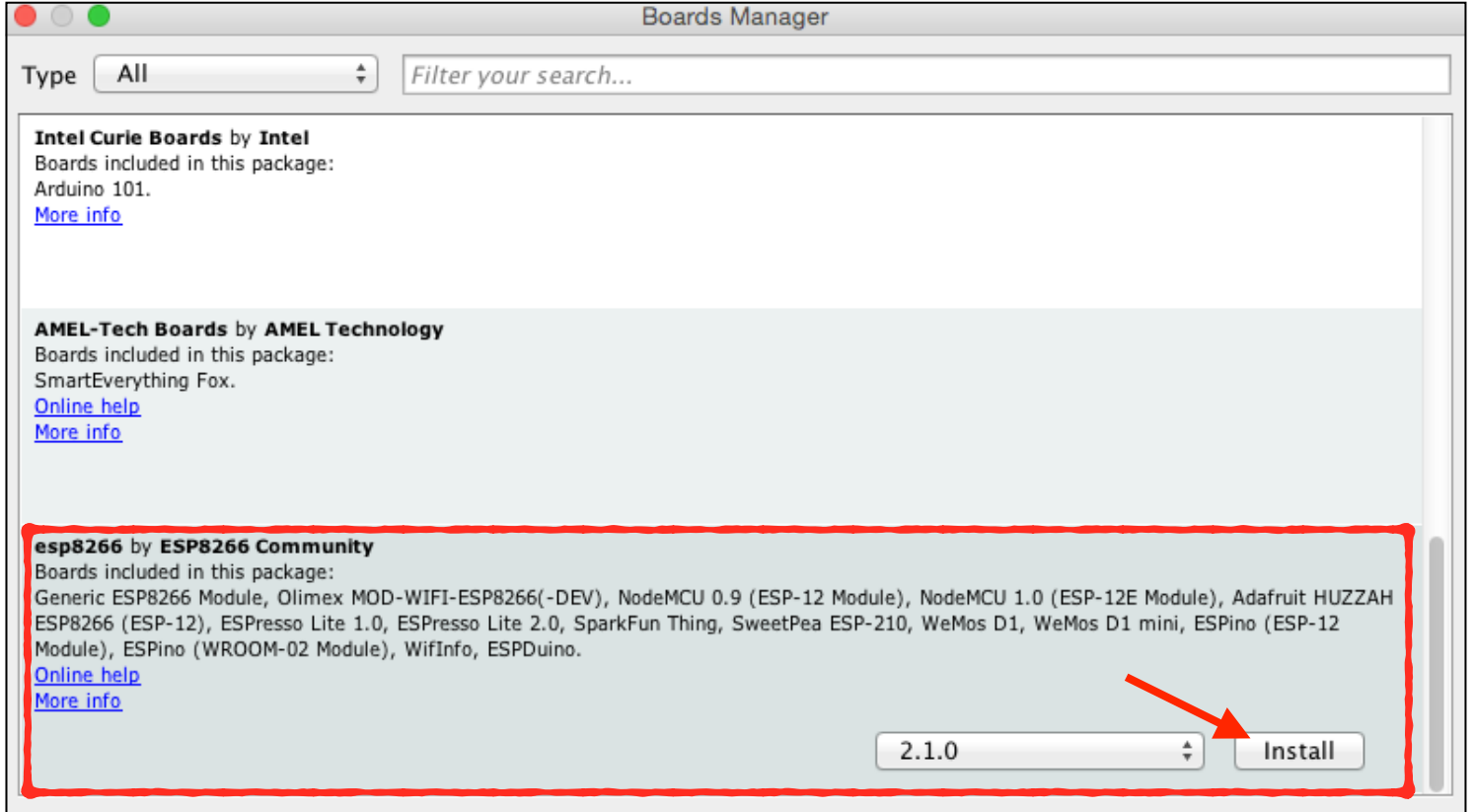




NodeMCU



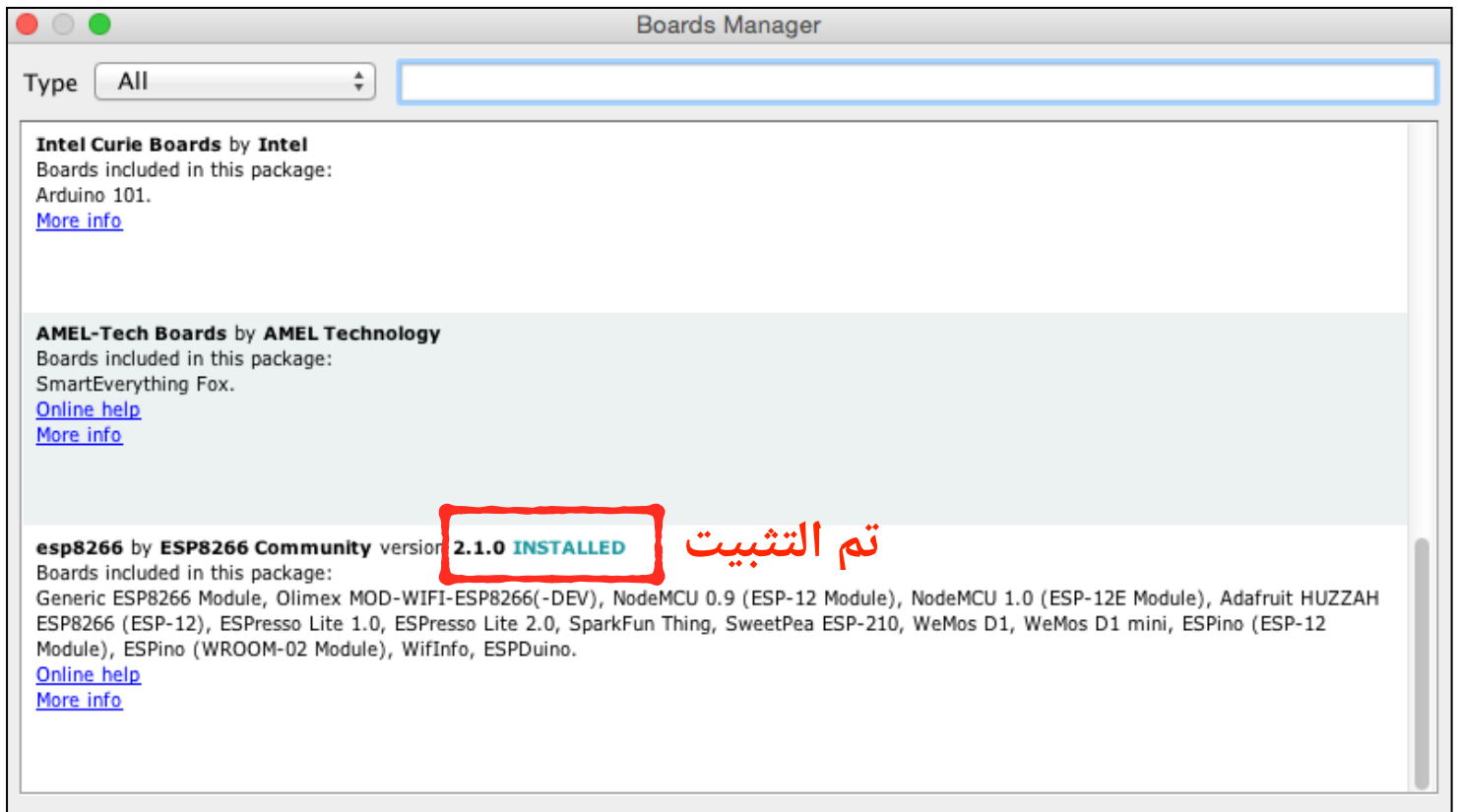
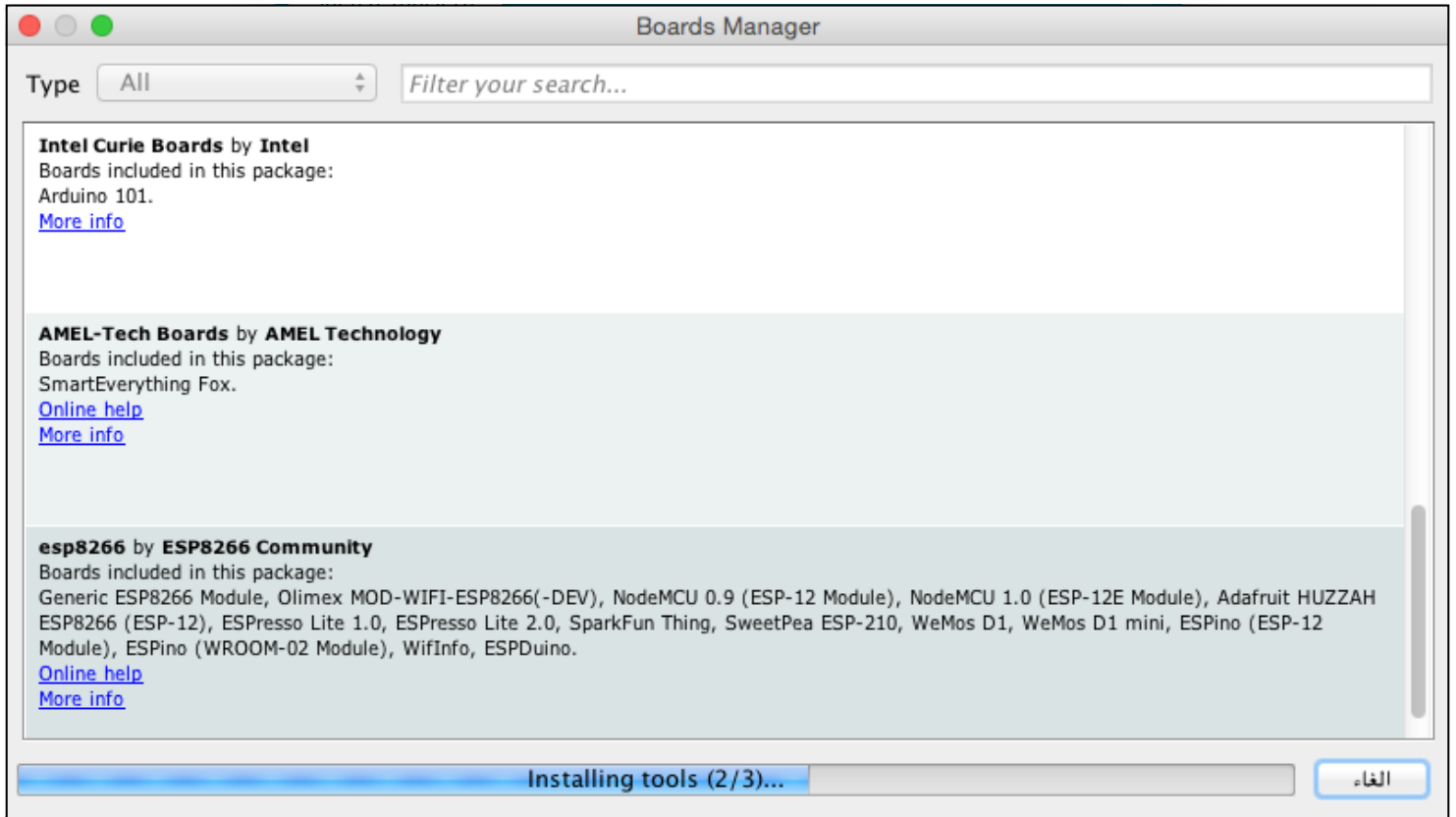
ثم بعد الضغط على Boards Manager (إدارة اللوحات) ستظهر لنا الصفحة التالية:



سنلاحظ من الصورة السابقة وجود العديد من الحزم. وسنبحث عن حزمة esp8266 ونحدد عليها ثم نضغط على زر **Install** وسيبدأ تثبيت الحزمة كما هو موضح في الصورة التالية:



NodeMCU





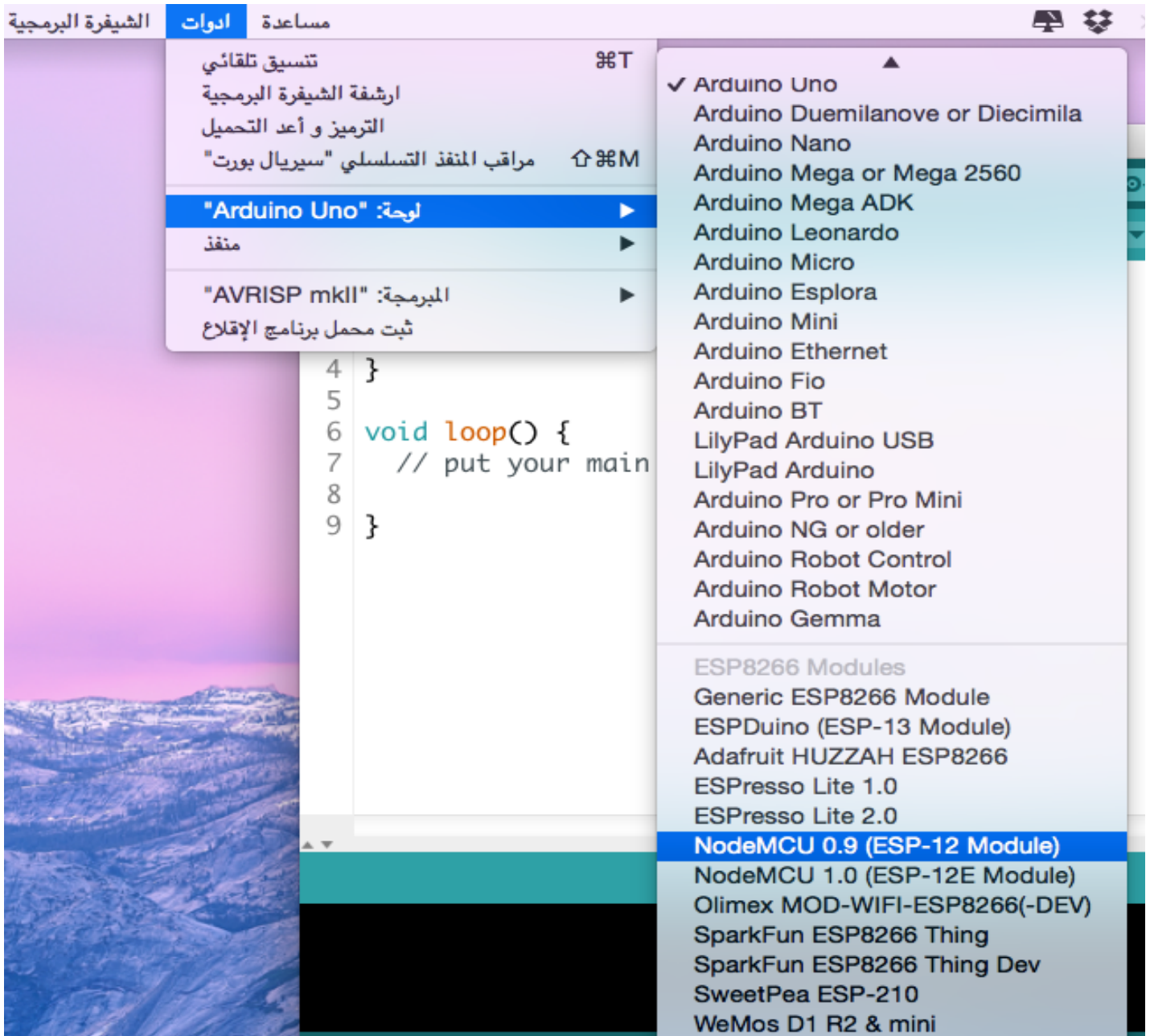
NodeMCU



بعد ذلك سننتقل الى المسار التالي:

أدوات (Tools) ← لوحة (Board)

وسنلاحظ وجود لوحات إضافية ومن ضمنها لوحة NodeMCU بإصداريها الاصدار (0.9) والاصدار (1.0) وسنختار الاصدار المطلوب كما هو موضح في الصورة التالية:





البداية السريعة

الطريق إلى التحكم



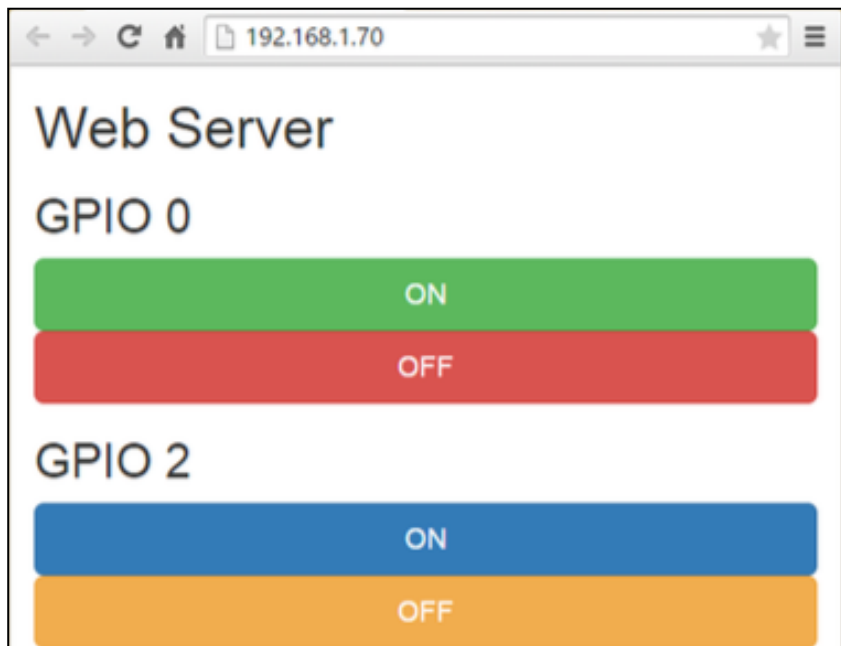
مقدمة:

في هذا الباب سنقوم بعمل مجموعة من مشاريع التحكم التي تعتمد على الاتصال اللاسلكي باستخدام تقنية الواي فاي (wifi).

وستتحكم بهذه المشاريع لاسلكيا عن طريق استخدام الهواتف الذكية او الاجهزة اللوحية.

وبما أننا سنستخدم تقنية الواي فاي (wifi)، فهذا يعني أنه يمكننا التحكم بمشروعنا بأحد طريقتين:

الطريقة الأولى هو أن نقوم بعمل تطبيق (Application) لنتحكم من خلاله بالمشروع أو استخدام تطبيق جاهز.



الطريقة الثانية وهو ان نقوم بعمل صفحة الكترونية او موقع الكتروني لنتحكم من خلاله بالمشروع.

وفي هذا الكتاب سنتعلم طريقة انشاء صفحة إلكترونية تسمى بخادم الشبكة (web server)

وتحتوي هذه الصفحة على واجهة تمكننا من التحكم بمشروعنا.



الفصل الأول

صفحة خازم الشبكة



إنشاء صفحة شبكة الخادم:

لإنشاء صفحة خادم الشبكة نحن نحتاج الى تعلم عدة لغات برمجية مثل (HTML، CSS، JavaScript)، ولكن في هذا الكتاب سنختصر هذه المسافة ونتعلم استخدام البوتستراب (Bootstrap).

ماهو البوتستراب:



هو إطار عمل (framework) بلغة HTML و CSS و JavaScript. أي أنه بدل من تعلم ثلاث لغات لإنشاء صفحة شبكة الخادم سنحتاج فقط لتعلم إطار عمل واحد أو لغة واحدة تجمع بين هذه الثلاث لغات.

ولتعلم طريقة استخدام البوتستراب أنصح باستخدام الموقع التالي:

<http://www.w3schools.com/bootstrap/default.asp>

من خلال هذا الموقع نستطيع التعلم والتطبيق ومشاهدة النتائج. سأشرح في الصفحات التالية طريقة استخدام الموقع.



طريقة استخدام الموقع:

<http://www.w3schools.com/bootstrap/default.asp>

بعد الدخول على الرابط ستظهر لنا الصفحة الرئيسية كما هو موضح في الصورة التالية:

Bootstrap Tutorial

BS HOME

BS Get Started

BS Grid Basic

BS Typography هذه قائمة بأسماء الدروس الخاصة بالبوستتراب

BS Tables

BS Images

BS Jumbotron

BS Wells

BS Alerts

BS Buttons

BS Button Groups

BS Glyphicons

BS Badges/Labels

BS Progress Bars

BS Pagination

BS Pager

BS List Groups

BS Panels

BS Dropdowns

Bootstrap 3 Tutorial

« W3Schools Home

B

Bootstrap is the most popular HTML responsive, mobile-first web sites.

Bootstrap is completely free to dow

Start learning Bootstrap 3 now!

Try it Yourself Examples

This Bootstrap tutorial contains hundreds of Bootstrap ex

With our online editor, you can edit the code, and click on



جميع الدروس تعمل بنفس الطريقة لذلك سيكون الشرح على درس واحد فقط.

فعلى سبيل المثال سنختار الدرس (BS Buttons) ومن خلاله سنتعلم طريقة انشاء الازارير (المفاتيح) وستظهر لنا الصفحة التالية:

Button Styles

Bootstrap provides seven styles of buttons: اشكال الازارير وألوانها



To achieve the button styles above, Bootstrap has the following classes:

- `.btn-default`
- `.btn-primary`
- `.btn-success`
- `.btn-info`
- `.btn-warning`
- `.btn-danger`
- `.btn-link`

صيغة كتابة أمر
تنفيذ لون معين

The following example shows the code for the different button styles:

بعد ذلك سنذهب الى المثال للتدرب على الشيفرة البرمجية الخاصة بإنشاء الازارير (المفاتيح) كما هو موضح في الصورة التالية:

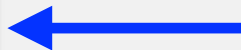


The following example shows the code for the different button styles:

Example

```
<button type="button" class="btn btn-default">Default</button>  
<button type="button" class="btn btn-primary">Primary</button>  
<button type="button" class="btn btn-success">Success</button>  
<button type="button" class="btn btn-info">Info</button>  
<button type="button" class="btn btn-warning">Warning</button>  
<button type="button" class="btn btn-danger">Danger</button>  
<button type="button" class="btn btn-link">Link</button>
```

Try it Yourself »



بعد أن نضغط على زر **Try it Yourself** ستظهر لنا صفحة التطبيق
والتعديل على الشيفرة البرمجية ومشاهدة النتائج كما هو موضح في الصورة
التالية:



Edit This Code:

لمشاهدة النتائج

See Result »

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Bootstrap Example</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.0/jquery.min.js"></script>
  <script src="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js"></script>
</head>
<body>
```

نستطيع التعديل والاضافة

في هذه الخانة

Result:

Button Styles

Default

Primary

Success

Info

Warning

Danger

Link

نتائج الشيفرة البرمجية

ستظهر في هذه الخانة

فلو أردنا على سبيل المثال حذف الزر الأحمر و الزر الأصفر فإننا سنتوجه الى خانة الشيفرة البرمجية و سنقوم بتعديل بسيط في الشيفرة البرمجية. وبعد الانتهاء من التعديل سنضغط على زر **See Result** وستظهر لنا النتائج من دون الزر الأحمر و الأصفر كما هو موضح في الصور التالية:



Edit This Code:

```
<div class="container">
  <h2>Button Styles</h2>
  <button type="button" class="btn btn-default">Default</button>
  <button type="button" class="btn btn-primary">Primary</button>
  <button type="button" class="btn btn-success">Success</button>
  <button type="button" class="btn btn-info">Info</button>
  <button type="button" class="btn btn-warning">Warning</button>
  <button type="button" class="btn btn-danger">Danger</button>
  <button type="button" class="btn btn-link">Link</button>
</div>
```

سنقوم بحذف السطرين الموضحين في الصورة أعلاه. وبعد ذلك سنضغط على زر **See Result** وستظهر لنا النتائج كما هو موضح في الصورة التالية:

Edit This Code:

```
<div class="container">
  <h2>Button Styles</h2>
  <button type="button" class="btn btn-default">Default</button>
  <button type="button" class="btn btn-primary">Primary</button>
  <button type="button" class="btn btn-success">Success</button>
  <button type="button" class="btn btn-info">Info</button>
  <button type="button" class="btn btn-link">Link</button>
</div>
```

Result:

Button Styles

Default

Primary

Success

Info

Link



ملاحظة حول الشيفرة البرمجية الخاصة بالبوستراتاب:

نلاحظ من الصورة التالية أن الجزئية المحددة من الشيفرة البرمجية ستتكرر معنا في جميع الأمثلة تقريبا من دون تغيير.

Edit This Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Bootstrap Example</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-s
  <link rel="stylesheet" href="http://maxcdn.bootstrapcdn.com/
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.
  <script src="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/
</head>
<body>
```



الفصل الثلثي

المشاريع

(برنامج *ESPlorer*)



المشروع الأول (الفلاش) :

فكرة المشروع:

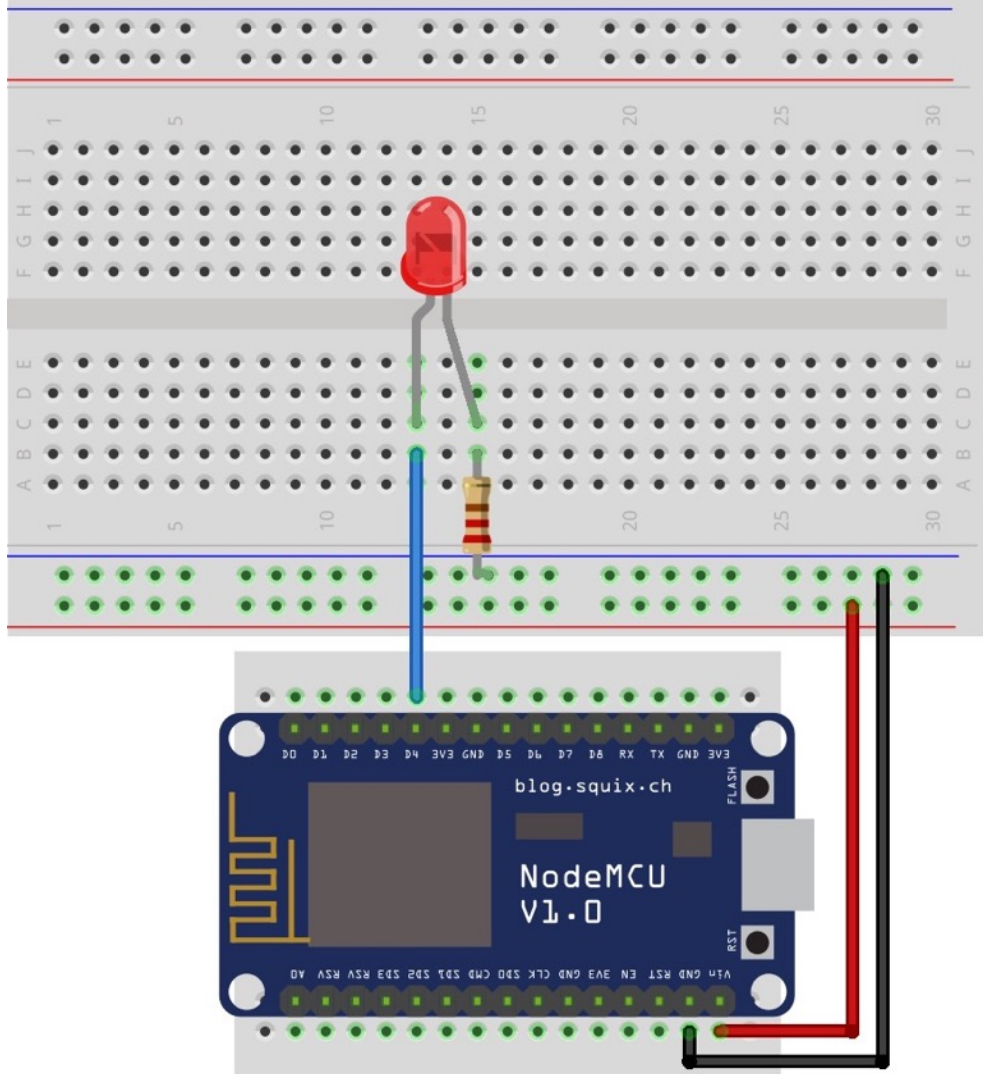
سنقوم بعمل فلاش (تشغيل و إطفاء) للدايود الضوئي (LED). وسنركز في هذا المشروع على شرح الاوامر البرمجية.

الأدوات المستخدمة:

- لوحة NodeMCU
- حامل لوحة NodeMCU
- لوحة تثبيت القطع الالكترونية (Breadboard)
- مقاومة 220 اوم
- دايود ضوئي (LED)



الدائرة الالكترونية:



قمنا بتوصيل الطرف الموجب للديود الضوئي (LED) في المدخل D4 من لوحة NodeMCU والطرف السالب للديود الضوئي (LED) متصل بالارضي (GND) من خلال المقاومة 220.



كتابة الشيفرة البرمجية وشرحها:

الشيفرة البرمجية لهذا المشروع موجودة على الرابط التالي:

www.github.com/flash.lua

بعد ذلك سنقوم بفتح برنامج ESPlorer لكتابة الشيفرة البرمجية.

```
New
1 led=4
2 x=0
3 gpio.mode(led,gpio.OUTPUT)
4 tmr.alarm(0,1000,1,function()
5   if x==0 then
6     x=1
7     gpio.write(led,gpio.HIGH)
8   else
9     x=0
10    gpio.write(led,gpio.LOW)
11  end
12 end)
13
```

● IDLE

في الصفحات التالية سنقوم بشرح هذه الاوامر البرمجية. وبعدها سنقوم بعملية رفع الشيفرة البرمجية على لوحة NodeMCU.



طريقة الشرح ستكون عبارة عن جدول يحتوي على عمودين:

العمود الأول: أرقام سطور الاوامر البرمجية.

العمود الثاني: شرح الاوامر البرمجية.

```
1 led=4
2 x=0
3 gpio.mode(led,gpio.OUTPUT)
```

السطر	الشرح
1	قمنا بتعريف متغير بإسم led وهو يشير الى المدخل D4
2	قمنا بتعريف متغير بإسم x وحددناه بقيمة افتراضية وهي 0
3	تهيئة الدايمود الضوئي كخرج (output)



```
4 tmr.alarm(0,1000,1,function()  
5 if x==0 then  
6     gpio.write(led,gpio.HIGH)  
7     x=1  
8 else  
9     gpio.write(led,gpio.LOW)  
10    x=0  
11 end  
12 end)
```

السطر	الشرح
4	دالة مؤقتة تقوم بتنفيذ أمر معين كل فترة زمنية محددة بوحدة ملي ثانية. وهذه الفترة محددة بـ 1000 ملي ثانية (1 ثانية).
5	هذا هو الامر الذي سينفذ داخل الدالة المؤقتة وهو عبارة عن دالة شرطية ستختبر هل قيمة المتغير x تساوي 0 ام لا. فإذا كانت تساوي 0 فسيتم تشغيل الدايود الضوئي. وأما إن كانت تساوي 1 فسيتم اطفاء الدايود الضوئي. وتكرر هذه العملية كل 1000 ملي ثانية
11	نهاية الدالة الشرطية if
12	نهاية الدالة المؤقتة (tmr.alarm)



في هذه الصفحة والتي تليها سأشرح الدالة المؤقتة (tmr.alarm) بشكل مفصل:

```
tmr.alarm( 0, 1000, 1, function()  
    print( "hello world" )  
end )
```

نلاحظ أن دالة (tmr.alarm) تحتوي على 4 عناصر:

العنصر الأول (0) : يشير الى الرقم الخاص للدالة ويأخذ القيم من 0 الى 6 ولا يشترط الترتيب.

العنصر الثاني (1000) : الفترة الزمنية بوحدة ملي ثانية.

العنصر الثالث (1) : يأخذ قيمتين إما 1 أو 0. فإذا كانت قيمته ب 1 فهذا يعني أنه سيتم تنفيذ الامر كل ثانية (1000 ملي ثانية) باستمرار دون توقف. وأما إذا كانت قيمته ب 0 فهذا يعني أنه سيقوم بتنفيذ الأمر لمرة واحدة فقط بعد مرور ثانية واحدة دون تكرار

العنصر الرابع (function) : يشير الى الامر المراد تنفيذه. وفي هذا المثال فإن الأمر المراد تنفيذه هو طباعة العبارة (hello world).

المفهوم الكامل لهذا المثال: سيتم طباعة العبارة (hello world) كل ثانية واحدة باستمرار دون توقف.



قد تحتوي الشيفرة البرمجية على دوال متعددة من دالة (tmr.alarm)
وهنا تأتي فائدة الرقم الخاص بالدالة

```
tmr.alarm( 0, 2000, 1, function()  
    print( "green" )  
end )
```

```
tmr.alarm( 4, 5000 ,0, function()  
    print( "black" )  
end )
```

```
tmr.alarm( 2, 300, 1, function()  
    print( "white" )  
end )
```

دالة (tmr.alarm) الاولى: رقمها الخاص هو 0. وهي تعني أن سيتم طباعة العبارة (green) كل ثانيتين باستمرار دون توقف.

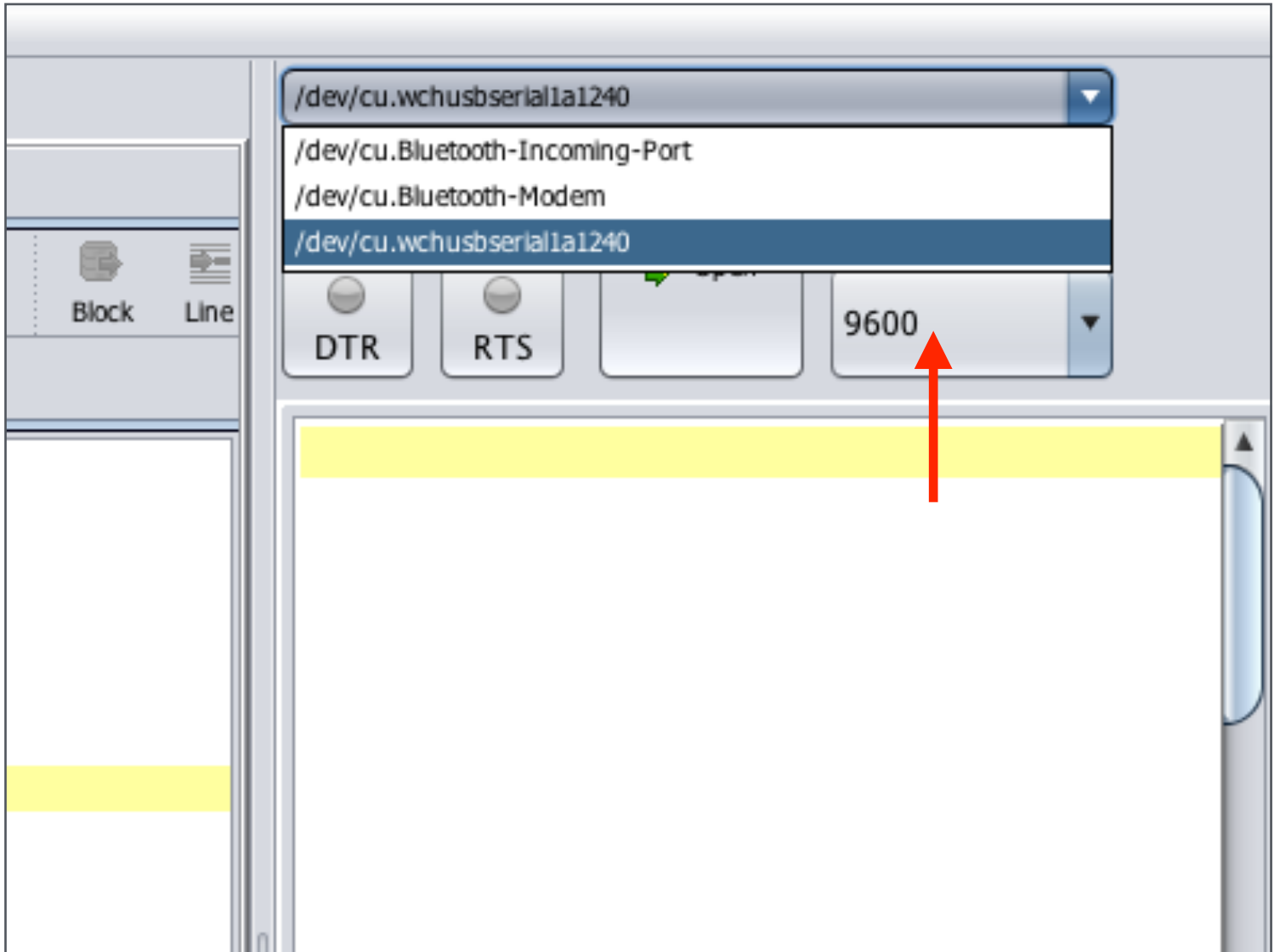
دالة (tmr.alarm) الثانية: رقمها الخاص هو 4. وهي تعني أن سيتم طباعة العبارة (black) بعد مرور 5 ثواني لمرة واحدة دون تكرار.

دالة (tmr.alarm) الثالثة: رقمها الخاص هو 2. وهي تعني أن سيتم طباعة العبارة (white) كل 300 ملي ثانية باستمرار دون توقف.



رفع البرنامج على لوحة NodeMCU وتشغيل المشروع:

بعد الانتهاء من كتابة الشيفرة البرمجية سنقوم بتوصيل لوحة NodeMCU بالحاسب. وبعدها سنقوم بإختيار المنفذ المتصل بلوحة NodeMCU. وأيضا سنختار 9600 كسرعة نقل البيانات كما هو موضح في الصورة التالية:



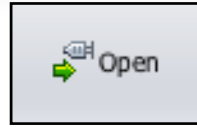
إن لم تجد المنفذ ضمن الخيارات فعليك بالضغط على زر التحديث وبعدها سيظهر معك المنفذ بإذن الله.



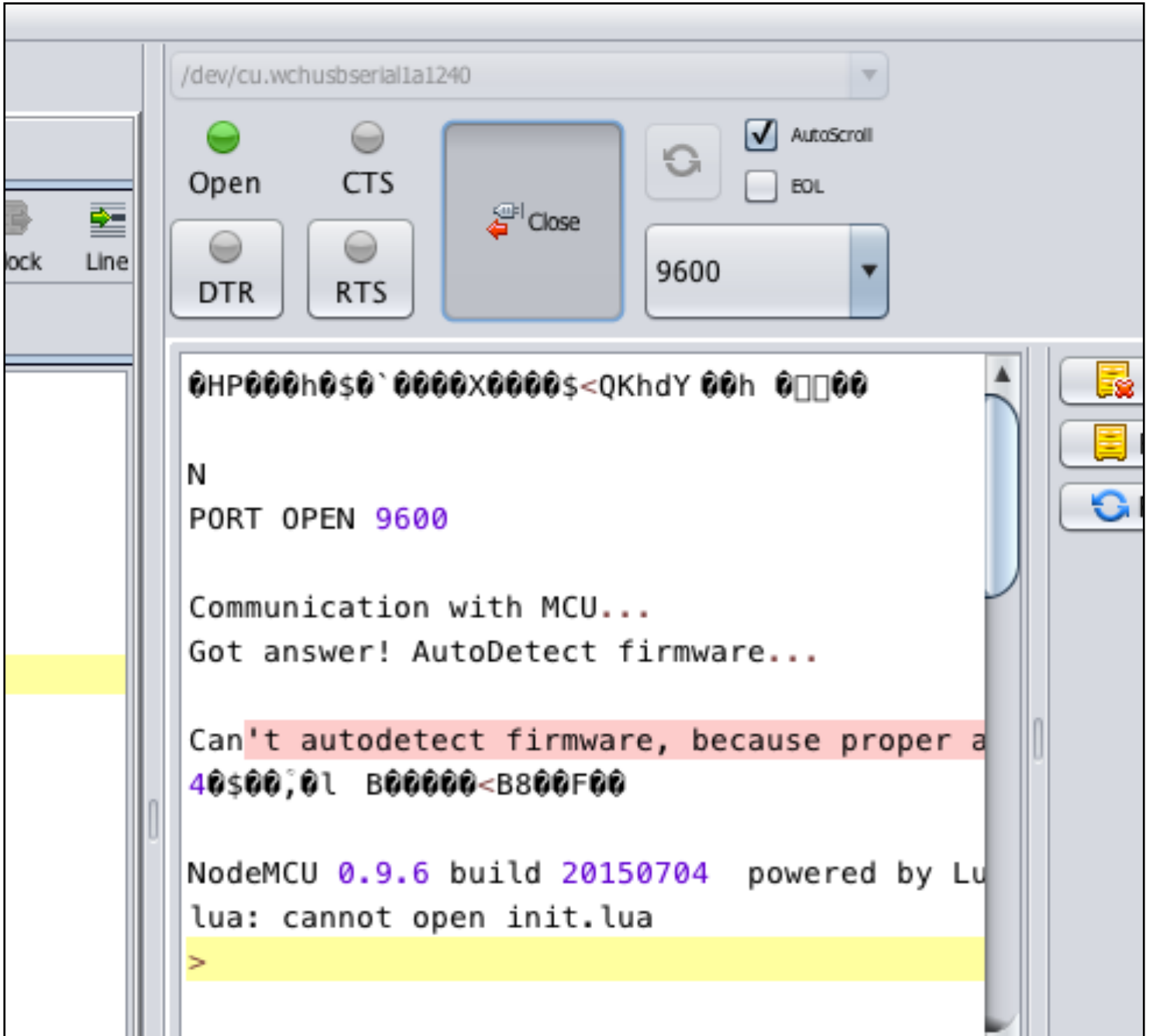
NodeMCU

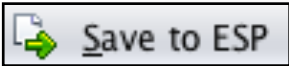


وسيداً عملية الاتصال كما



ثم سنضغط على زر تفعيل الاتصال
هو موضح في الصورة التالية:





بعد ذلك سنرفع البرنامج بالضغط على زر (save to ESP)



أو بالضغط على زر (send to ESP)

والفرق بين الاثنين هو:

• زر (save to ESP): يقوم بحفظ البرنامج داخل شريحة ESP8266 حتى عند فصل اللوحة من الحاسب وتشغيلها باستخدام مصدر خارجي للتغذية (بطاريات) فإن المشروع سيعمل لأن البرنامج محفوظ داخل شريحة ESP8266.

• زر (send to ESP): يقوم فقط بقراءة البرنامج دون حفظه ولا يمكن تشغيل المشروع عند فصل اللوحة عن الحاسب وإستخدام مصدر خارجي للتغذية (بطاريات) لأن البرنامج غير محفوظ داخل شريحة ESP8266.

سأقوم بالضغط على زر (save to ESP) و سيطلب منا أن نقوم بحفظ الشيفرة البرمجية بإسم وسوف أسميه بـ `init.lua` كما هو موضح في الصورة التالية:

File Name:

Files of Type:

Save Cancel



NodeMCU



بعد الانتهاء من حفظ الشيفرة بإسم سيتم رفع البرنامج كما هو موضح في الصورة التالية:

The screenshot shows the NodeMCU IDE interface. The main editor displays the 'init.lua' file with the following code:

```
1 led=4
2 x=0
3 gpio.mode(led,gpio.OUTPUT)
4 tmr.alarm(0,1000,1,function()
5 if x==0 then
6     x=1
7     gpio.write(led,gpio.HIGH)
8 else
9     x=0
10    gpio.write(led,gpio.LOW)
11 end
12 end
13
```

The terminal window on the right shows the upload process:

```
> file.remove("init.lua");
> file.open("init.lua","w+");
> w = file.writeline
> w([[led=4]]);
> w([[x=0]]);
> w([[gpio.mode(led,gpio.OUTPUT)]];
> w([[tmr.alarm(0,1000,1,function()]];
> w([[if x==0 then]];
> w([[     x=1]];
> w([[     gpio.write(led,gpio.HIGH)]];
> w([[else]];
> w([[     x=0]];
> w([[     gpio.write(led,gpio.LOW)]];
> w([[end]];
> w([[end]]);
```

The status bar at the bottom indicates 'BUSY' and a progress bar at 93%.

بعد انتهاء الرفع سنلاحظ أن الدايود الضوئي بدأ بعملية الفلاش كل ثانية

رابط فيديو لمشاهدة تشغيل المشروع:

<https://www.youtube.com/watch?v=FIecLg0plZM>



ملاحظة حول تسمية ملفات الشيفرة البرمجية:

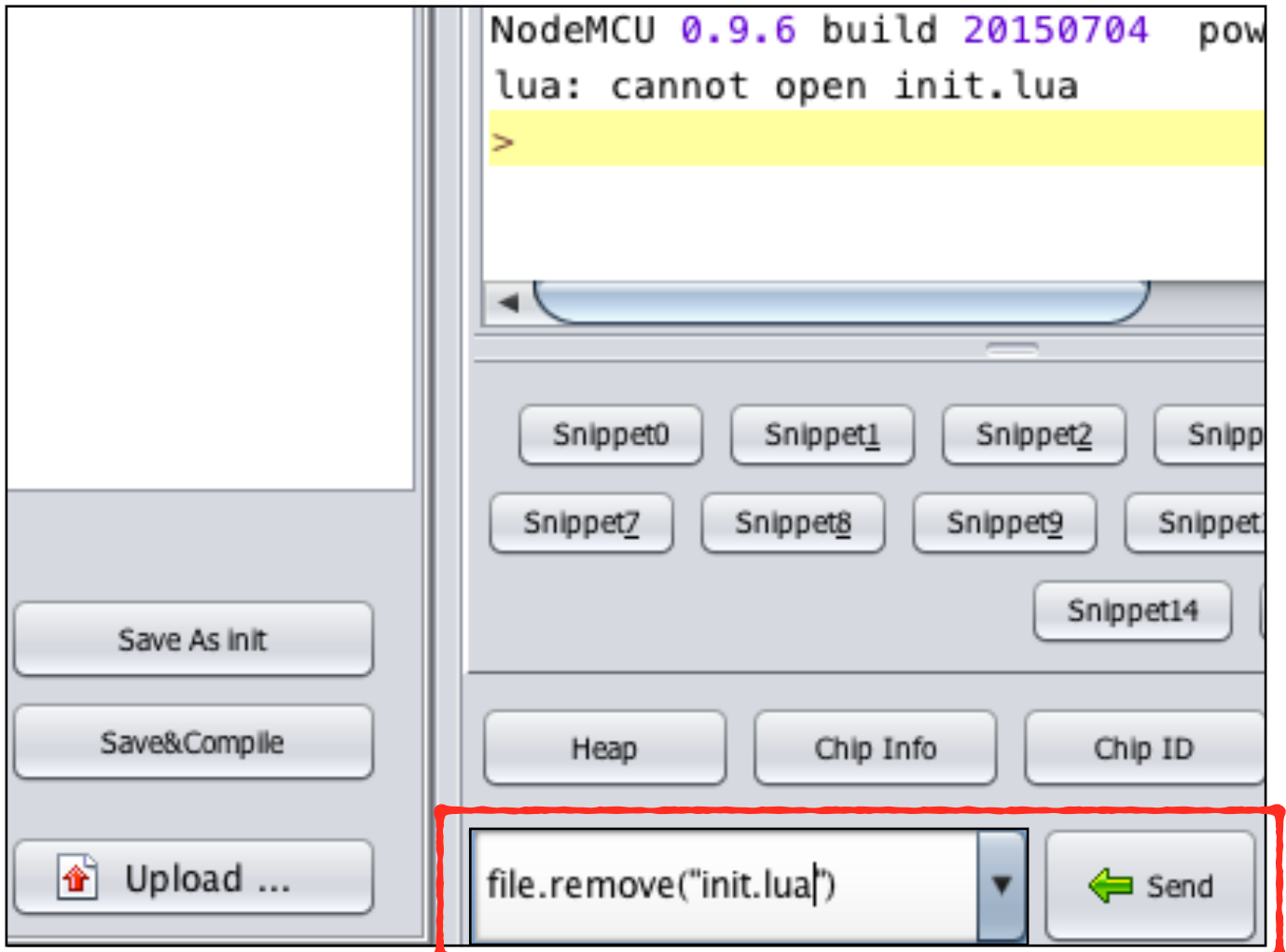
حتى يعمل البرنامج المحفوظ على شريحة ESP8266 تلقائياً من غير مشاكل، فإنه يفضل تسمية جميع ملفات الشيفرة البرمجية بإسم `.init.lua`.
وأفضل طريقة هي أن يتم عمل مجلد خاص لكل شيفرة بحيث يتم تسمية المجلد بإسم المشروع الذي نريده وأما ملف الشيفرة فسيكون بإسم `init.lua` والمثال التالي يوضح المعنى:





حذف البرنامج المحفوظ على شريحة ESP8266:

لا يشترط عمل هذه الخطوة كلما أردنا رفع برنامج جديد. فنستطيع رفع برنامج جديد على شريحة ESP8266 دون حذف البرامج القديمة. ولكن إذا أردنا حذف البرنامج نهائياً لتصبح شريحة ESP8266 خالية منه فنقوم بكتابة `file.remove("init.lua")` في الخانة الموضحة في الصورة ثم سنضغط على زر `send` وبعدها سيتم حذف البرنامج. أما إذا أردنا حذف جميع البرامج من شريحة ESP8266 فنقوم بكتابة `file.format()`





المشروع الثاني (PWM):

فكرة المشروع:

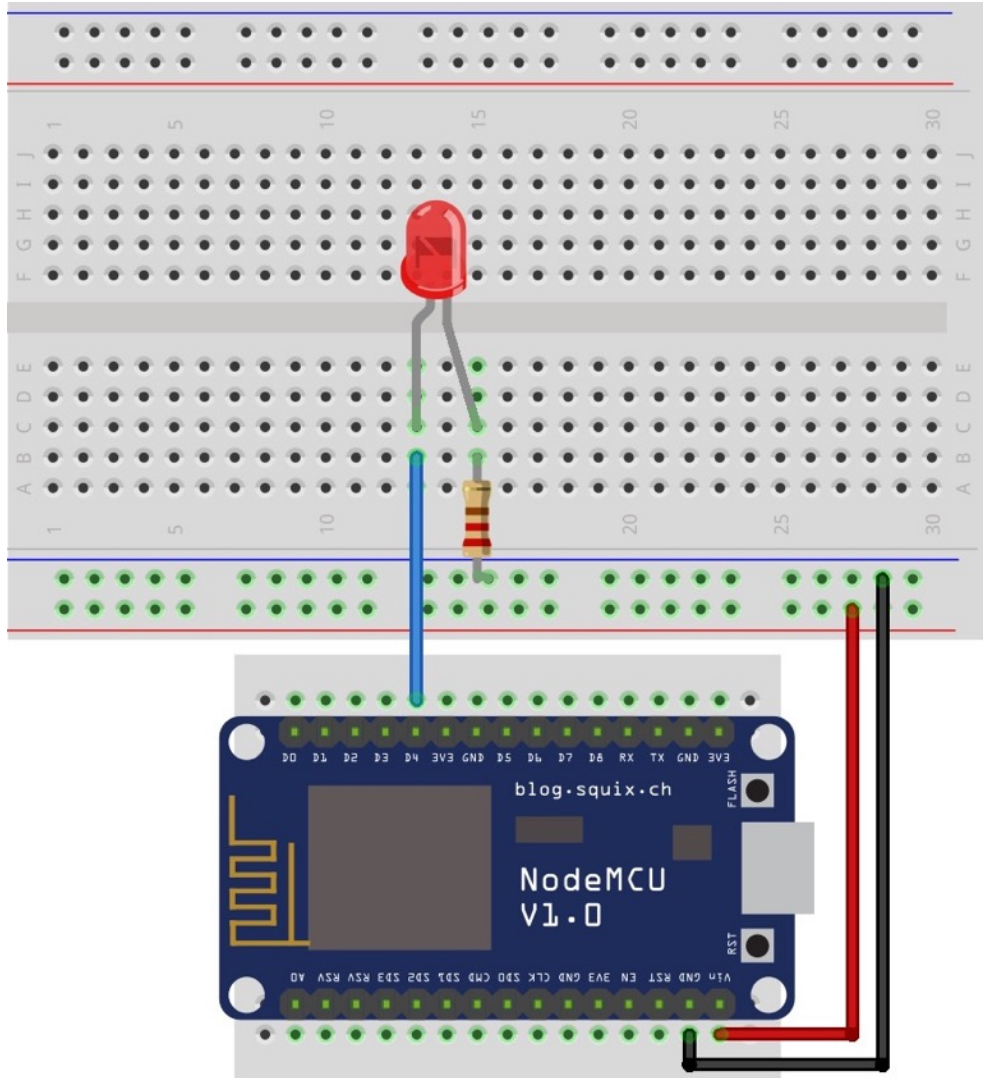
سنقوم بتشغيل الدايود الضوئي باستخدام خاصية PWM. وسنركز في هذا المشروع على شرح الاوامر البرمجية.

الأدوات المستخدمة:

- لوحة NodeMCU
- حامل لوحة NodeMCU
- لوحة تثبيت القطع الالكترونية (Breadboard)
- مقاومة 220 اوم
- دايود ضوئي (LED)



الدائرة الالكترونية:



قمنا بتوصيل الطرف الموجب للديود الضوئي (LED) في المدخل D4 من لوحة NodeMCU والطرف السالب للديود الضوئي (LED) متصل بالارضي (GND) من خلال المقاومة 220.



كتابة الشيفرة البرمجية وشرحها:

الشيفرة البرمجية لهذا المشروع موجودة على الرابط التالي:

www.github/pwm.lua

بعد ذلك سنقوم بفتح برنامج ESPlorer لكتابة الشيفرة البرمجية.

```
init.lua
1 led=4
2 c=0
3 x=0
4
5 gpio.mode(led,gpio.OUTPUT)
```

السطر	الشرح
1	قمنا بتعريف متغير باسم led وهو يشير الى المدخل D4
2	قمنا بتعريف متغير باسم c وحددناه بقيمة افتراضية وهي 0
3	قمنا بتعريف متغير باسم x وحددناه بقيمة افتراضية وهي 0
5	تهيئة الدايمود الضوئي كخرج (output)



```
6 pwm.setup(led, 1000, 0)
7 pwm.start(led)
```

السطر	الشرح
6	هذه الأمر لضبط خصائص PWM.
7	لبدء تشغيل الدايود الضوئي بخاصية PWM.

الأمر `pwm.setup(led, 1000, 0)` يحتوي على 3 عناصر:

العنصر الأول (**led**): يشير الى المدخل الذي نريد تنفيذ خاصية PWM عليه. وفي هذا المشروع فإن المدخل المستخدم هو D4 الذي قمنا بتعريفه بـ `led` العنصر الثاني (**1000**): يشير الى التردد (Frequency) الذي ستعمل به خاصية PWM. و يأخذ قيم متعددة من 0 إلى 1000.

العنصر الثالث (**0**): يشير الى قيمة الـ `duty cycle`. ويأخذ قيم متعددة من 0 إلى 1023 وهذه القيم هي التي تتحكم بشدة التشغيل. وفي هذا المشروع سنتحكم بالدايود الضوئي، لذلك فإن القيمة 0 ستجعل الدايود الضوئي ينطفئ. والقيمة 1023 ستجعل الدايود الضوئي يعمل بأعلى شدة إضاءة. والقيمة 512 ستجعل الدايود الضوئي يعمل بشدة إضاءة متوسطة. وكذلك بقية القيم فإن شدة إضاءة الدايود الضوئي تزداد كلما زادت القيمة من 0 إلى 1023 والعكس صحيح.



```
9  for x=0,2,1 do
10
11  for c=0,1023,1 do
12
13  pwm.setduty(led,c)
14  tmr.delay(1000)
15  end
```

السطر	الشرح
9	دالة فور (for) الأولى: ستقوم بتكرار تنفيذ الاوامر التي داخلها بعدد 3 مرات (من 0 الى 2)
11	دالة فور (for) الثانية: هي أحد الأوامر التي بداخل دالة فور الأولى. وهي أيضا ستقوم بتكرار تنفيذ الأوامر التي داخلها بعدد 1024 مرة (من 0 الى 1023) تصاعديا.
13	هذا الأمر هو أحد الأوامر التي بداخل دالة فور الثانية. وهو لتشغيل الدايود الضوئي بقيمة المتغير c. حيث أن المتغير c يأخذ القيم من 0 الى 1023 تصاعديا (شدة الإضاءة ستزيد تدريجيا).
14	هذا الأمر أيضا يكون داخل دالة فور الثانية. وهو عبارة عن دالة تأخير بوحدة ميكرو ثانية. ومدة التأخير هنا هي 1000 ميكرو ثانية
15	نهاية دالة فور (for) الثانية.



```
17 for c=1023,0,-1 do
18
19 pwm.setduty(led,c)
20 tmr.delay(1000)
21 end
22
23 end
```

السطر	الشرح
17	دالة فور (for) الثالثة: هي أحد الأوامر التي بداخل دالة فور الأولى. وهي أيضا ستقوم بتكرار تنفيذ الأوامر التي داخلها بعدد 1024 مرة (من 1023 الى 0) تنازليا.
19	هذا الأمر هو أحد الأوامر التي بداخل دالة فور الثالثة. وهو لتشغيل الدايود الضوئي بقيمة المتغير C. حيث أن المتغير C يأخذ القيم من 1023 الى 0 تنازليا (شدة الإضاءة ستقل تدريجيا).
20	هذا الأمر أيضا يكون داخل دالة فور الثانية. وهو عبارة عن دالة تأخير بوحدة ميكرو ثانية. ومدة التأخير هنا هي 1000 ميكرو ثانية
21	نهاية دالة فور (for) الثالثة.
23	نهاية دالة فور (for) الأولى.



في هذه الصفحة والصفحتين التي تليها سأشرح دالة فور (for) بشكل مفصل:

```
for x=0, 9, 1 do  
  print("Arabic")  
end
```

دالة فور (for) هي دالة شرطية تكرارية تحتوي على 3 قيم رئيسية وهي:
القيمة الأولى (x=0): تعبر عن القيمة الابتدائية للدالة.

القيمة الثانية (9): تعبر عن القيمة النهائية التي ستتوقف عندها الدالة.

القيمة الثالثة (1): مقدار الزيادة بعد كل دورة (تكرار).

المفهوم الكامل لهذا المثال: في البداية ستكون قيمة المتغير x بـ 0 وسيتم طباعة العبارة (Arabic). ثم سيضاف 1 الى قيمة x وستصبح قيمته بـ 1 وسيتم طباعة العبارة (Arabic) مرة أخرى. ثم سيضاف 1 الى قيمة x وستصبح قيمته بـ 2 وسيتم طباعة العبارة (Arabic) مرة أخرى. ثم سيضاف 1 الى قيمة x وستصبح قيمته بـ 3 وسيتم طباعة العبارة (Arabic) مرة أخرى. وهكذا يستمر بنفس الطريقة حتى تصبح قيمة x بـ 9 وسيتم طباعة العبارة (Arabic) مرة أخرى. ثم سيضاف 1 الى قيمة x وستصبح قيمته بـ 10 وعندها لن يتم طباعة العبارة (Arabic) لأن 10 أكبر من القيمة النهائية لدالة فور



```
for x=1, 5, 2 do
```

```
  print("good")
```

```
end
```

القيمة الأولى ($x=1$): تعبر عن القيمة الابتدائية للدالة.

القيمة الثانية (5): تعبر عن القيمة النهائية التي ستتوقف عندها الدالة.

القيمة الثالثة (2): مقدار الزيادة بعد كل دورة (تكرار).

المفهوم الكامل لهذا المثال: في البداية ستكون قيمة المتغير x بـ 1 وسيتم

طباعة العبارة (good). ثم سيضاف 2 الى قيمة x وستصبح قيمته بـ 3

وسيتم طباعة العبارة (good) مرة أخرى. ثم سيضاف 2 الى قيمة x

وستصبح قيمته بـ 5 وسيتم طباعة العبارة (good) مرة أخرى. ثم

سيضاف 2 الى قيمة x وستصبح قيمته بـ 7 وعندها لن يتم طباعة العبارة

(good) لأن 7 أكبر من القيمة النهائية لدالة فور (for)



```
for x=3, 0, -1 do
```

```
  print("man")
```

```
end
```

القيمة الأولى ($x=3$): تعبر عن القيمة الابتدائية للدالة.

القيمة الثانية (0): تعبر عن القيمة النهائية التي ستتوقف عندها الدالة.

القيمة الثالثة (-1): مقدار التناقص بعد كل دورة (تكرار).

المفهوم الكامل لهذا المثال: في البداية ستكون قيمة المتغير x بـ 3 وسيتم

طباعة العبارة (man). ثم سيطرح 1 من قيمة x وستصبح قيمته بـ 2

وسيتم طباعة العبارة (man) مرة أخرى. ثم سيطرح 1 من قيمة x

وستصبح قيمته بـ 1 وسيتم طباعة العبارة (man) مرة أخرى. ثم سيطرح

1 من قيمة x وستصبح قيمته بـ 0 وسيتم طباعة العبارة (man) مرة أخرى.

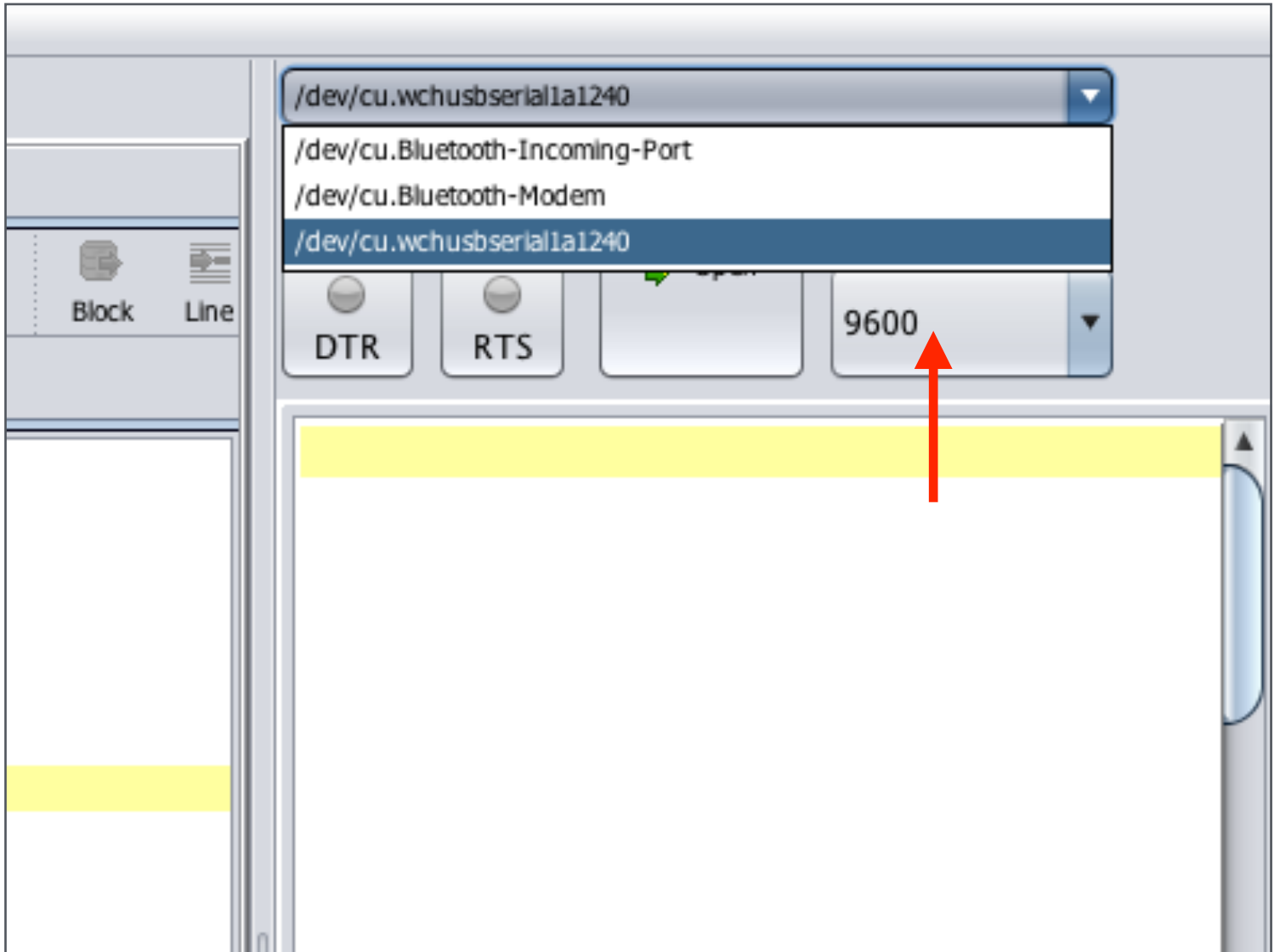
ثم سيطرح 1 من قيمة x وستصبح قيمته بـ -1 وعندها لن يتم طباعة

العبارة (good) لأن -1 أصغر من القيمة النهائية لدالة فور (for)



رفع البرنامج على لوحة NodeMCU وتشغيل المشروع:

بعد الانتهاء من كتابة الشيفرة البرمجية سنقوم بتوصيل لوحة NodeMCU بالحاسب. وبعدها سنقوم بإختيار المنفذ المتصل بلوحة NodeMCU. وأيضا سنختار 9600 كسرعة نقل البيانات كما هو موضح في الصورة التالية:



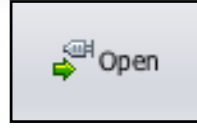
إن لم تجد المنفذ ضمن الخيارات فعليك بالضغط على زر التحديث وبعدها سيظهر معك المنفذ بإذن الله.



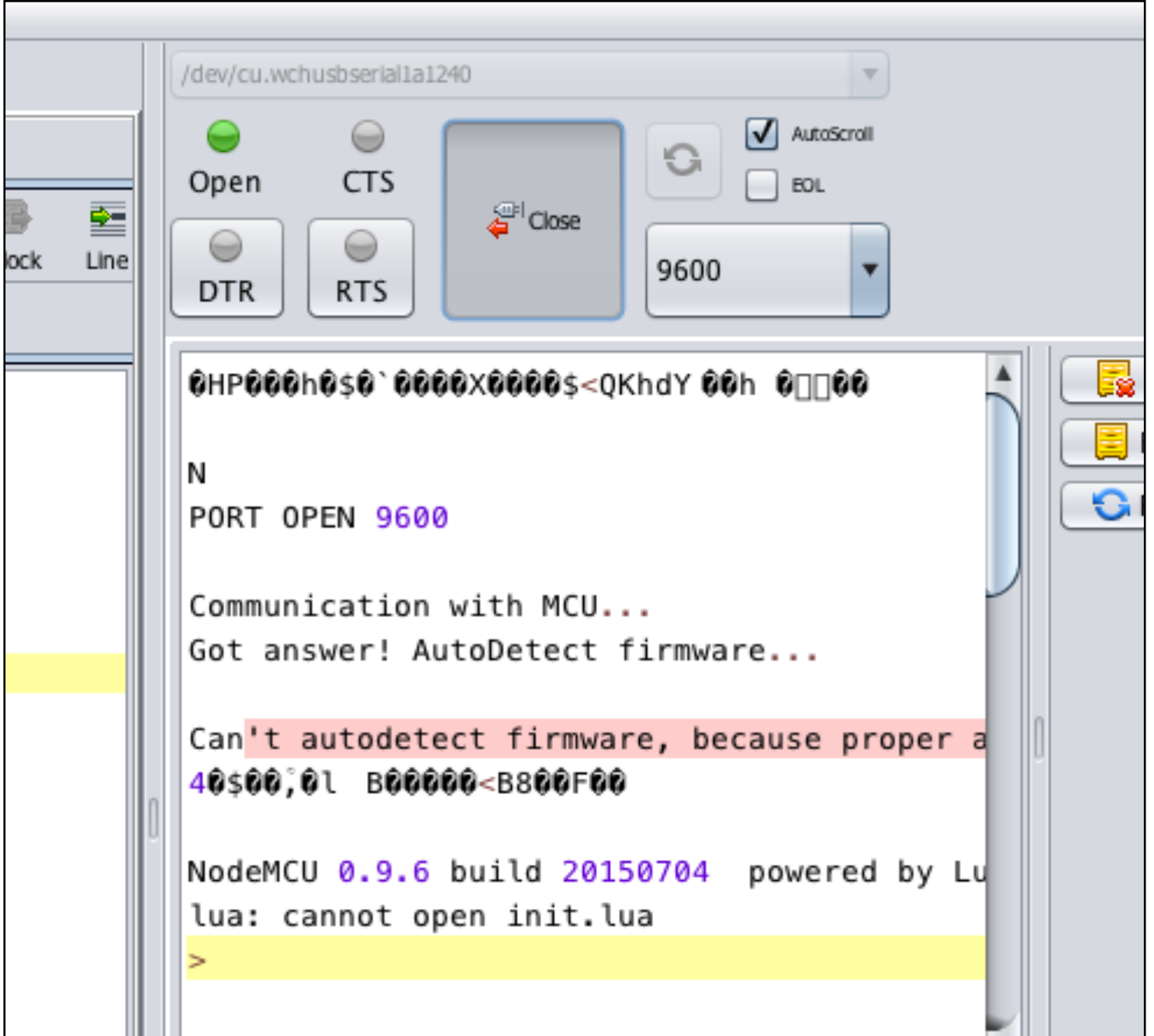
NodeMCU

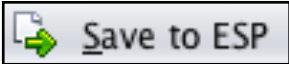


وسيداً عملية الاتصال كما



ثم سنضغط على زر تفعيل الاتصال
هو موضح في الصورة التالية:





بعد ذلك سنرفع البرنامج بالضغط على زر (save to ESP)



أو بالضغط على زر (send to ESP)

والفرق بين الاثنين هو:

- زر (save to ESP): يقوم بحفظ البرنامج داخل شريحة ESP8266 حتى عند فصل اللوحة من الحاسب وتشغيلها باستخدام مصدر خارجي للتغذية (بطاريات) فإن المشروع سيعمل لأن البرنامج محفوظ داخل شريحة ESP8266.

- زر (send to ESP): يقوم فقط بقراءة البرنامج دون حفظه ولا يمكن تشغيل المشروع عند فصل اللوحة عن الحاسب وإستخدام مصدر خارجي للتغذية (بطاريات) لأن البرنامج غير محفوظ داخل شريحة ESP8266.

سأقوم بالضغط على زر (save to ESP) و سيطلب منا أن نقوم بحفظ الشيفرة البرمجية بإسم وسوف أسميه بـ `init.lua` كما هو موضح في الصورة التالية:

File Name:

Files of Type:

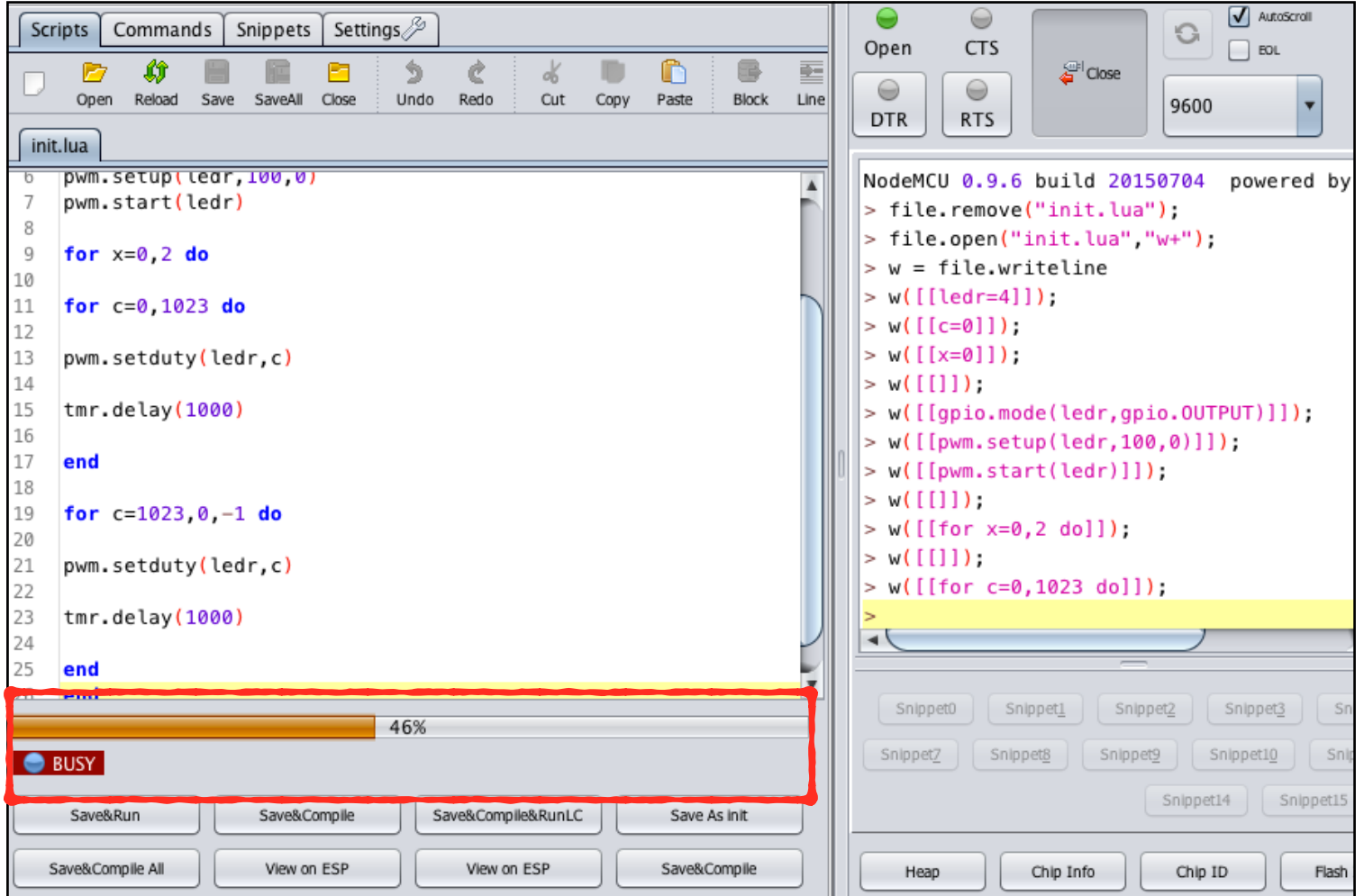
Save Cancel



NodeMCU



بعد الانتهاء من حفظ الشيفرة بإسم سيتم رفع البرنامج كما هو موضح في الصورة التالية:



بعد انتهاء الرفع سنلاحظ أنه ستزيد شدة إضاءة الدايدود الضوئي تدريجيا الى أن تصل إلى أعلى شدة إضاءة ثم تبدأ بالإنخفاض حتى ينطفئ. ستتكرر هذه العملية 3 مرات

رابط فيديو لمشاهدة تشغيل المشروع:

<https://www.youtube.com/watch?v=7-YWds1xuEc>



ملاحظة حول تسمية ملفات الشيفرة البرمجية:

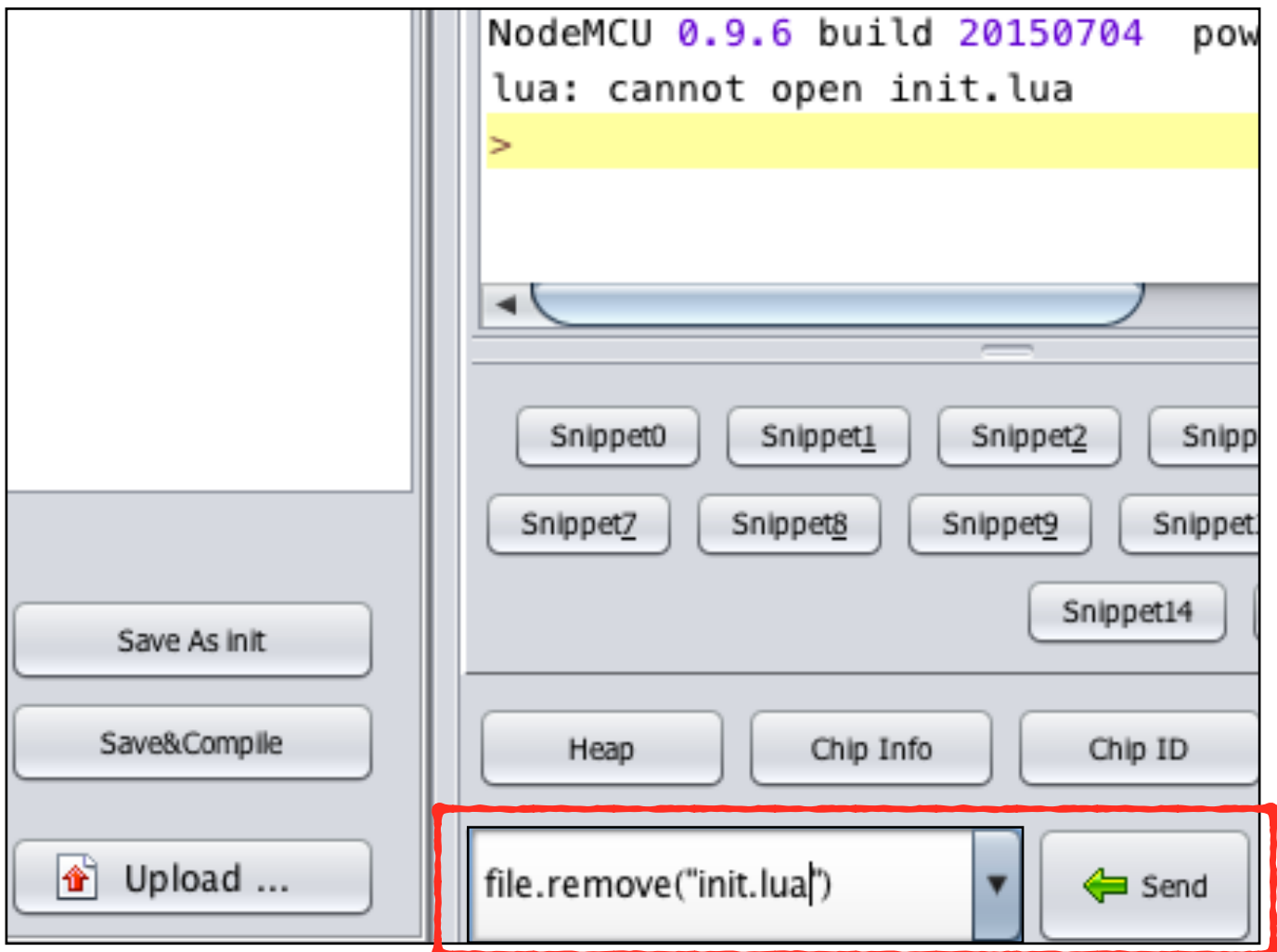
حتى يعمل البرنامج المحفوظ على شريحة ESP8266 تلقائياً من غير مشاكل، فإنه يفضل تسمية جميع ملفات الشيفرة البرمجية بإسم `.init.lua`.
وأفضل طريقة هي أن يتم عمل مجلد خاص لكل شيفرة بحيث يتم تسمية المجلد بإسم المشروع الذي نريده وأما ملف الشيفرة فسيكون بإسم `init.lua` والمثال التالي يوضح المعنى:





حذف البرنامج المحفوظ على شريحة ESP8266:

لا يشترط عمل هذه الخطوة كلما أردنا رفع برنامج جديد. فنستطيع رفع برنامج جديد على شريحة ESP8266 دون حذف البرامج القديمة. ولكن إذا أردنا حذف البرنامج نهائياً لتصبح شريحة ESP8266 خالية منه فنقوم بكتابة (`file.remove("init.lua")`) في الخانة الموضحة في الصورة ثم سنضغط على زر `send` وبعدها سيتم حذف البرنامج. أما إذا أردنا حذف جميع البرامج من شريحة ESP8266 فنقوم بكتابة (`file.format()`)





المشروع الثالث (واجهة تشغيل وإطفاء) :

فكرة المشروع:

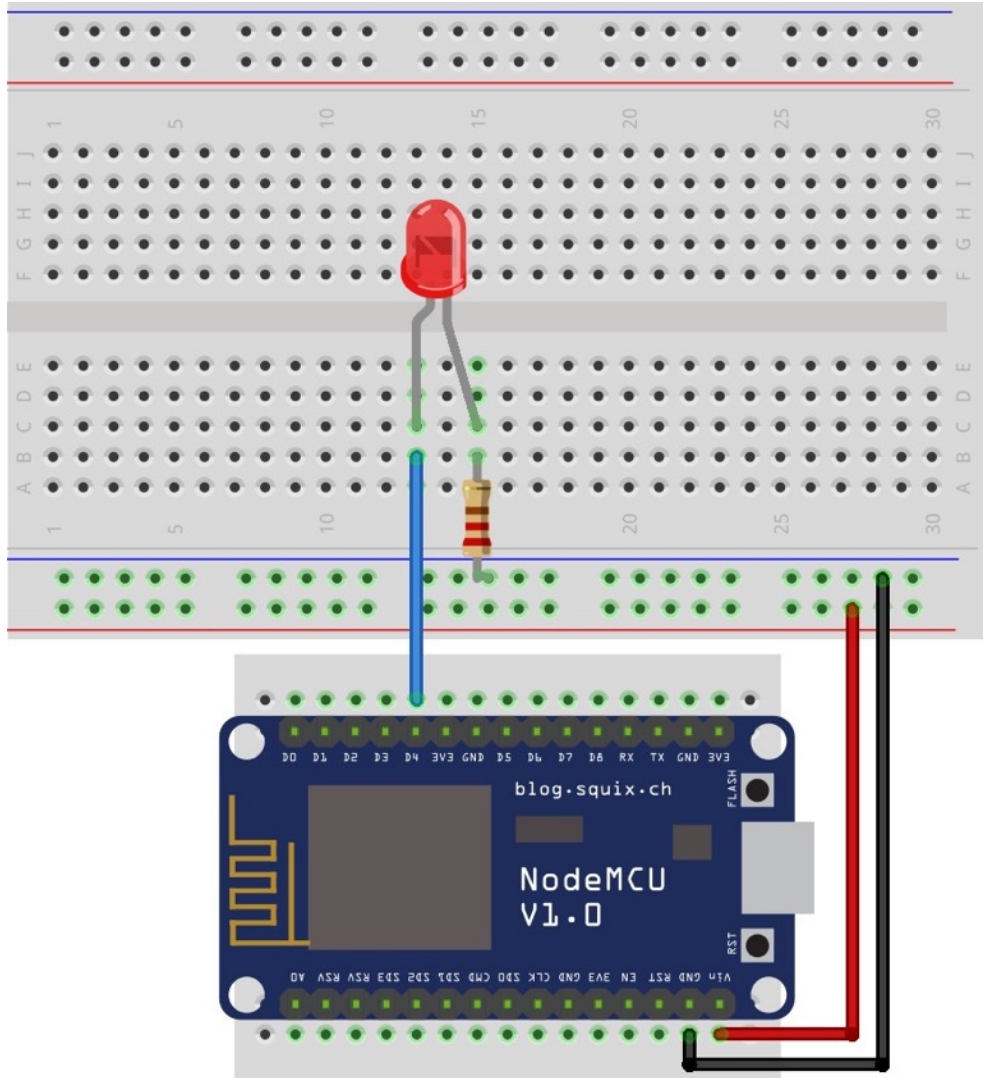
سنقوم بإنشاء صفحة خادم الشبكة تحتوي على مفتاح لتشغيل الدايود الضوئي ومفتاح آخر لإطفائه.

الأدوات المستخدمة:

- لوحة NodeMCU
- حامل لوحة NodeMCU
- لوحة تثبيت القطع الالكترونية (Breadboard)
- مقاومة 220 اوم
- دايود ضوئي (LED)



الدائرة الالكترونية:



قمنا بتوصيل الطرف الموجب للديود الضوئي (LED) في المدخل D4 من لوحة NodeMCU والطرف السالب للديود الضوئي (LED) متصل بالارضي (GND) من خلال المقاومة 220.



كتابة الشيفرة البرمجية وشرحها:

الشيفرة البرمجية لهذا المشروع موجودة على الرابط التالي:

www.github/on-off-button.lua

بعد ذلك سنقوم بفتح برنامج ESPlorer لكتابة الشيفرة البرمجية.

```
1 wifi.setmode(wifi.STATION)
2 wifi.sta.config("YOUR_NETWORK_NAME", "YOUR_NETWORK_PASSWORD")
3 print(wifi.sta.getip())
4
5 led = 4
6 gpio.mode(led, gpio.OUTPUT)
```

السطر	الشرح
1	إعداد شريحة ESP8266 لإستقبال اشارة الواي فاي (wifi). أي أنه سيتم تشغيلها كعميل (client).
2	للإتصال بشبكة الواي فاي (wifi). وتحتوي على خانتين: your_network_name: سنكتب اسم الشبكة التي نريد الاتصال بها your_network_password: سنكتب الرقم السري الخاص بالشبكة
3	لطباعة عنوان IP الذي سنستخدمه للدخول الى صفحة خادم الشبكة
5	قمنا بتعريف متغير بإسم led وهو يشير الى المدخل D4
6	تهيئة الدايمود الضوئي كخرج (output)



NodeMCU



```

8  srv=net.createServer(net.TCP)
9  srv:listen(80,function(conn)

```

السطر	الشرح
8	المنفذ 80 (port 80) يستخدم للاتصال بين الخادم و العميل. وسنقوم
9	بإنشاء صفحة خادم الشبكة على هذا المنفذ

```

11  conn:on("receive", function(client,request)
12  local buf = "";
13      buf = buf.."HTTP/1.1 200 OK\n\n"
14  local _, _, method, path, vars = string.find(request, "([A-Z]+) (.+)?(.+) HTTP");
15  if(method == nil)then
16      _, _, method, path = string.find(request, "([A-Z]+) (.+) HTTP");
17  end
18
19  local _GET = {}
20  if (vars ~= nil)then
21  for k, v in string.gmatch(vars, "(%w+)=(%w+)&*" ) do
22      _GET[k] = v
23  end
24  end

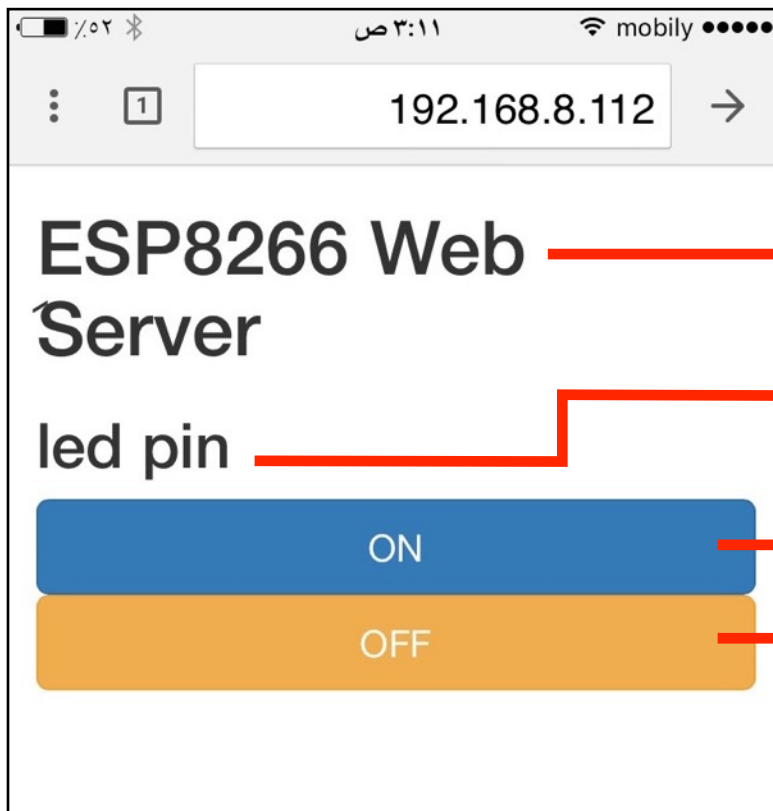
```

السطر	الشرح
11	داخل هذه الدالة سنقوم بكتابة الاوامر التي تمكننا من تصميم وتنسيق صفحة خادم الشبكة.
12	مجموعة من المتغيرات التي ستقوم بحفظ وتخزين صفحة خادم
24	الشبكة وأيضا حفظ وتخزين رابط الصفحة (URL).



```
26 buf = buf.."<head>";
27 buf = buf.."<meta charset='utf-8'>";
28 buf = buf.."<meta http-equiv='X-UA-Compatible' content='IE=edge'>";
29 buf = buf.."<meta name='viewport' content='width=device-width, initial-scale=1'>";
30 buf = buf.."<script src='https://code.jquery.com/jquery-2.1.3.min.js'></script>";
31 buf = buf.."<link rel='stylesheet' href='https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css'>";
32 buf = buf.."</head><div class='container'>";
33 buf = buf.."<h1>ESP8266 Web Server</h1>";
34 buf = buf.."<h2>led pin</h2>";
35 buf = buf.."<div class='row'>";
36 buf = buf.."<div class='col-md-2'><a href='?pin=ON' class='btn btn-block btn-lg btn-primary' role='button'>ON</a></div>";
37 buf = buf.."<div class='col-md-2'><a href='?pin=OFF' class='btn btn-block btn-lg btn-warning' role='button'>OFF</a></div>";
38 buf = buf.."</div></div>";
```

السطر	الشرح
26	هذه الأوامر هي التي تستخدم لإنشاء الصفحات الالكترونية باستخدام البوتستراب (Bootstrap) وقد تحدثنا عنها في الفصل السابق
38	(صفحة خادم الشبكة).



السطر 33 من الشيفرة البرمجية

السطر 34 من الشيفرة البرمجية

السطر 36 من الشيفرة البرمجية

السطر 37 من الشيفرة البرمجية



للدخول الى صفحة خادم الشبكة سنقوم بكتابة عنوان IP في المتصفح كما هو ظاهر في الصورة السابقة **192.168.8.112** (يختلف من شخص لشخص آخر).

وعند الضغط على مفتاح التشغيل (ON) ذو اللون الأزرق، فإن المتصفح تلقائيا سيذهب الى العنوان **192.168.8.112/?pin=ON** وسنلاحظ أن

المتغير pin أصبحت قيمته تساوي .ON



وعند الضغط على مفتاح الاطفاء (OFF) ذو اللون الأصفر، فإن المتصفح تلقائيا سيذهب الى العنوان **192.168.8.112/?pin=OFF** وسنلاحظ أن

المتغير pin أصبحت قيمته تساوي .OFF





```
40  if(_GET.pin == "ON") then
41      gpio.write(led, gpio.HIGH);
42  elseif(_GET.pin == "OFF") then
43      gpio.write(led, gpio.LOW);
44  end
```

السطر	الشرح
40	سيختبر هل قيمة المتغير pin هي ON أو لا. فإذا كانت هي فسيتم
41	تشغيل الدايود الضوئي.
42	سيختبر هل قيمة المتغير pin هي OFF أو لا. فإذا كانت هي فسيتم
43	إطفاء الدايود الضوئي.
44	نهاية الدالة الشرطية if.



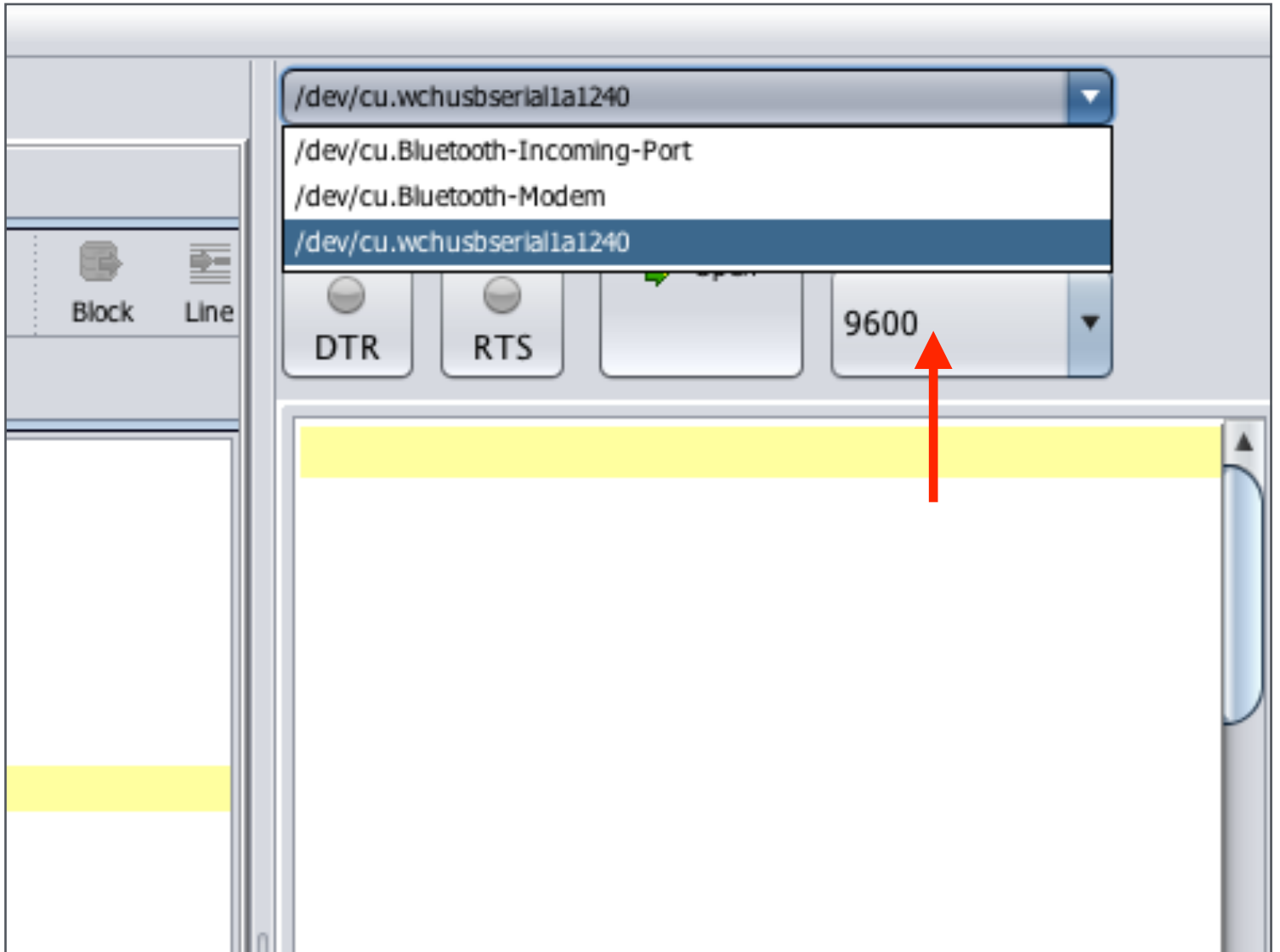
```
45     client:send(buf);
46     client:close();
47     collectgarbage();
48 end)
49 end)
```

السطر	الشرح
45	لعرض محتويات صفحة خادم الشبكة التي قمنا بإنشائها باستخدام Bootstrap.
46	اغلاق اتصال العميل
47	هذه الدالة تسمى بجامع النفايات حيث تقوم بالبحث عن الموارد والبيانات الغير مستخدمة في الذاكرة ومن ثم إزالتها من أجل الحصول على مساحة فارغة في الذاكرة لإستخدامها لأغراض أخرى
48	نهاية الدالة <code>conn:on("receive",function(client,request)</code>
49	نهاية الدالة <code>srv:listen(80,function(conn)</code>



رفع البرنامج على لوحة NodeMCU وتشغيل المشروع:

بعد الانتهاء من كتابة الشيفرة البرمجية سنقوم بتوصيل لوحة NodeMCU بالحاسب. وبعدها سنقوم بإختيار المنفذ المتصل بلوحة NodeMCU. وأيضا سنختار 9600 كسرعة نقل البيانات كما هو موضح في الصورة التالية:



إن لم تجد المنفذ ضمن الخيارات فعليك بالضغط على زر التحديث وبعدها سيظهر معك المنفذ بإذن الله.



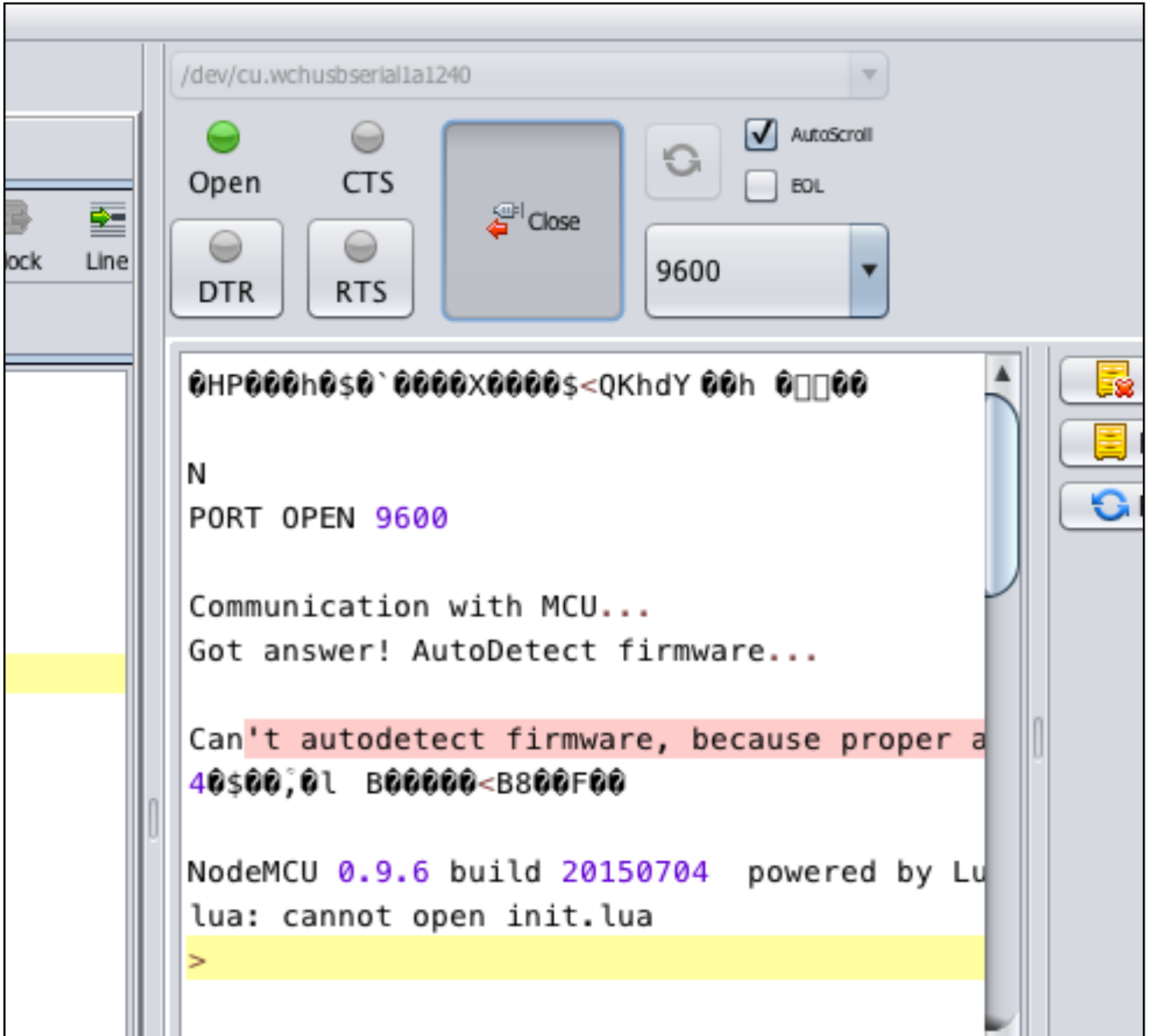
NodeMCU



وسيداً عملية الاتصال كما



ثم سنضغط على زر تفعيل الاتصال
هو موضح في الصورة التالية:





بعد ذلك سنرفع البرنامج بالضغط على زر (save to ESP)



أو بالضغط على زر (send to ESP)

والفرق بين الاثنين هو:

• زر (save to ESP): يقوم بحفظ البرنامج داخل شريحة ESP8266 حتى عند فصل اللوحة من الحاسب وتشغيلها باستخدام مصدر خارجي للتغذية (بطاريات) فإن المشروع سيعمل لأن البرنامج محفوظ داخل شريحة ESP8266.

• زر (send to ESP): يقوم فقط بقراءة البرنامج دون حفظه ولا يمكن تشغيل المشروع عند فصل اللوحة عن الحاسب وإستخدام مصدر خارجي للتغذية (بطاريات) لأن البرنامج غير محفوظ داخل شريحة ESP8266.

سأقوم بالضغط على زر (save to ESP) و سيطلب منا أن نقوم بحفظ الشيفرة البرمجية بإسم وسوف أسميه بـ `init.lua` كما هو موضح في الصورة التالية:

File Name:

Files of Type:



NodeMCU



بعد الانتهاء من حفظ الشيفرة بإسم سيتم رفع البرنامج كما هو موضح في الصورة التالية:

The screenshot displays the NodeMCU IDE interface. On the left, a code editor shows a Lua script named 'init.lua' with the following content:

```
33 buf = buf.."<h1>ESP8266 Web Server</h1>";
34 buf = buf.."<h2>led pin</h2>";
35 buf = buf.."<div class=\"row\">";
36 buf = buf.."<div class=\"col-md-2\"><a href=\"?pin=ON\">ON</a>";
37 buf = buf.."<div class=\"col-md-2\"><a href=\"?pin=OFF\">OFF</a>";
38 buf = buf.."</div></div>";
39
40 if(_GET.pin == "ON")then
41     gpio.write(led, gpio.HIGH);
42 elseif(_GET.pin == "OFF") then
43     gpio.write(led, gpio.LOW);
44 end
45     client:send(buf);
46     client:close();
47 collectgarbage();
48 end)
49 end)
```

Below the code editor, a progress bar shows 37% completion. A red box highlights the status bar, which displays "BUSY" and the file path "/Users/inventor/Desktop/webserver/init.lua". The bottom of the IDE features several buttons: "Save&Run", "Save&Compile", "Save&Compile&RunLC", "Save As Init", "Save&Compile All", "View on ESP", "Send to ESP", "Run", and "Upload ...".

On the right side of the IDE, there is a terminal window showing the output of the script. The output includes the following lines:

```
> w([[led = 4]]);
> w([[gpio.mode(led, gpio.OUTPUT)]];
> w([[ ]]);
> w([[srv=net.createServer(net.TCP)]];
> w([[srv:listen(80,function(conn)]];
> w([[ conn:on("receive", function(cl
> w([[ ]]);
> w([[local buf = "";]]);
> w([[ buf = buf.."HTTP/1.1 200 OK
> w([[local _, _, method, path, vars = s
> w([[if(method == nil)then]];
> w([[ _, _, method, path = s
```

Below the terminal, there are buttons for "Heap", "Chip Info", "Chip ID", and "Flash". At the bottom right, there is a "Send" button and a "CR" checkbox.



الحصول على عنوان IP:

بعد الانتهاء من عملية رفع البرنامج سيظهر لنا عنوان IP بالأسفل كما هو موضح في الصورة التالية:

```
> w([[      buf = buf.."<div class=\"col-
> w([[      buf = buf.."</div></div>"]);]])
> w([[ ]]);
> w([[if(_GET.pin == "ON")then]]);
> w([[      gpio.write(led, gpio.HI
> w([[elseif(_GET.pin == "OFF")then]]);
> w([[      gpio.write(led, gpio.LO
> w([[end]]);
> w([[      client:send(buf);]])];
> w([[      client:close();]])];
> w([[collectgarbage();]])];
> w([[end]]);
> w([[end]]);
> file.close();
> dofile("init.lua");
192.168.8.112 255.255.255.0 192.168.8.1
```

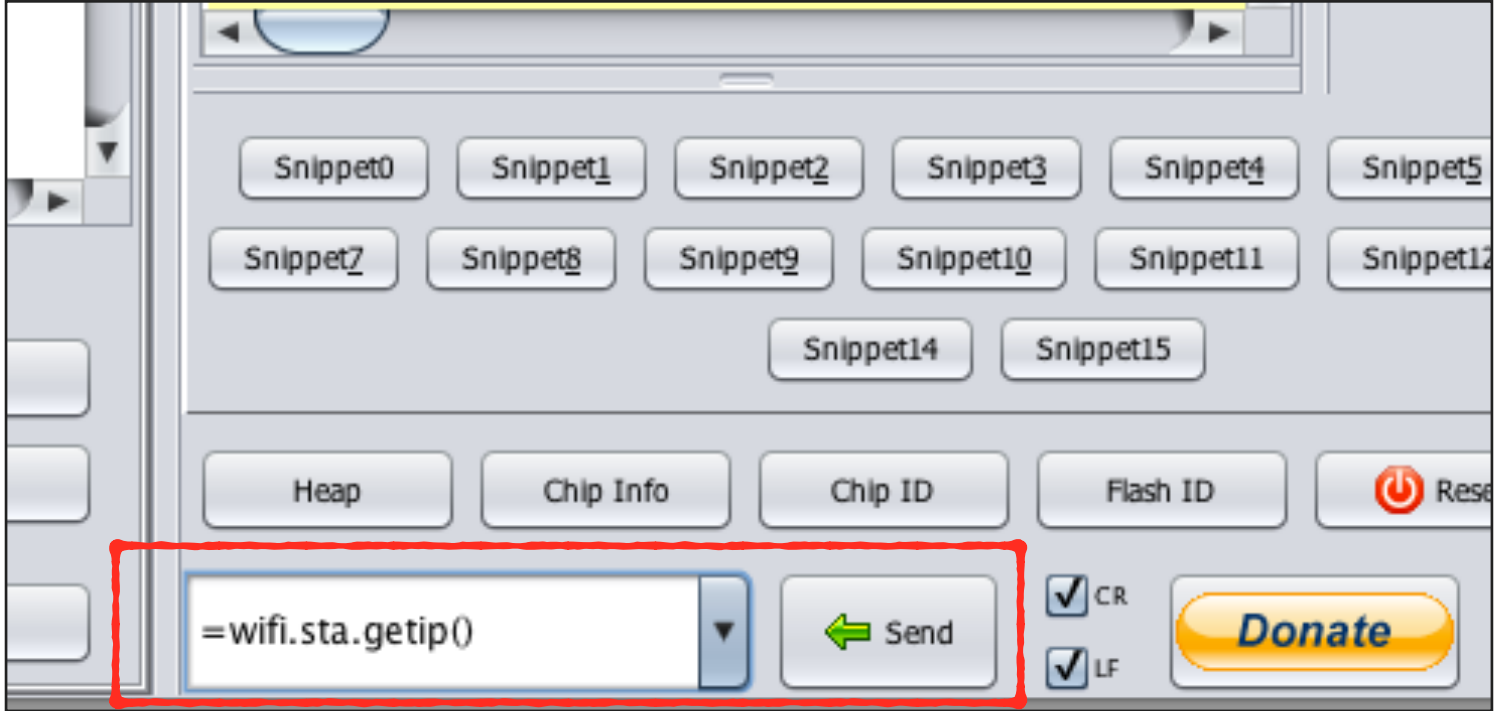
وفي حال أن عنوان IP لم يظهر على الشاشة، فسنقوم بعمل التالي:



NodeMCU



سنقوم بكتابة الأمر `=wifi.sta.getip()` في خانة الإرسال الموضحة في الصورة ثم سنضغط على زر `send` وبعدها سيتم طباعة عنوان IP.



رابط فيديو لمشاهدة تشغيل المشروع:

<https://www.youtube.com/watch?v=ew-xRSakZe4>



ملاحظة حول تسمية ملفات الشيفرة البرمجية:

حتى يعمل البرنامج المحفوظ على شريحة ESP8266 تلقائياً من غير مشاكل، فإنه يفضل تسمية جميع ملفات الشيفرة البرمجية بإسم `.init.lua`.
وأفضل طريقة هي أن يتم عمل مجلد خاص لكل شيفرة بحيث يتم تسمية المجلد بإسم المشروع الذي نريده وأما ملف الشيفرة فسيكون بإسم `init.lua` والمثال التالي يوضح المعنى:



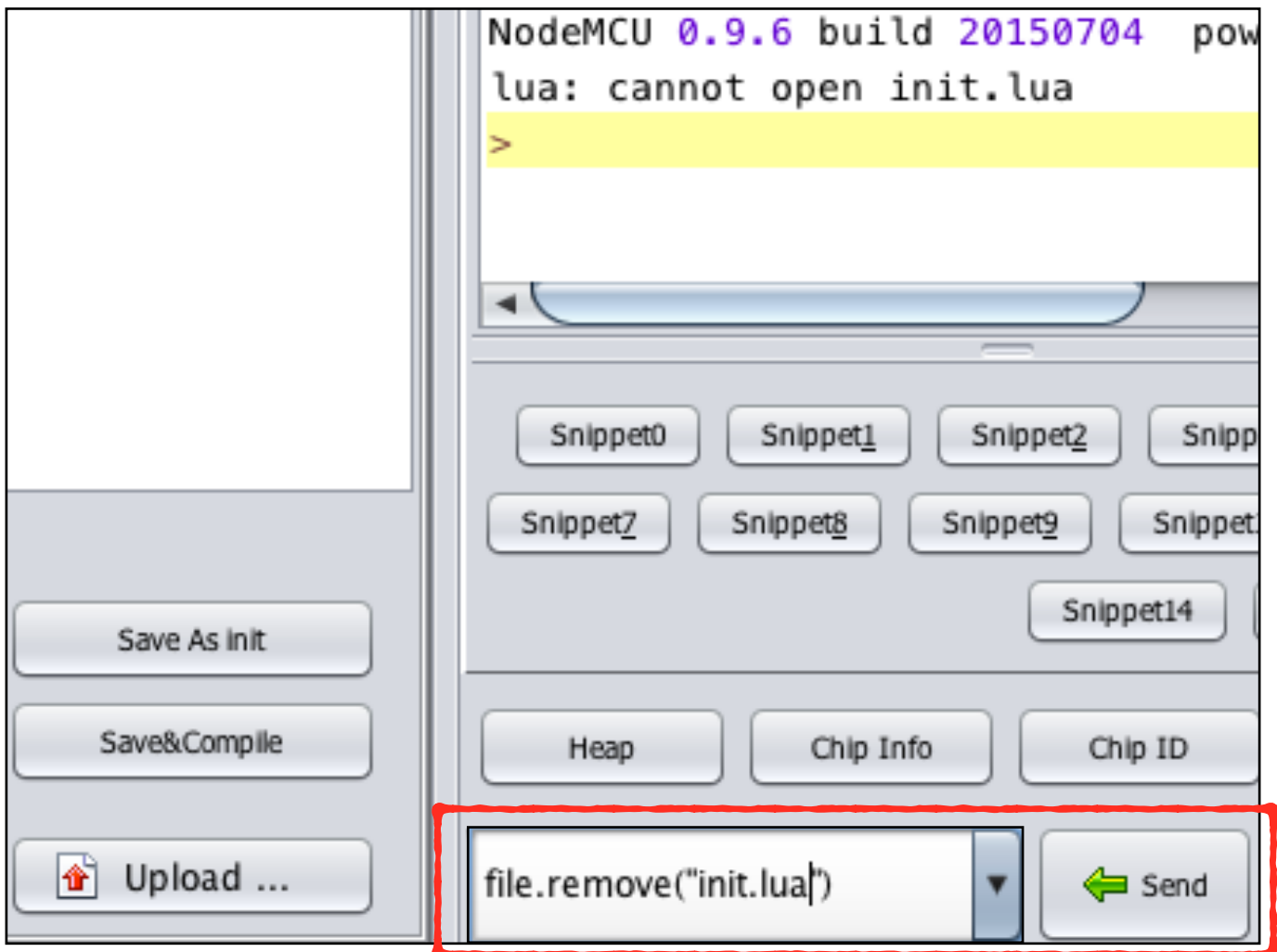
ملاحظات قبل تشغيل المشروع:

- يفضل دائماً إعادة تشغيل لوحة NodeMCU بعد رفع البرنامج الجديد.
لأنه من غير إعادة تشغيل فإنه سيتم تشغيل البرنامج القديم.
- حتى نستطيع الدخول الى صفحة خادم الشبكة يجب ان تكون شبكة الواي فاي (wifi) المتصلة بشريحة ESP8266 هي نفس الشبكة المتصلة بالجهاز الذي نريد التحكم من خلاله.



حذف البرنامج المحفوظ على شريحة ESP8266:

لا يشترط عمل هذه الخطوة كلما أردنا رفع برنامج جديد. فنستطيع رفع برنامج جديد على شريحة ESP8266 دون حذف البرامج القديمة. ولكن إذا أردنا حذف البرنامج نهائياً لتصبح شريحة ESP8266 خالية منه فنقوم بكتابة (`file.remove("init.lua")`) في الخانة الموضحة في الصورة ثم سنضغط على زر `send` وبعدها سيتم حذف البرنامج. أما إذا أردنا حذف جميع البرامج من شريحة ESP8266 فنقوم بكتابة (`file.format()`)





المشروع الرابع (عرض بيانات حساس الضوء):

فكرة المشروع:

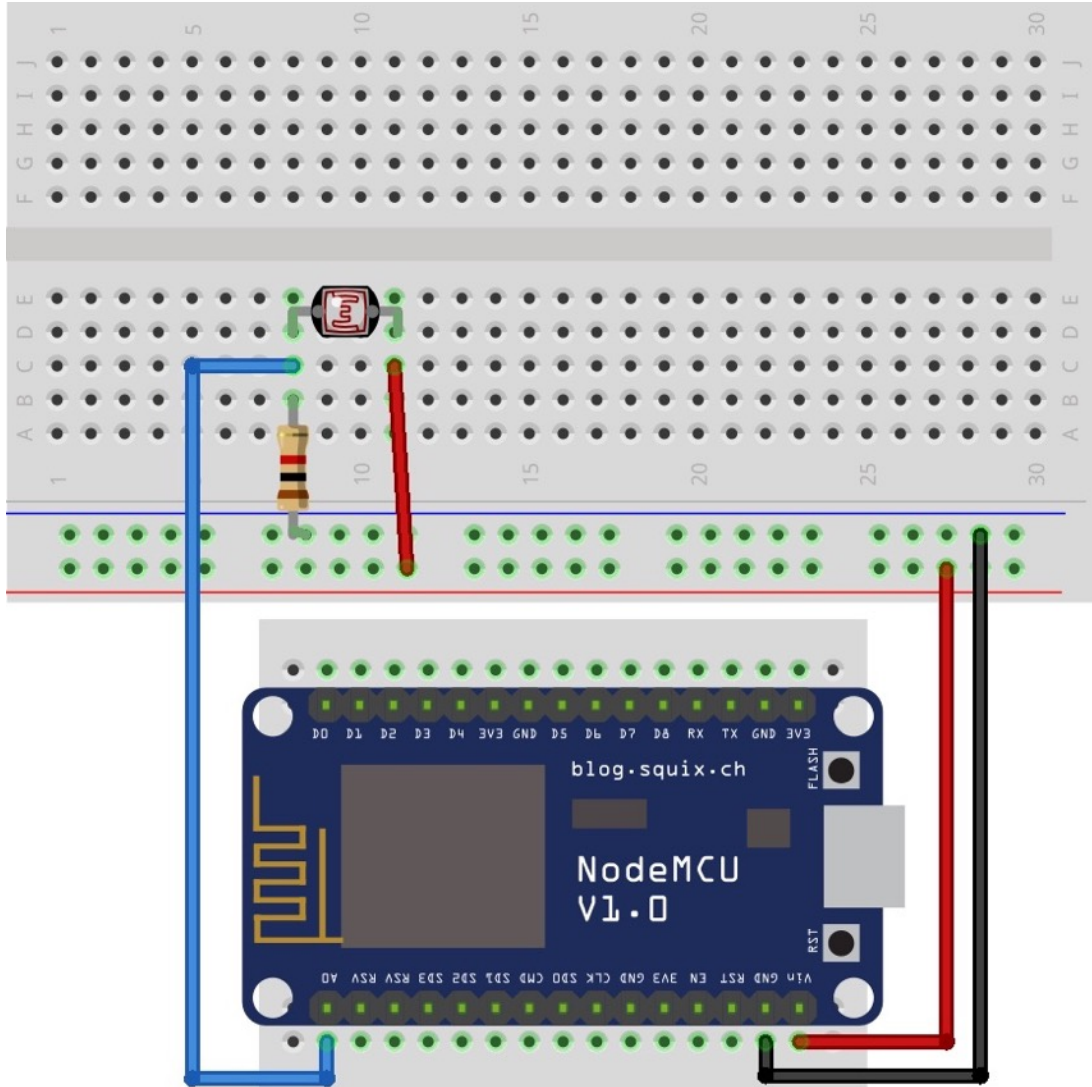
سنقوم بعرض التغير في قوة شدة الاضاءة على صفحة خادم الشبكة

الأدوات المستخدمة:

- لوحة NodeMCU
- حامل لوحة NodeMCU
- مقاومة ضوئية (LDR)
- مقاومة 1 كيلو
- لوحة تثبيت القطع الالكترونية (Breadboard)



الدائرة الالكترونية:



قمنا بتوصيل الطرف الأول للمقاومة الضوئية بالمصدر 5v. والطرف الثاني للمقاومة الضوئية متصل بالأرضي (GND) من خلال المقاومة 1 كيلو. والنقطة المشتركة بين المقاومة الضوئية والمقاومة 1 كيلو متصلة في المدخل A0 من لوحة NodeMCU.



كتابة الشيفرة البرمجية وشرحها:

الشيفرة البرمجية لهذا المشروع موجودة على الرابط التالي:

www.github/ldr.lua

بعد ذلك سنقوم بفتح برنامج ESPlorer لكتابة الشيفرة البرمجية.

```
1 wifi.setmode(wifi.STATION)
2 wifi.sta.config("ssid", "pass")
3 print(wifi.sta.getip())
4 tmr.delay(1000000)
```

السطر	الشرح
1	إعداد شريحة ESP8266 لإستقبال اشارة الواي فاي (wifi). أي أنه سيتم تشغيلها كعميل (client).
2	للإتصال بشبكة الواي فاي (wifi). وتحتوي على خانتين: ssid: سنكتب اسم الشبكة التي نريد الاتصال بها pass: سنكتب الرقم السري الخاص بالشبكة
3	لطباعة عنوان IP الذي سنستخدمه للدخول الى صفحة خادم الشبكة
5	دالة تأخير بوحدة ميكرو ثانية. ومدة التأخير هنا هي ثانية واحدة



```
6 ldr_value=0
7 percent_light=0
8 percent_dark=0
9 bimb=1
10
11 function ldr()
12 ldr_value = adc.read(0)
13 percent_light = ldr_value*(100/1024)
14 percent_dark = 100 - percent_light
15 end
```

السطر	الشرح
6	مجموعة من المتغيرات قيمتها الافتراضية بـ 0
8	
9	متغير سنستخدمه كعداد. وقيمته الافتراضية بـ 1
11	قمنا بإنشاء دالة باسم ldr()
12	الامر adc.read(0) يستخدم لقراءة القيم من المدخل A0 الذي تتصل به المقاومة الضوئية. وسيتم حفظ هذه القيم في المتغير ldr_value
13	لحساب نسبة شدة الاضاءة. وحفظ القيم في المتغير percent_light
14	لحساب نسبة شدة الظلمة. وحفظ القيم في المتغير percent_dark
15	نهاية الدالة ldr()



```
17 tmr.alarm(1,5000, 1, function()  
18   ldr() bimb=bimb+1  
19   if bimb==5 then bimb=0 wifi.sta.connect() print("Reconnect")  
20   end  
21 end)
```

السطر	الشرح
17	دالة مؤقتة تقوم بتنفيذ أمر معين كل فترة زمنية محددة بوحدة ملي ثانية. وهذه الفترة محددة بـ 5000 ملي ثانية (5 ثواني).
18	داخل الدالة المؤقتة سيتم تنفيذ دالة ldr() وأيضا سيتم زيادة قيمة المتغير bimb بمقدار 1 في كل مرة.
19	سيختبر هل قيمة المتغير bimb تساوي 5 ام لا. فإذا كانت تساوي 5، فستصبح قيمة bimb=0 وسيتم الاتصال بالواي فاي (wifi) وأيضا سيتم طباعة العبارة Reconnect
20	نهاية الدالة الشرطية if
21	نهاية الدالة المؤقتة (tmr.alarm)



NodeMCU



```
23 | srv=net.createServer(net.TCP) srv:listen(80,function(conn)  
24 |   conn:on("receive",function(conn,payload)
```

السطر	الشرح
23	المنفذ 80 (port 80) يستخدم للاتصال بين الخادم و العميل. وسنقوم بإنشاء صفحة خادم الشبكة على هذا المنفذ
24	داخل هذه الدالة سنقوم بكتابة الاوامر التي تمكننا من تصميم وتنسيق صفحة خادم الشبكة.



NodeMCU



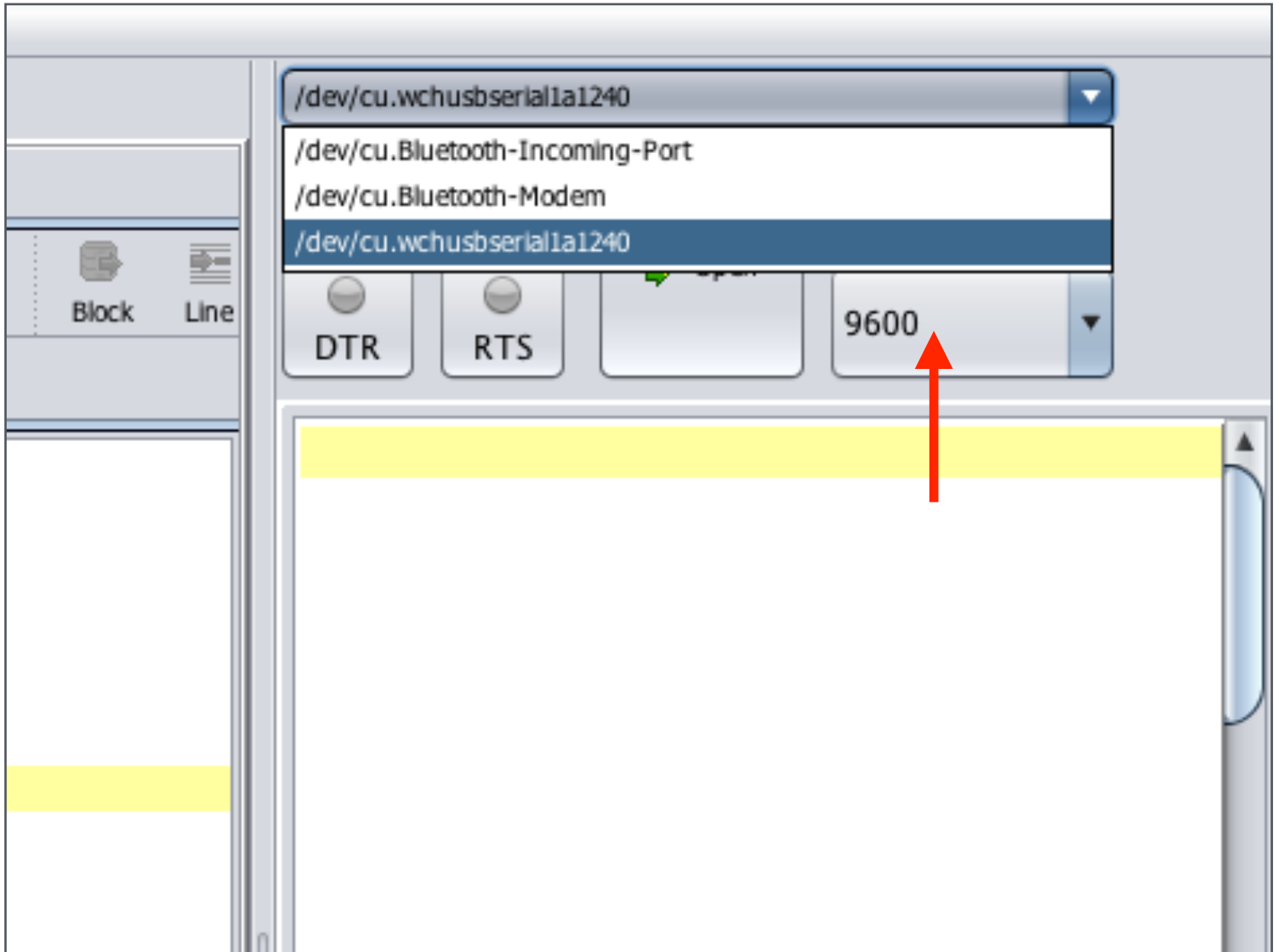
```
43     conn:on("sent",function(conn) conn:close() end)
44     end)
45 end)
```

السطر	الشرح
43	لعرض محتويات صفحة خادم الشبكة التي قمنا بإنشائها باستخدام Bootstrap.
44	نهاية الدالة <code>conn:send('HTTP/1.1 200 OK\r\n....')</code>
45	نهاية الدالة <code>conn:on("receive",function(conn,payload)</code>



رفع البرنامج على لوحة NodeMCU وتشغيل المشروع:

بعد الانتهاء من كتابة الشيفرة البرمجية سنقوم بتوصيل لوحة NodeMCU بالحاسب. وبعدها سنقوم بإختيار المنفذ المتصل بلوحة NodeMCU. وأيضا سنختار 9600 كسرعة نقل البيانات كما هو موضح في الصورة التالية:



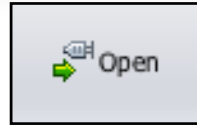
إن لم تجد المنفذ ضمن الخيارات فعليك بالضغط على زر التحديث وبعدها سيظهر معك المنفذ بإذن الله.



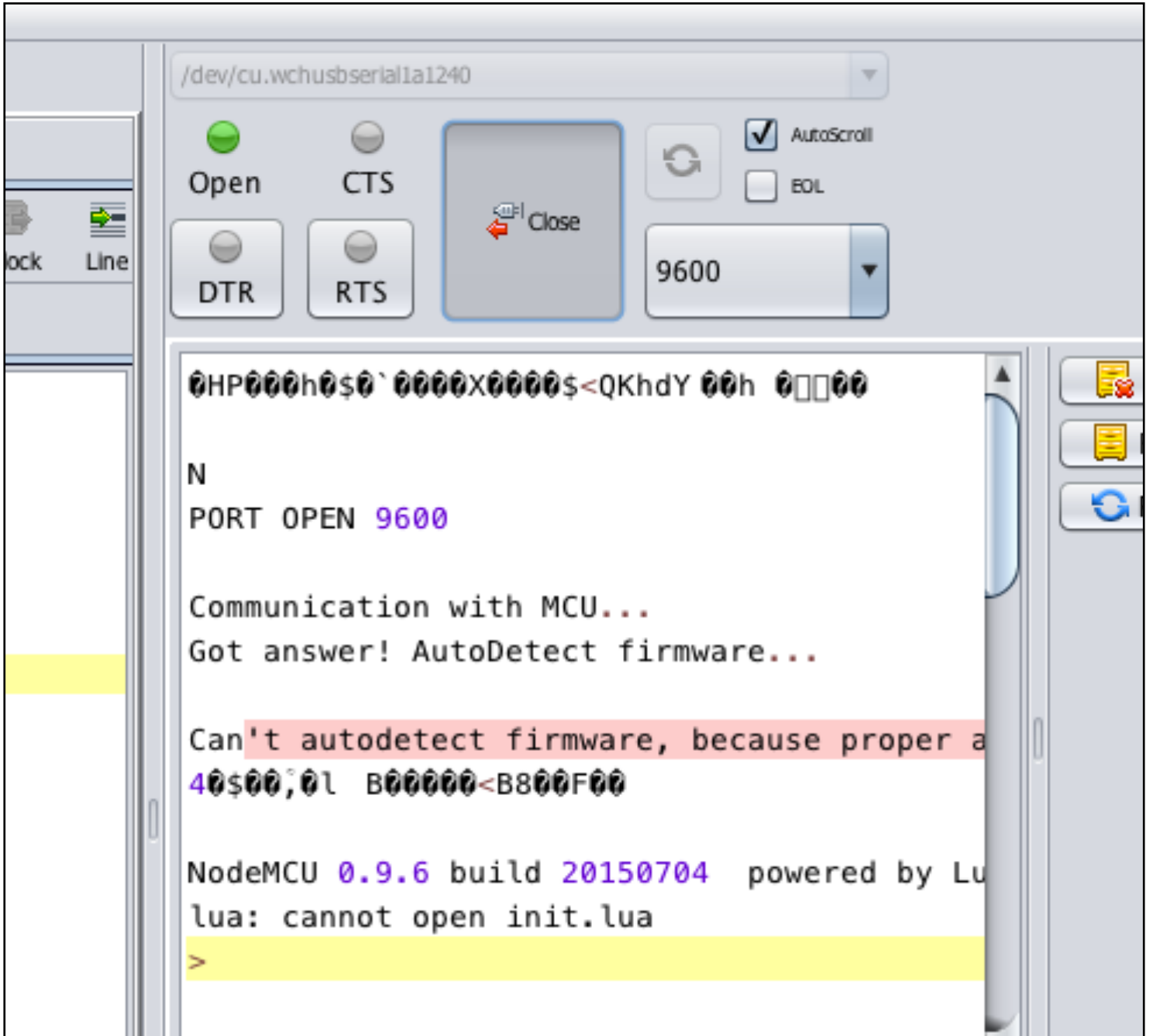
NodeMCU

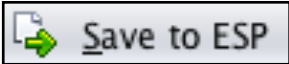


وسيداً عملية الاتصال كما



ثم سنضغط على زر تفعيل الاتصال
هو موضح في الصورة التالية:





بعد ذلك سنرفع البرنامج بالضغط على زر (save to ESP)



أو بالضغط على زر (send to ESP)

والفرق بين الاثنين هو:

- زر (save to ESP): يقوم بحفظ البرنامج داخل شريحة ESP8266 حتى عند فصل اللوحة من الحاسب وتشغيلها باستخدام مصدر خارجي للتغذية (بطاريات) فإن المشروع سيعمل لأن البرنامج محفوظ داخل شريحة ESP8266.

- زر (send to ESP): يقوم فقط بقراءة البرنامج دون حفظه ولا يمكن تشغيل المشروع عند فصل اللوحة عن الحاسب وإستخدام مصدر خارجي للتغذية (بطاريات) لأن البرنامج غير محفوظ داخل شريحة ESP8266.

سأقوم بالضغط على زر (save to ESP) و سيطلب منا أن نقوم بحفظ الشيفرة البرمجية بإسم وسوف أسميه بـ `init.lua` كما هو موضح في الصورة التالية:

File Name:

Files of Type:

Save Cancel



NodeMCU



بعد الانتهاء من حفظ الشيفرة بإسم سيتم رفع البرنامج كما هو موضح في الصورة التالية:

The screenshot displays the NodeMCU IDE interface. On the left, a code editor shows a Lua script named 'init.lua'. The script configures a Wi-Fi station, prints the IP address, and sets up a light sensor (ldr) with a timer. A red box highlights the status bar at the bottom of the editor, which shows 'BUSY' and the file path '/Users/inventor/Desktop/ldr/init.lua'. On the right, the serial monitor displays the output of the script, showing the IP address and the results of the sensor readings. The serial monitor also shows a baud rate of 9600 and a 'Send' button.

```
1 wifi.setmode(wifi.STATION)
2 wifi.sta.config("HUAWAI-E5330","fgm3fb9q")
3 print(wifi.sta.getip())
4 tmr.delay(1000000)
5
6 ldr_value=0
7 percent_light=0
8 percent_dark=0
9 bimb=1
10
11 function ldr()
12 ldr_value = adc.read(0)
13 percent_light = ldr_value*(100/1024)
14 percent_dark = 100 - percent_light
15 end
16
17 tmr.alarm(1,5000, 1, function()
18 ldr() bimb=bimb+1
19 if bimb == 5 then bimb = 0 end end) end
```

```
> w = file.writeline
> w([[wifi.setmode(wifi.STATION)]]);
> w([[wifi.sta.config("HUAWAI-E5330","fgm3fb9q")]]);
> w([[print(wifi.sta.getip())]]);
> w([[tmr.delay(1000000)]]);
> w([[ ]]);
> w([[ldr_value=0]]);
> w([[percent_light=0]]);
> w([[percent_dark=0]]);
> w([[bimb=1]]);
> w([[ ]]);
> w([[function ldr()]]);
> w([[ldr_value = adc.read(0)]]);
> w([[percent_light = ldr_value*(100/1024)]]);
```



الحصول على عنوان IP:

بعد الانتهاء من عملية رفع البرنامج سيظهر لنا عنوان IP بالأسفل كما هو موضح في الصورة التالية:

```
> w([[      buf = buf.."<div class=\"col-
> w([[      buf = buf.."</div></div>"]);]])
> w([[ ]]);
> w([[if(_GET.pin == "ON")then]]);
> w([[      gpio.write(led, gpio.HI
> w([[elseif(_GET.pin == "OFF")then]]);
> w([[      gpio.write(led, gpio.LO
> w([[end]]);
> w([[      client:send(buf);]])];
> w([[      client:close();]])];
> w([[collectgarbage();]])];
> w([[end]]);
> w([[end]]);
> file.close();
> dofile("init.lua");
192.168.8.101 255.255.255.0 192.168.8.1
```

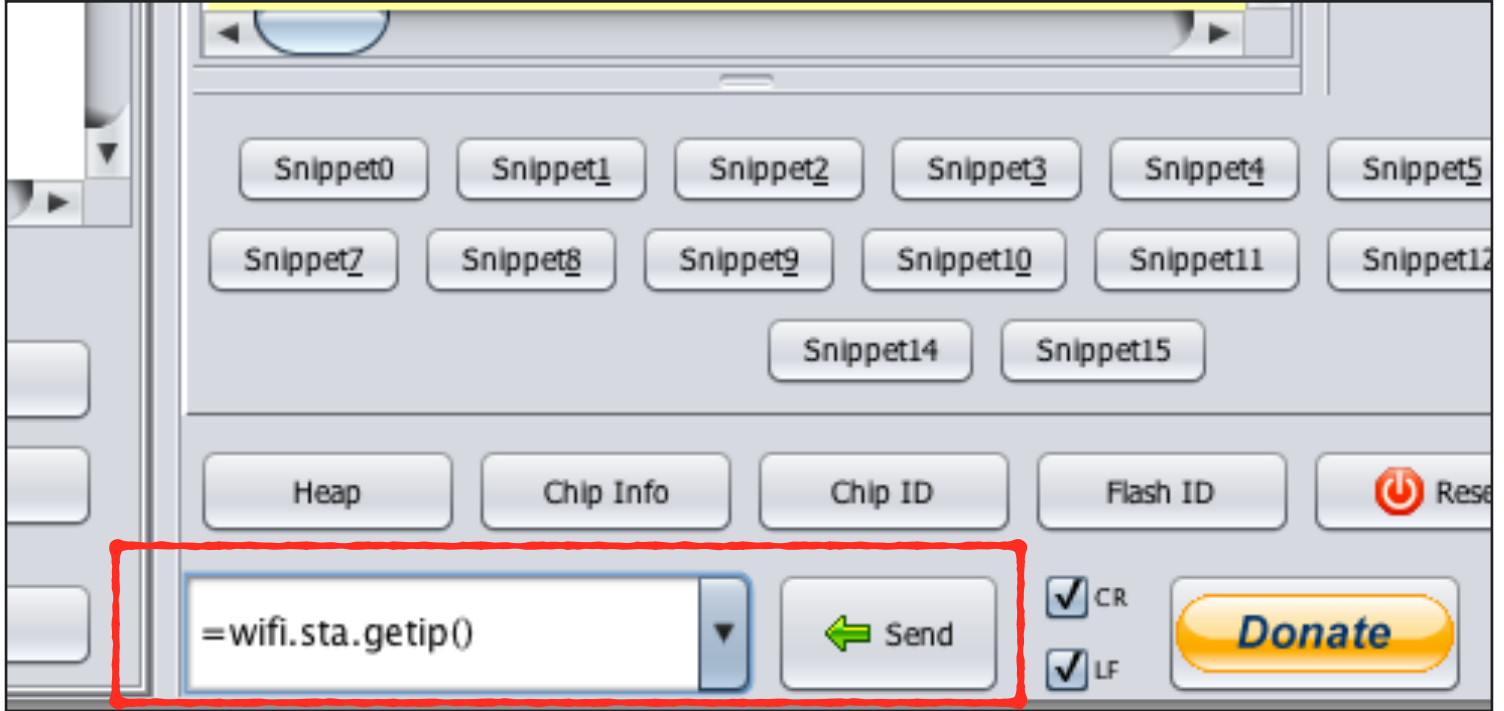
وفي حال أن عنوان IP لم يظهر على الشاشة، فسنقوم بعمل التالي:



NodeMCU



سنقوم بكتابة الأمر `=wifi.sta.getip()` في خانة الإرسال الموضحة في الصورة ثم سنضغط على زر `send` وبعدها سيتم طباعة عنوان IP.



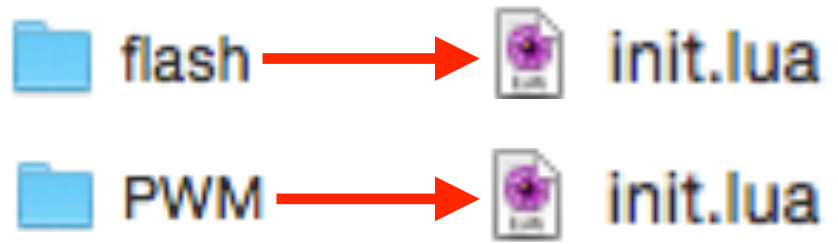
رابط فيديو لمشاهدة تشغيل المشروع:

<https://www.youtube.com/watch?v=Wask4NJp9TQ>



ملاحظة حول تسمية ملفات الشيفرة البرمجية:

حتى يعمل البرنامج المحفوظ على شريحة ESP8266 تلقائياً من غير مشاكل، فإنه يفضل تسمية جميع ملفات الشيفرة البرمجية بإسم `.init.lua`.
وأفضل طريقة هي أن يتم عمل مجلد خاص لكل شيفرة بحيث يتم تسمية المجلد بإسم المشروع الذي نريده وأما ملف الشيفرة فسيكون بإسم `init.lua` والمثال التالي يوضح المعنى:



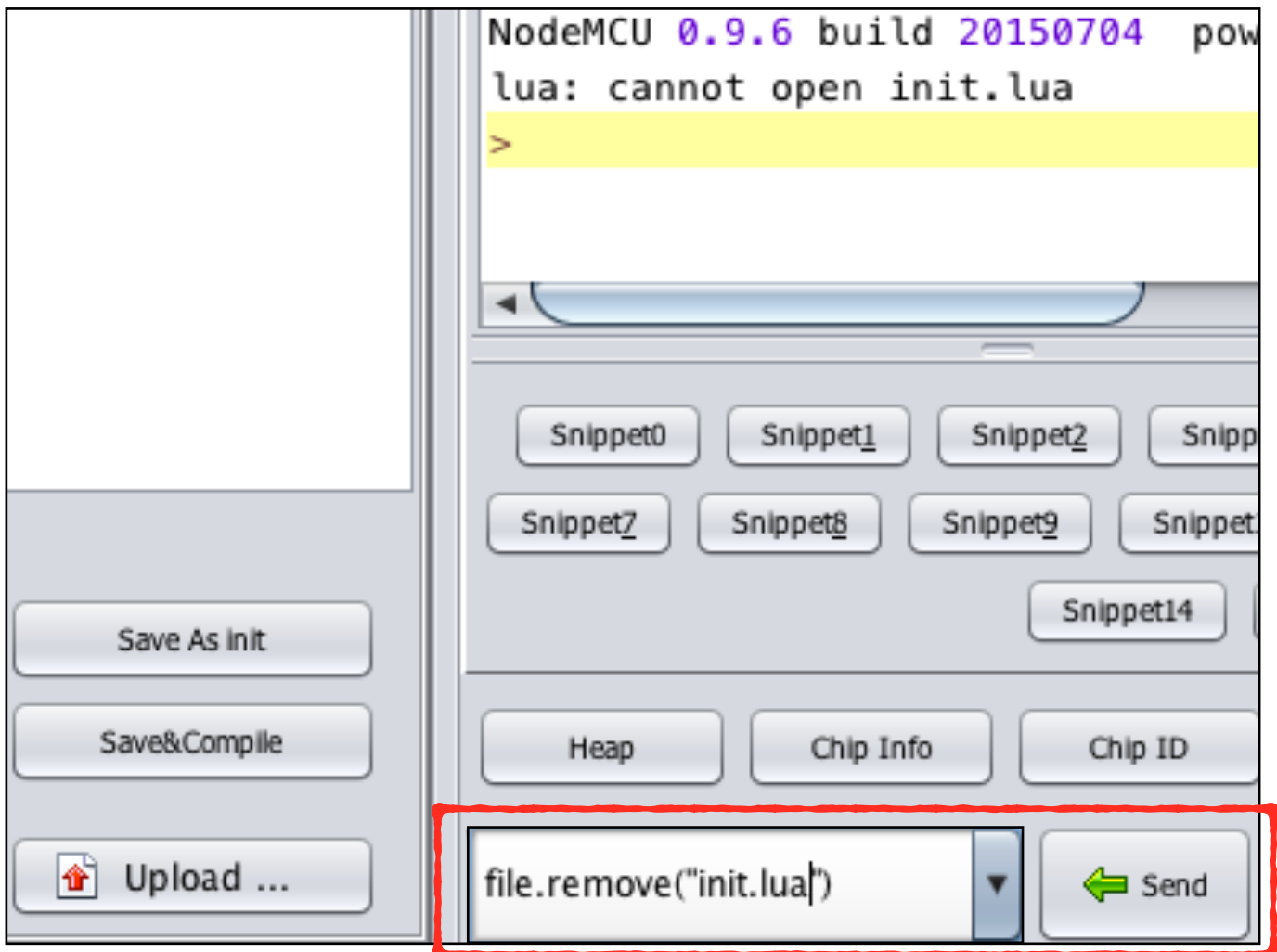
ملاحظات قبل تشغيل المشروع:

- يفضل دائماً إعادة تشغيل لوحة NodeMCU بعد رفع البرنامج الجديد.
لأنه من غير إعادة تشغيل فإنه سيتم تشغيل البرنامج القديم.
- حتى نستطيع الدخول الى صفحة خادم الشبكة يجب ان تكون شبكة الواي فاي (wifi) المتصلة بشريحة ESP8266 هي نفس الشبكة المتصلة بالجهاز الذي نريد التحكم من خلاله.



حذف البرنامج المحفوظ على شريحة ESP8266:

لا يشترط عمل هذه الخطوة كلما أردنا رفع برنامج جديد. فنستطيع رفع برنامج جديد على شريحة ESP8266 دون حذف البرامج القديمة. ولكن إذا أردنا حذف البرنامج نهائياً لتصبح شريحة ESP8266 خالية منه فنقوم بكتابة (`file.remove("init.lua")`) في الخانة الموضحة في الصورة ثم سنضغط على زر `send` وبعدها سيتم حذف البرنامج. أما إذا أردنا حذف جميع البرامج من شريحة ESP8266 فنقوم بكتابة (`file.format()`)





المشروع الخامس (التحكم بالدايود متعدد الألوان):

فكرة المشروع:

سنقوم بعمل واجهة نختار من خلالها اللون الذي نريد إظهاره على الدايود الضوئي

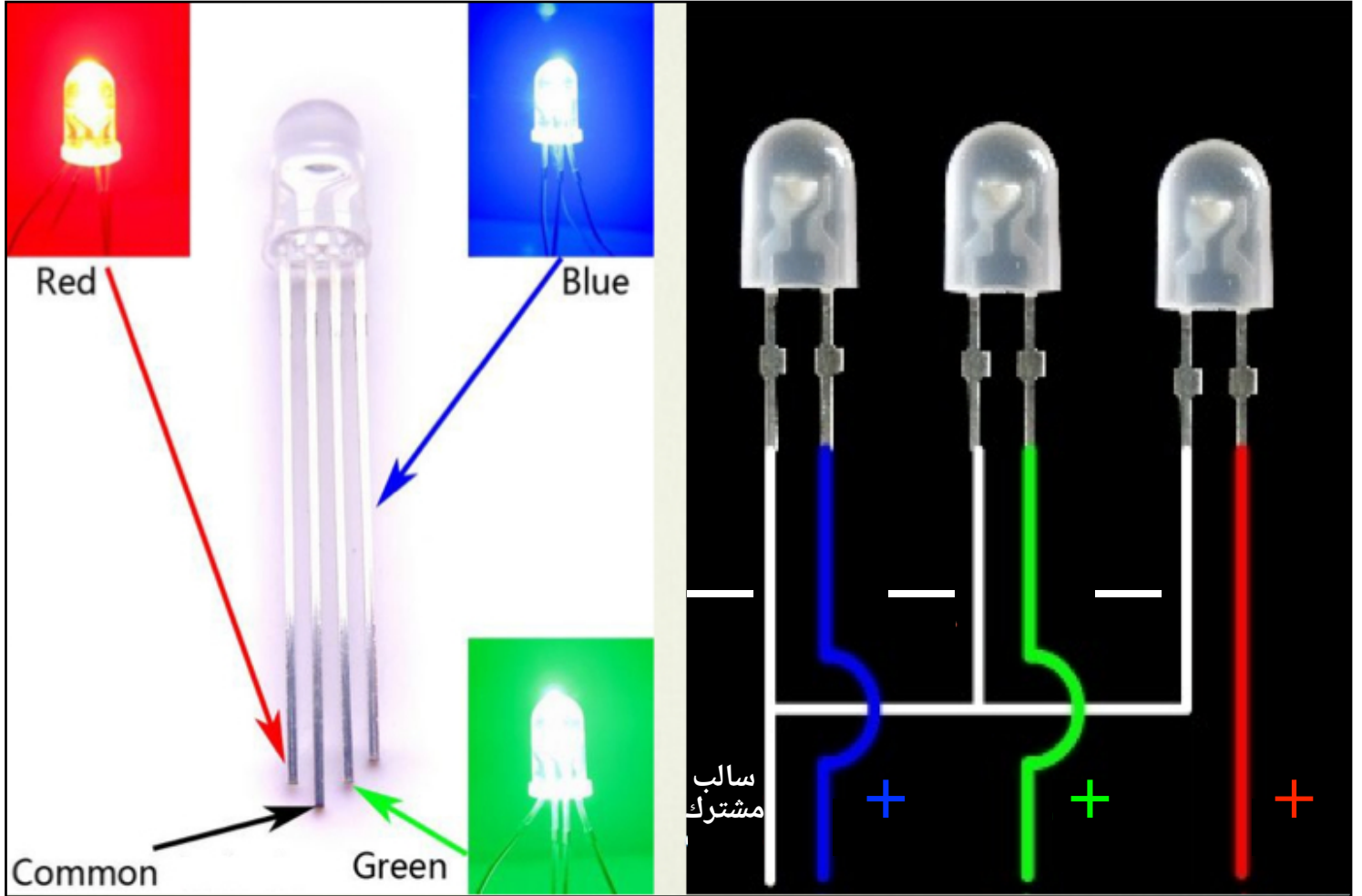
الأدوات المستخدمة:

- لوحة NodeMCU
- حامل لوحة NodeMCU
- دايود ضوئي ثلاثي (RGB LED) من نوع موجب مشترك
- 3 مقاومات 220 (يمكن الإستغناء عنها عند التشغيل على 3.3v)
- لوحة تثبيت القطع الالكترونية (Breadboard)



أنواع الدايود الضوئي الثلاثي:

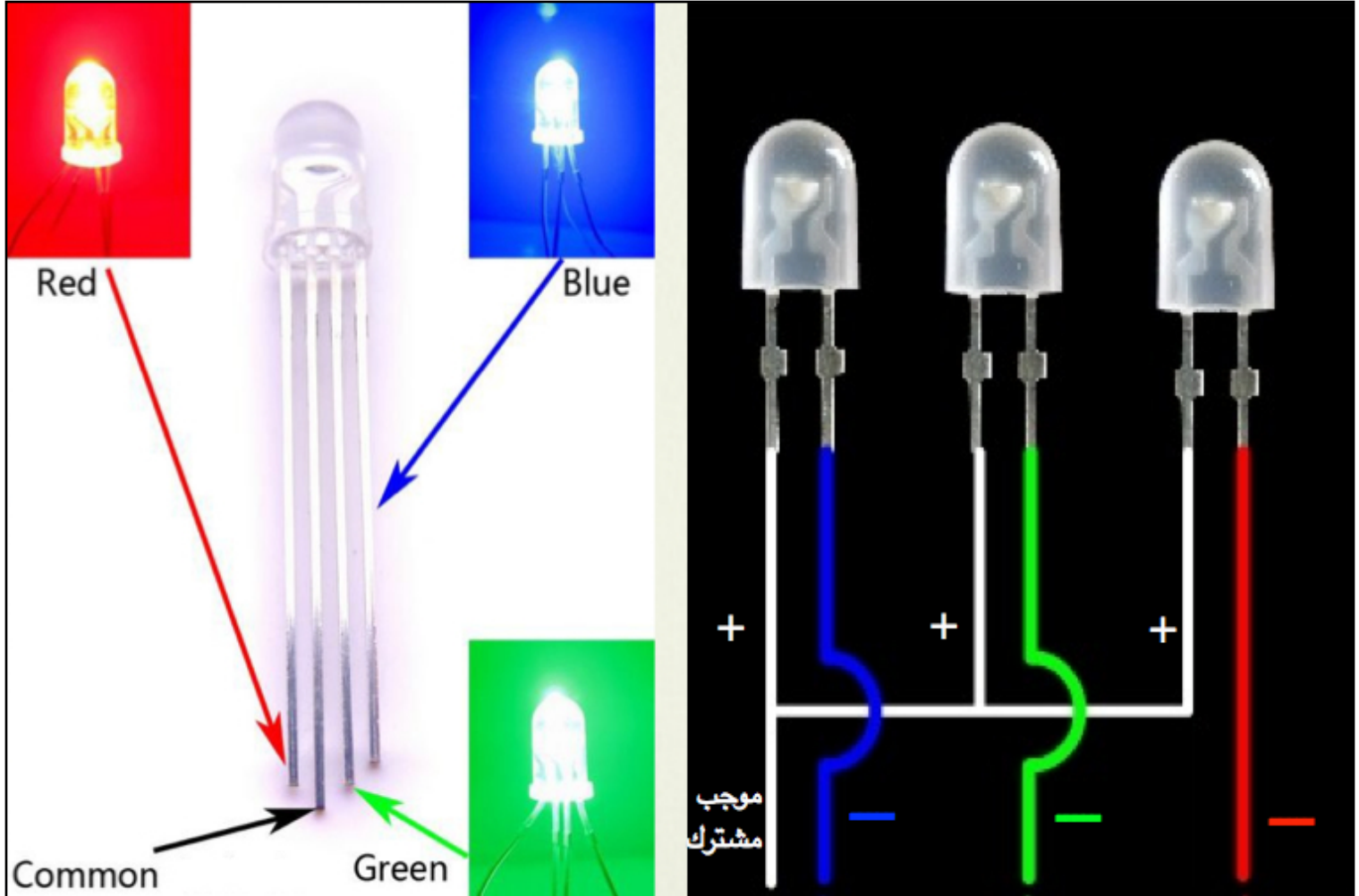
(1) سالب مشترك (common cathode):



نلاحظ من الصورة أن الدايود الثلاثي هو في الأصل عبارة عن 3 دايودات (الأحمر، الأخضر، الأزرق) مجتمعة مع بعضهم في دايود واحد. حيث أن جميع الأطراف السالبة متصلة مع بعضهم في طرف واحد. لذلك هذا النوع يسمى بسالب مشترك. وتعتبر هذه الألوان الثلاثة هي الألوان الأساسية، و نستطيع من خلال دمجها بنسب معينة الحصول على بقية الألوان.



(2) موجب مشترك (common anode):



نلاحظ من الصورة أن الدايود الثلاثي هو في الأصل عبارة عن 3 دايودات (الأحمر، الأخضر، الأزرق) مجتمعة مع بعضهم في دايود واحد. حيث أن جميع الأطراف الموجبة متصلة مع بعضهم في طرف واحد. لذلك هذا النوع يسمى بموجب مشترك. وتعتبر هذه الألوان الثلاثة هي الألوان الأساسية، و نستطيع من خلال دمجها بنسب معينة الحصول على بقية الألوان.



طريقة عمل السالب المشترك (common cathode):

سنقوم بتوصيل الطرف السالب المشترك بالأرضي (GND). وبقية الأطراف سيتم توصيلها بمدخل لوحة NodeMCU. وهذا يعني أن الدايود سيعمل بالقيمة الرقمية **1 (HIGH)** وسينطفئ بالقيمة الرقمية **0 (LOW)**. وسيعمل بالقيم التماثلية من **0** إلى **1023**. حيث أن القيمة **0** تمثل أقل شدة إضاءة والقيمة **1023** تمثل أعلى شدة إضاءة.

طريقة عمل الموجب المشترك (common anode):

سنقوم بتوصيل الطرف الموجب المشترك بالمصدر 5v. وبقية الأطراف سيتم توصيلها بمدخل لوحة NodeMCU. وهذا يعني أن الدايود سيعمل بالقيمة الرقمية **0 (LOW)** وسينطفئ بالقيمة الرقمية **1 (HIGH)**. وسيعمل بالقيم التماثلية من **1023** إلى **0**. حيث أن القيمة **1023** تمثل أقل شدة إضاءة والقيمة **0** تمثل أعلى شدة إضاءة.

ملاحظة بالنسبة للقيم التماثلية:

عدد القيم التماثلية يتحدد حسب عدد البت (bit).

8 بت (8 bit): عدد القيم التماثلية = $2^8 = 256$ (من 0 إلى 255)

10 بت (10 bit): عدد القيم التماثلية = $2^{10} = 1024$ (من 0 إلى 1023)

المدخل في لوحة NodeMCU تتكون من 10 بت (10 bit)



توليد المزيد من الألوان:

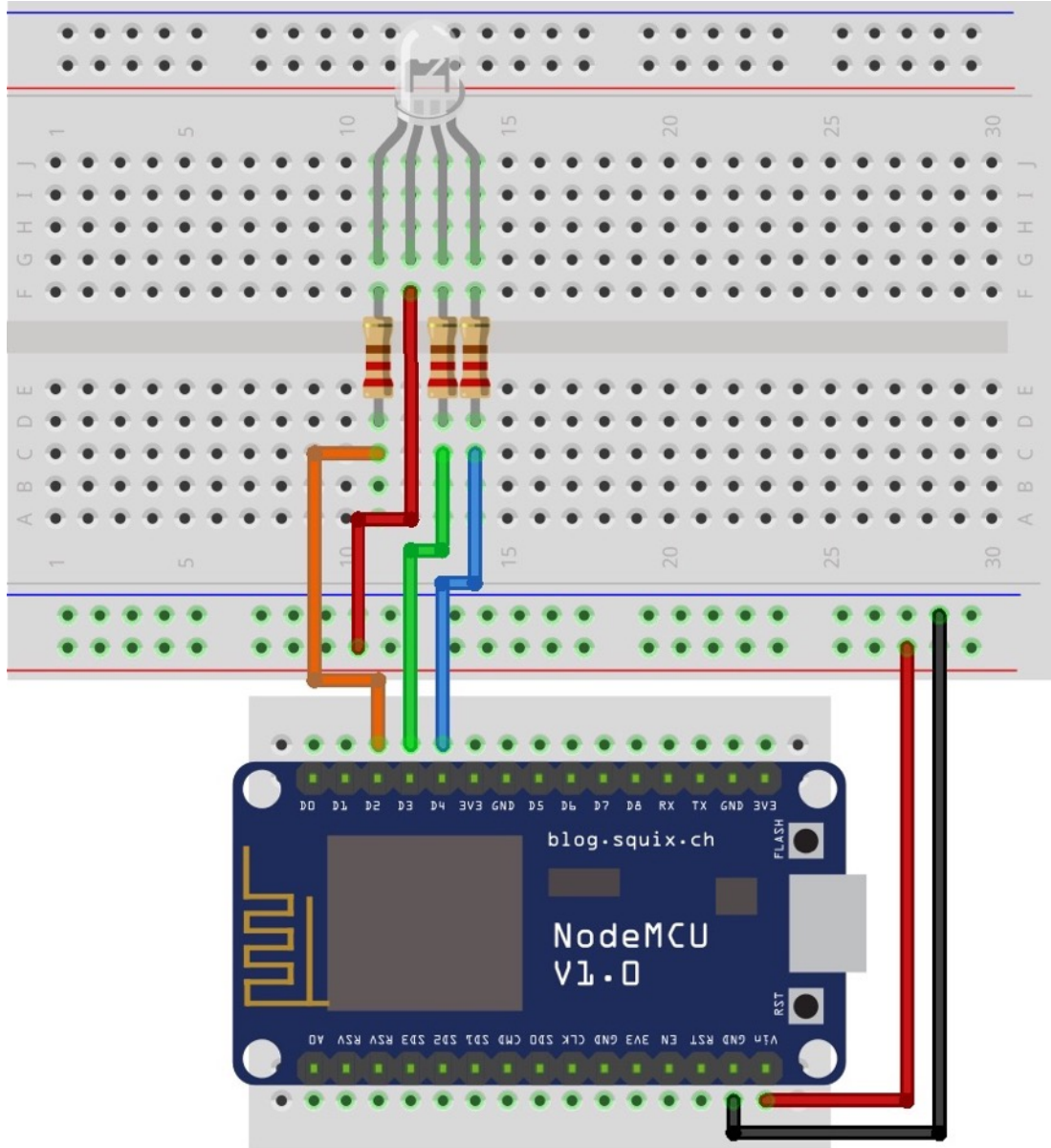
اللون	الرمز					
	سالب مشترك			موجب مشترك		
	الطرف الأحمر (R)	الطرف الأخضر (G)	الطرف الأزرق (B)	الطرف الأحمر (R)	الطرف الأخضر (G)	الطرف الأزرق (B)
	0	0	0	1023	1023	1023
	1023	0	0	0	1023	1023
	0	1023	0	1023	0	1023
	0	0	1023	1023	1023	0
	1023	1023	0	0	0	1023
	1023	590	0	0	433	1023
	0	1023	1023	1023	0	0
	1023	0	1023	0	1023	0
	1023	1023	1023	0	0	0



NodeMCU



الدائرة الالكترونية:



قمنا بتوصيل الطرف الموجب المشترك بالمصدر 5v. والطرف الأحمر متصل في المدخل D2. والطرف الأخضر متصل في المدخل D3. والطرف الأزرق متصل في المدخل D4 من لوحة NodeMCU.



كتابة الشيفرة البرمجية وشرحها:

الشيفرة البرمجية لهذا المشروع موجودة على الرابط التالي:

www.github/rgb.lua

بعد ذلك سنقوم بفتح برنامج ESPlorer لكتابة الشيفرة البرمجية.

```
1 wifi.setmode(wifi.STATION)
2 wifi.sta.config("ssid", "pass")
3 print(wifi.sta.getip())
```

السطر	الشرح
1	إعداد شريحة ESP8266 لإستقبال اشارة الواي فاي (wifi). أي أنه سيتم تشغيلها كعميل (client).
2	للإتصال بشبكة الواي فاي (wifi). وتحتوي على خانتين: ssid: سنكتب اسم الشبكة التي نريد الاتصال بها pass: سنكتب الرقم السري الخاص بالشبكة
3	لطباعة عنوان IP الذي سنستخدمه للدخول الى صفحة خادم الشبكة



```
5  r=2
6  g=3
7  b=4
8
9  gpio.mode(r,gpio.OUTPUT)
10 gpio.mode(g,gpio.OUTPUT)
11 gpio.mode(b,gpio.OUTPUT)
```

السطر	الشرح
5	قمنا بتعريف ثلاث متغيرات: متغير بإسم r (الطرف الأحمر) وهو يشير الى المدخل D2.
7	متغير بإسم g (الطرف الأخضر) وهو يشير الى المدخل D3. متغير بإسم b (الطرف الأزرق) وهو يشير الى المدخل D4.
9	تهيئة طرف الدايود الأحمر كخرج (output)
11	تهيئة طرف الدايود الأخضر كخرج (output) تهيئة طرف الدايود الأزرق كخرج (output)



```
12 pwm.setup(r, 1000, 1023)
13 pwm.setup(g, 1000, 1023)
14 pwm.setup(b, 1000, 1023)
15 pwm.start(r)
16 pwm.start(g)
17 pwm.start(b)
```

السطر	الشرح
12	ضبط خصائص PWM للطرف الأحمر. حيث أن التردد 1000 والقيمة الابتدائية 1023 (هذا يعني أن الدايمود الأحمر سيكون منطفىء)
13	ضبط خصائص PWM للطرف الأخضر. حيث أن التردد 1000 والقيمة الابتدائية 1023 (هذا يعني أن الدايمود الأخضر سيكون منطفىء)
14	ضبط خصائص PWM للطرف الأزرق. حيث أن التردد 1000 والقيمة الابتدائية 1023 (هذا يعني أن الدايمود الأزرق سيكون منطفىء)
15	لبدء تشغيل الدايمود الأحمر بخاصية PWM.
16	لبدء تشغيل الدايمود الأخضر بخاصية PWM.
17	لبدء تشغيل الدايمود الأزرق بخاصية PWM.



```
19 function off()  
20 pwm.setduty(r,1023)  
21 pwm.setduty(g,1023)  
22 pwm.setduty(b,1023)  
23 end  
24  
25 function red()  
26 pwm.setduty(r,0)  
27 pwm.setduty(g,1023)  
28 pwm.setduty(b,1023)  
29 end  
30  
31 function green()  
32 pwm.setduty(r,1023)  
33 pwm.setduty(g,0)  
34 pwm.setduty(b,1023)  
35 end
```

السطر	الشرح
19	قمنا بإنشاء دالة بإسم off() وظيفتها إطفاء الدايود الثلاثي عن طريق
23	تشغيل أطراف الدايود بالقيم المناسبة. (راجع جدول توليد الألوان)
25	قمنا بإنشاء دالة بإسم red() وظيفتها إضاءة الدايود الثلاثي باللون
29	الأحمر عن طريق تشغيل أطراف الدايود بالقيم المناسبة.
31	قمنا بإنشاء دالة بإسم green() وظيفتها إضاءة الدايود الثلاثي باللون
35	الأخضر عن طريق تشغيل أطراف الدايود بالقيم المناسبة.



```
37 function blue()  
38 pwm.setduty(r,1023)  
39 pwm.setduty(g,1023)  
40 pwm.setduty(b,0)  
41 end  
42  
43 function white()  
44 pwm.setduty(r,0)  
45 pwm.setduty(g,0)  
46 pwm.setduty(b,0)  
47 end  
48  
49 function yellow()  
50 pwm.setduty(r,0)  
51 pwm.setduty(g,0)  
52 pwm.setduty(b,1023)  
53 end
```

السطر	الشرح
37	قمنا بإنشاء دالة باسم <code>blue()</code> وظيفتها إضاءة الدايود الثلاثي باللون الأزرق عن طريق تشغيل أطراف الدايود بالقيم المناسبة.
41	
43	قمنا بإنشاء دالة باسم <code>white()</code> وظيفتها إضاءة الدايود الثلاثي باللون الأبيض عن طريق تشغيل أطراف الدايود بالقيم المناسبة.
47	
49	قمنا بإنشاء دالة باسم <code>yellow()</code> وظيفتها إضاءة الدايود الثلاثي باللون الأصفر عن طريق تشغيل أطراف الدايود بالقيم المناسبة.
53	



```
55 function aqua()  
56   pwm.setduty(r,1023)  
57   pwm.setduty(g,0)  
58   pwm.setduty(b,0)  
59 end  
60  
61 function pink()  
62   pwm.setduty(r,0)  
63   pwm.setduty(g,1023)  
64   pwm.setduty(b,0)  
65 end  
66  
67 function orange()  
68   pwm.setduty(r,0)  
69   pwm.setduty(g,433)  
70   pwm.setduty(b,1023)  
71 end
```

السطر	الشرح
55	قمنا بإنشاء دالة بإسم aqua() وظيفتها إضاءة الدايود الثلاثي باللون
59	السماعي عن طريق تشغيل أطراف الدايود بالقيم المناسبة.
61	قمنا بإنشاء دالة بإسم pink() وظيفتها إضاءة الدايود الثلاثي باللون
65	الوردي عن طريق تشغيل أطراف الدايود بالقيم المناسبة.
67	قمنا بإنشاء دالة بإسم orange() وظيفتها إضاءة الدايود الثلاثي باللون
71	البرتقالي عن طريق تشغيل أطراف الدايود بالقيم المناسبة.



```
73  srv=net.createServer(net.TCP)
74  srv:listen(80,function(conn)
```

السطر	الشرح
73	المنفذ 80 (port 80) يستخدم للاتصال بين الخادم و العميل. وسنقوم
74	بإنشاء صفحة خادم الشبكة على هذا المنفذ

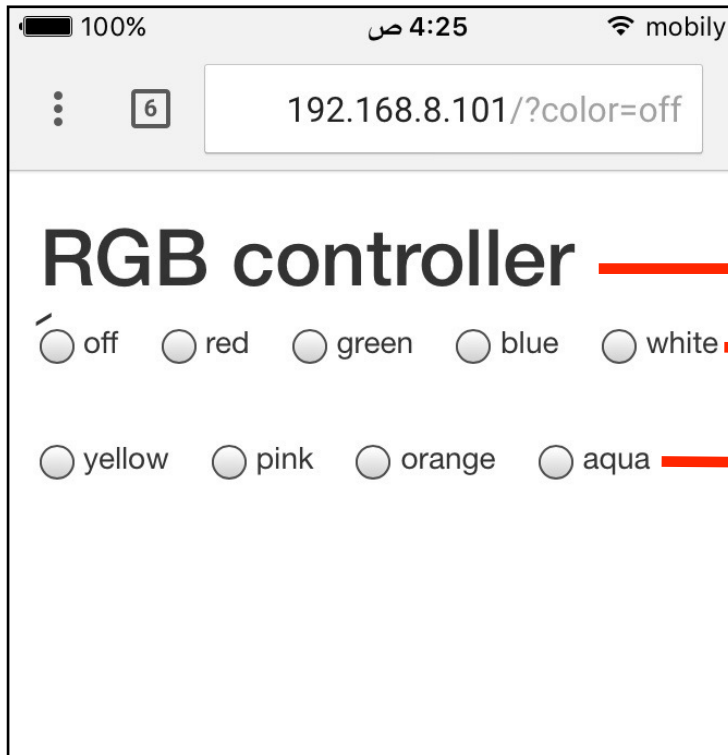
```
75      conn:on("receive", function(client,request)
76
77  local buf = "";
78      buf = buf.."HTTP/1.1 200 OK\r\n\r\n"
79  local _, _, method, path, vars = string.find(request, "([A-Z]+) (.+)?(.+) HTTP");
80  if(method == nil)then
81      _, _, method, path = string.find(request, "([A-Z]+) (.+) HTTP");
82  end
83
84  local _GET = {}
85  if (vars ~= nil)then
86  for k, v in string.gmatch(vars, "(%w+)=(%w+)&*") do
87      _GET[k] = v
88  end
89  end
```

السطر	الشرح
75	داخل هذه الدالة سنقوم بكتابة الاوامر التي تمكننا من تصميم وتنسيق صفحة خادم الشبكة.
77	مجموعة من المتغيرات التي ستقوم بحفظ وتخزين صفحة خادم
89	الشبكة وأيضا حفظ وتخزين رابط الصفحة (URL).



```
90 buf = buf.."<!DOCTYPE HTML>"
91 buf = buf.."<html><head>";
92 buf = buf.."<meta charset=\"utf-8\">";
93 buf = buf.."<meta http-equiv=\"X-UA-Compatible\" content=\"IE=edge\">";
94 buf = buf.."<meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">";
95 buf = buf.."<script src=\"https://code.jquery.com/jquery-2.1.3.min.js\"></script>";
96 buf = buf.."<link rel=\"stylesheet\" href=\"https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css\">";
97 buf = buf.."</head><div class=\"container\">";
98 buf = buf.."<h1>RGB controller</h1>";
99 buf = buf.."<form role=\"form\">";
100 buf = buf.."<div class=\"radio\">";
101 buf = buf.."<label><input type=\"radio\" name=\"color\" value=\"off\" onclick=\"this.checked = !this.checked\">";
102 buf = buf.."<label><input type=\"radio\" name=\"color\" value=\"red\" onclick=\"this.checked = !this.checked\">";
103 buf = buf.."<label><input type=\"radio\" name=\"color\" value=\"green\" onclick=\"this.checked = !this.checked\">";
104 buf = buf.."<label><input type=\"radio\" name=\"color\" value=\"blue\" onclick=\"this.checked = !this.checked\">";
105 buf = buf.."<label><input type=\"radio\" name=\"color\" value=\"white\" onclick=\"this.checked = !this.checked\">";
106 buf = buf.."</label></div>";
107 buf = buf.."<div class=\"radio\">";
108 buf = buf.."<label><input type=\"radio\" name=\"color\" value=\"yellow\" onclick=\"this.checked = !this.checked\">";
109 buf = buf.."<label><input type=\"radio\" name=\"color\" value=\"pink\" onclick=\"this.checked = !this.checked\">";
110 buf = buf.."<label><input type=\"radio\" name=\"color\" value=\"orange\" onclick=\"this.checked = !this.checked\">";
111 buf = buf.."<label><input type=\"radio\" name=\"color\" value=\"aqua\" onclick=\"this.checked = !this.checked\">";
112 buf = buf.."</label></div>";
113 buf = buf.."</form>";
114 buf = buf.."</div>";
115 buf = buf.."</html>"
```

السطر	الشرح
90	هذه الأوامر هي التي تستخدم لإنشاء الصفحات الالكترونية باستخدام
115	البوتستراب (Bootstrap) وقد تحدثنا عنها في الفصل السابق (صفحة خادم الشبكة).



السطر 98 من الشيفرة البرمجية

الأسطر من 100 إلى 106 من الشيفرة البرمجية

الأسطر من 107 إلى 112 من الشيفرة البرمجية

نلاحظ أن عند الضغط على أي خيار فإن قيمة المتغير `color` الموجود في رابط الصفحة تتغير حسب الخيار. فمثلا:

عند الضغط على الخيار `off`، فإن قيمة المتغير `color` ستصبح `off`

`192.168.8.101/?color=off`

عند الضغط على الخيار `pink`، فإن قيمة المتغير `color` ستصبح `pink`

`192.168.8.101/?color=pink`



```
117 if(_GET.color == "off")then
118     off()
119 elseif(_GET.color == "red")then
120     red()
121 elseif(_GET.color == "green")then
122     green()
123 elseif(_GET.color == "blue")then
124     blue()
125 elseif(_GET.color == "white")then
126     white()
```

السطر	الشرح
117	سيختبر هل أصبحت قيمة المتغير color بـ off أو لا. فإذا كانت هي
118	فسيتم تنفيذ الدالة off().
119	سيختبر هل أصبحت قيمة المتغير color بـ red أو لا. فإذا كانت هي
120	فسيتم تنفيذ الدالة red().
121	سيختبر هل أصبحت قيمة المتغير color بـ green أو لا. فإذا كانت
122	هي فسيتم تنفيذ الدالة green().
123	سيختبر هل أصبحت قيمة المتغير color بـ blue أو لا. فإذا كانت هي
124	فسيتم تنفيذ الدالة blue().
125	سيختبر هل أصبحت قيمة المتغير color بـ white أو لا. فإذا كانت
126	هي فسيتم تنفيذ الدالة white().



```
127 elseif(_GET.color == "yellow")then
128     yellow()
129 elseif(_GET.color == "pink")then
130     pink()
131 elseif(_GET.color == "orange")then
132     orange()
133 elseif(_GET.color == "aqua")then
134     aqua()
135 end
```

السطر	الشرح
127	سيختبر هل أصبحت قيمة المتغير color بـ yellow أو لا. فإذا كانت
128	هي فسيتم تنفيذ الدالة yellow().
129	سيختبر هل أصبحت قيمة المتغير color بـ pink أو لا. فإذا كانت هي
130	فسيتم تنفيذ الدالة pink().
131	سيختبر هل أصبحت قيمة المتغير color بـ orange أو لا. فإذا كانت
132	هي فسيتم تنفيذ الدالة orange().
133	سيختبر هل أصبحت قيمة المتغير color بـ aqua أو لا. فإذا كانت هي
134	فسيتم تنفيذ الدالة aqua().
135	نهاية الدالة الشرطية if.



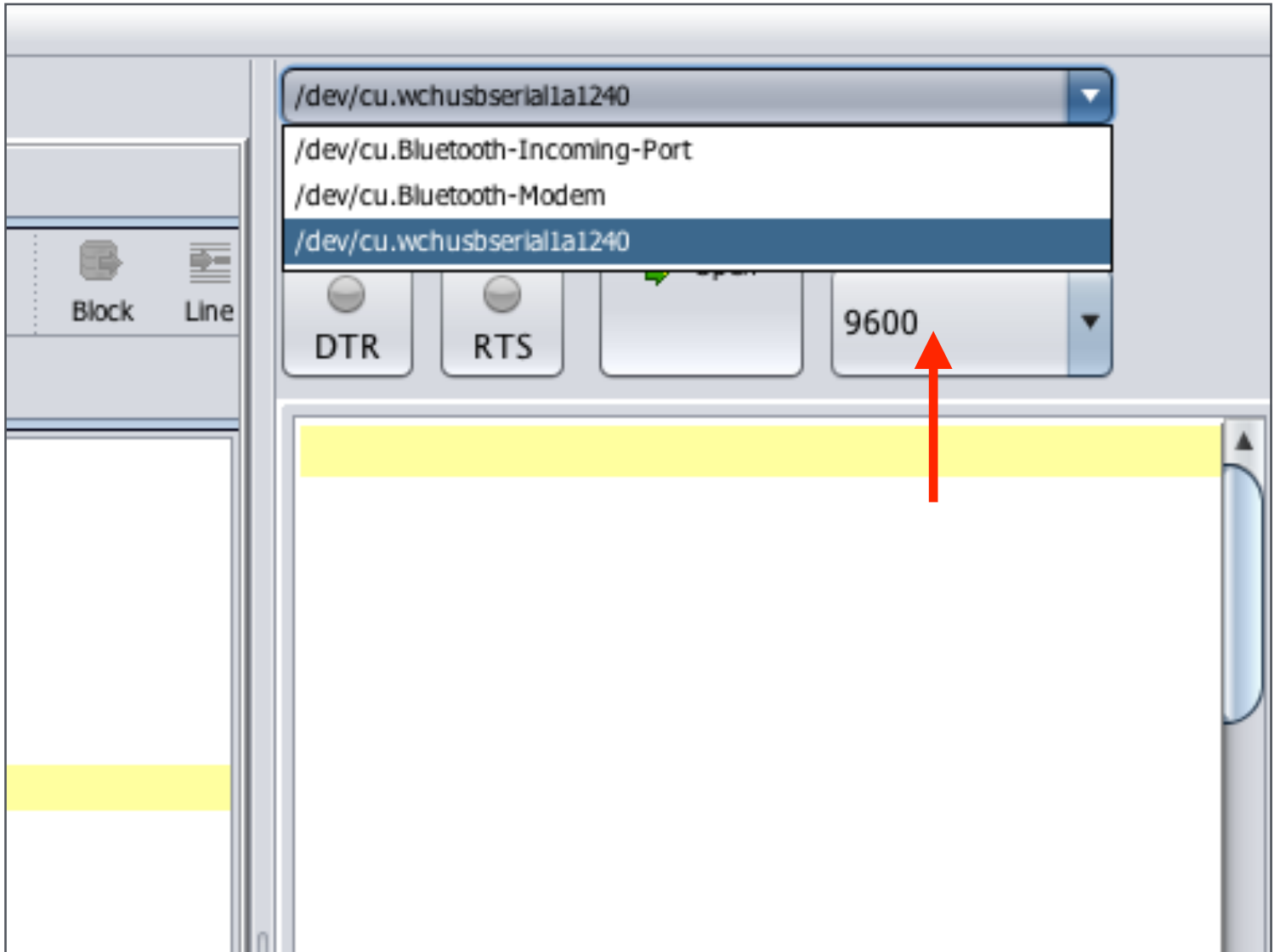
```
137     client:send(buf)
138     client:close()
139
140     collectgarbage()
141 end)
142 end)
```

السطر	الشرح
137	عرض محتويات صفحة خادم الشبكة التي قمنا بإنشائها باستخدام Bootstrap.
138	اغلق اتصال العميل
140	هذه الدالة تسمى بجامع النفايات حيث تقوم بالبحث عن الموارد والبيانات الغير مستخدمة في الذاكرة ومن ثم إزالتها من أجل الحصول على مساحة فارغة في الذاكرة لإستخدامها لأغراض أخرى
141	نهاية الدالة <code>conn:on("receive",function(client,request)</code>
142	نهاية الدالة <code>srv:listen(80,function(conn)</code>



رفع البرنامج على لوحة NodeMCU وتشغيل المشروع:

بعد الانتهاء من كتابة الشيفرة البرمجية سنقوم بتوصيل لوحة NodeMCU بالحاسب. وبعدها سنقوم بإختيار المنفذ المتصل بلوحة NodeMCU. وأيضا سنختار 9600 كسرعة نقل البيانات كما هو موضح في الصورة التالية:



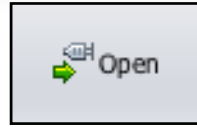
إن لم تجد المنفذ ضمن الخيارات فعليك بالضغط على زر التحديث وبعدها سيظهر معك المنفذ بإذن الله.



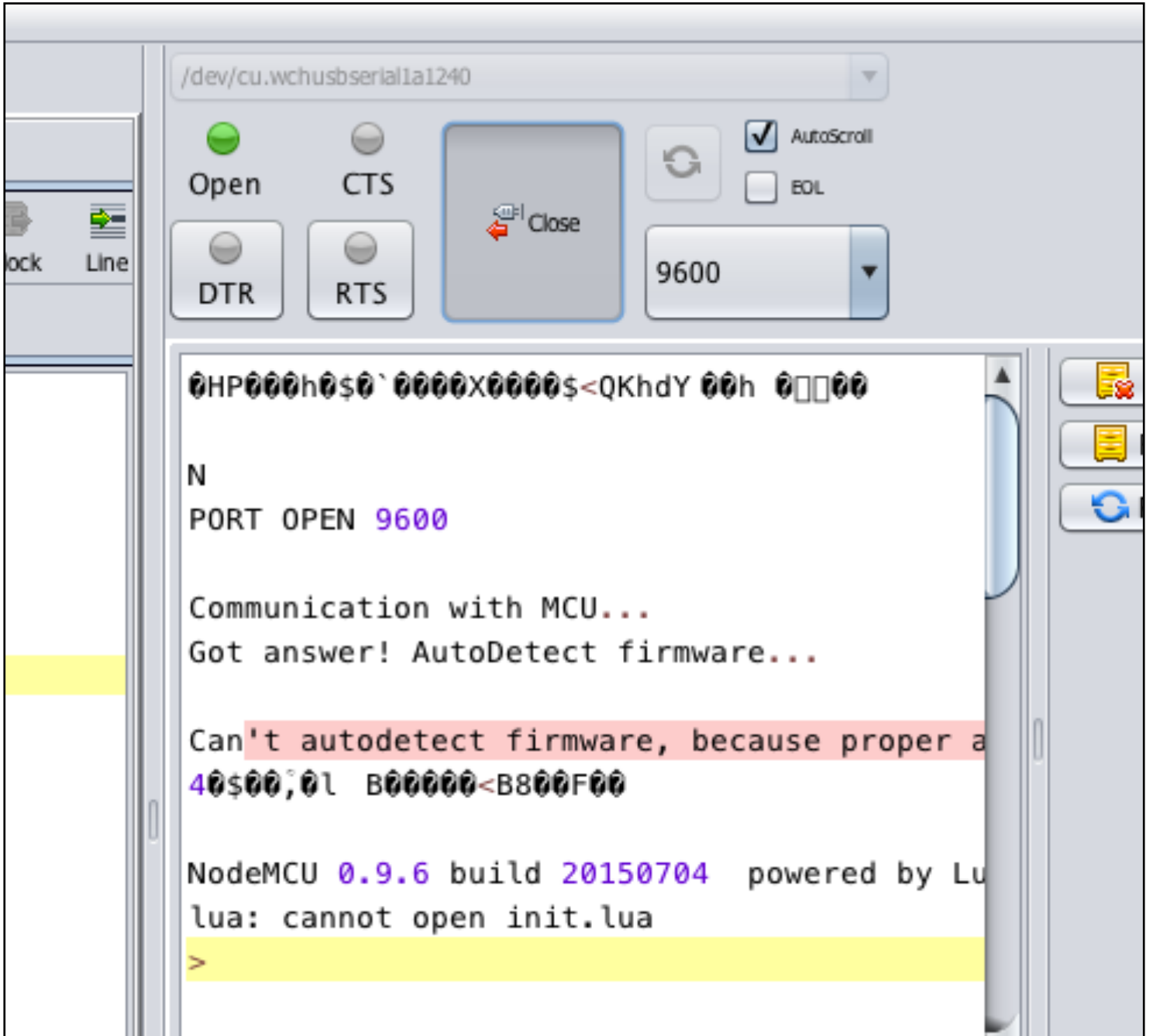
NodeMCU

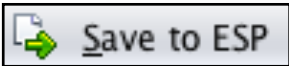


وسيدأ عملية الاتصال كما



ثم سنضغط على زر تفعيل الاتصال
هو موضح في الصورة التالية:





بعد ذلك سنرفع البرنامج بالضغط على زر (save to ESP)



أو بالضغط على زر (send to ESP)

والفرق بين الاثنين هو:

- زر (save to ESP): يقوم بحفظ البرنامج داخل شريحة ESP8266 فحتى عند فصل اللوحة من الحاسب وتشغيلها باستخدام مصدر خارجي للتغذية (بطاريات) فإن المشروع سيعمل لأن البرنامج محفوظ داخل شريحة ESP8266.

- زر (send to ESP): يقوم فقط بقراءة البرنامج دون حفظه ولا يمكن تشغيل المشروع عند فصل اللوحة عن الحاسب وإستخدام مصدر خارجي للتغذية (بطاريات) لأن البرنامج غير محفوظ داخل شريحة ESP8266.

سأقوم بالضغط على زر (save to ESP) و سيطلب منا أن نقوم بحفظ الشيفرة البرمجية بإسم وسوف أسميه بـ `init.lua` كما هو موضح في الصورة التالية:

File Name:

Files of Type:

Save Cancel



NodeMCU



بعد الانتهاء من حفظ الشيفرة بإسم سيتم رفع البرنامج كما هو موضح في الصورة التالية:

The screenshot shows the NodeMCU IDE interface. On the left, a Lua script named 'init.lua' is displayed with the following code:

```
3 print(wifi.sta.getip())
4
5 r=2
6 g=3
7 b=4
8
9 gpio.mode(r,gpio.OUTPUT)
10 gpio.mode(g,gpio.OUTPUT)
11 gpio.mode(b,gpio.OUTPUT)
12 pwm.setup(r,1000,0)
13 pwm.setup(g,1000,0)
14 pwm.setup(b,1000,0)
15 pwm.start(r)
16 pwm.start(g)
17 pwm.start(b)
18
19 function off()
20   pwm.setduty(r,1023)
21   pwm.setduty(g,1023)
```

The script is being executed, and the progress bar shows 31% completion. A red box highlights the 'BUSY' status and the file path: `/Users/inventor/Desktop/rob4/init.lua`. On the right, the serial terminal shows the output of the script, including the IP address and the execution of various PWM functions. The terminal also shows the command `=node.heap()` and the output `not enough memory`.

```
> dofile("init.lua");
not enough memory
```

ملاحظة: بعد الإنتهاء من عملية الرفع قد تظهر لنا عبارة `not enough memory`

والمقصود بها أنه تم الرفع بنجاح. ولكن أيضا تم إستهلاك معظم مساحة الذاكرة



الحصول على عنوان IP:

بعد الانتهاء من عملية رفع البرنامج سيظهر لنا عنوان IP بالأسفل كما هو موضح في الصورة التالية:

The screenshot shows the NodeMCU web interface. The main content area displays the following Lua code:

```
> w([[          buf = buf.."<div class=\"col-
> w([[          buf = buf.."</div></div>"]);]])
> w([[ ]]);
> w([[if(_GET.pin == "ON")then]]);
> w([[          gpio.write(led, gpio.HI
> w([[elseif(_GET.pin == "OFF")then]]);
> w([[          gpio.write(led, gpio.LO
> w([[end]]);
> w([[          client:send(buf);]])];
> w([[          client:close();]])];
> w([[collectgarbage();]])];
> w([[end]]);
> w([[end]]);
> file.close();
> dofile("init.lua");
192.168.8.101 255.255.255.0 192.168.8.1
```

The IP address **192.168.8.101** is highlighted in a red box, and a red arrow points to it from below. The interface also shows a toolbar with buttons for Copy, Paste, Block, Line, Open, CTS, DTR, RTS, and a baud rate of 9600. On the right, there are buttons for Format, FS Info, and Reload. At the bottom, there are snippet buttons labeled Snippet0 through Snippet15.

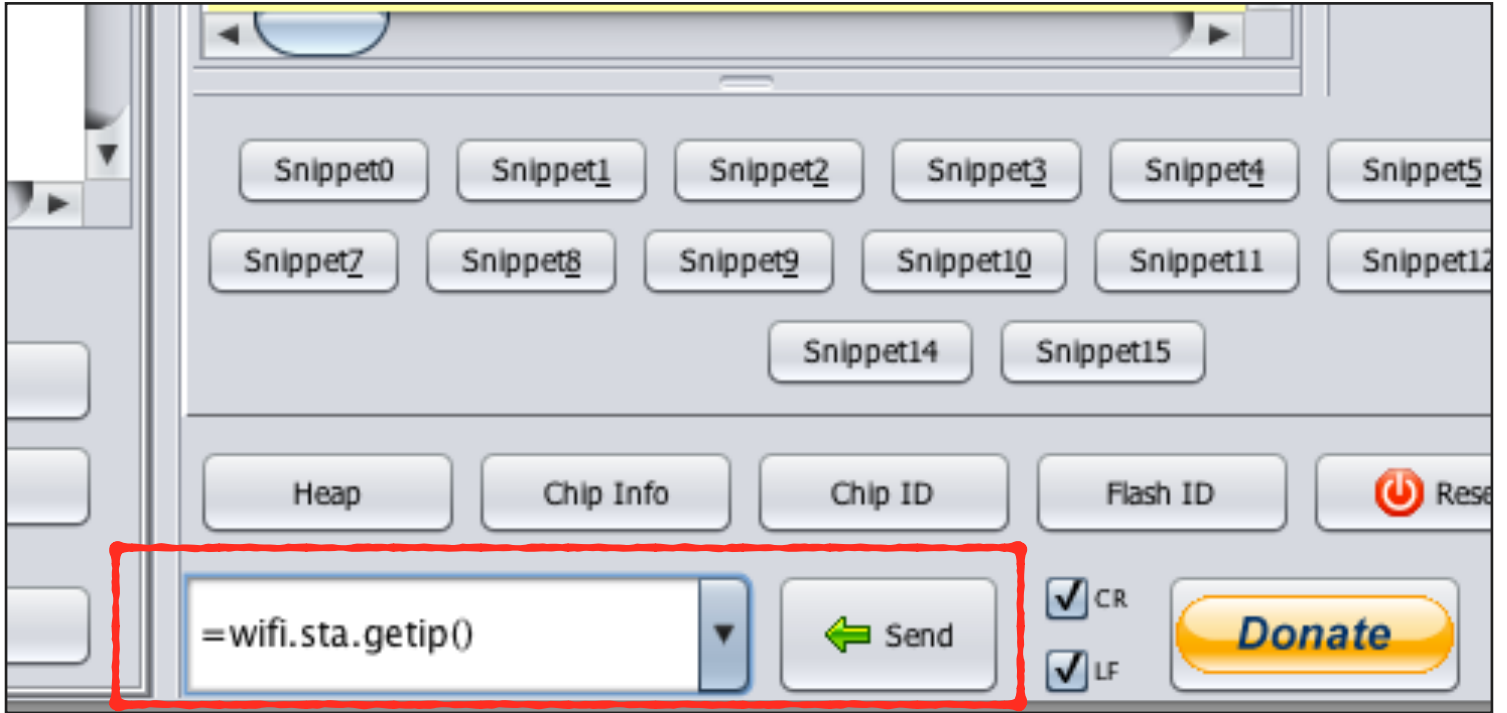
وفي حال أن عنوان IP لم يظهر على الشاشة، فسنقوم بعمل التالي:



NodeMCU



سنقوم بكتابة الأمر `=wifi.sta.getip()` في خانة الإرسال الموضحة في الصورة ثم سنضغط على زر `send` وبعدها سيتم طباعة عنوان IP.



رابط فيديو لمشاهدة تشغيل المشروع:

<https://www.youtube.com/watch?v=8dcl9n4ELhY>



ملاحظة حول تسمية ملفات الشيفرة البرمجية:

حتى يعمل البرنامج المحفوظ على شريحة ESP8266 تلقائياً من غير مشاكل، فإنه يفضل تسمية جميع ملفات الشيفرة البرمجية بإسم `.init.lua`.
وأفضل طريقة هي أن يتم عمل مجلد خاص لكل شيفرة بحيث يتم تسمية المجلد بإسم المشروع الذي نريده وأما ملف الشيفرة فسيكون بإسم `init.lua` والمثال التالي يوضح المعنى:



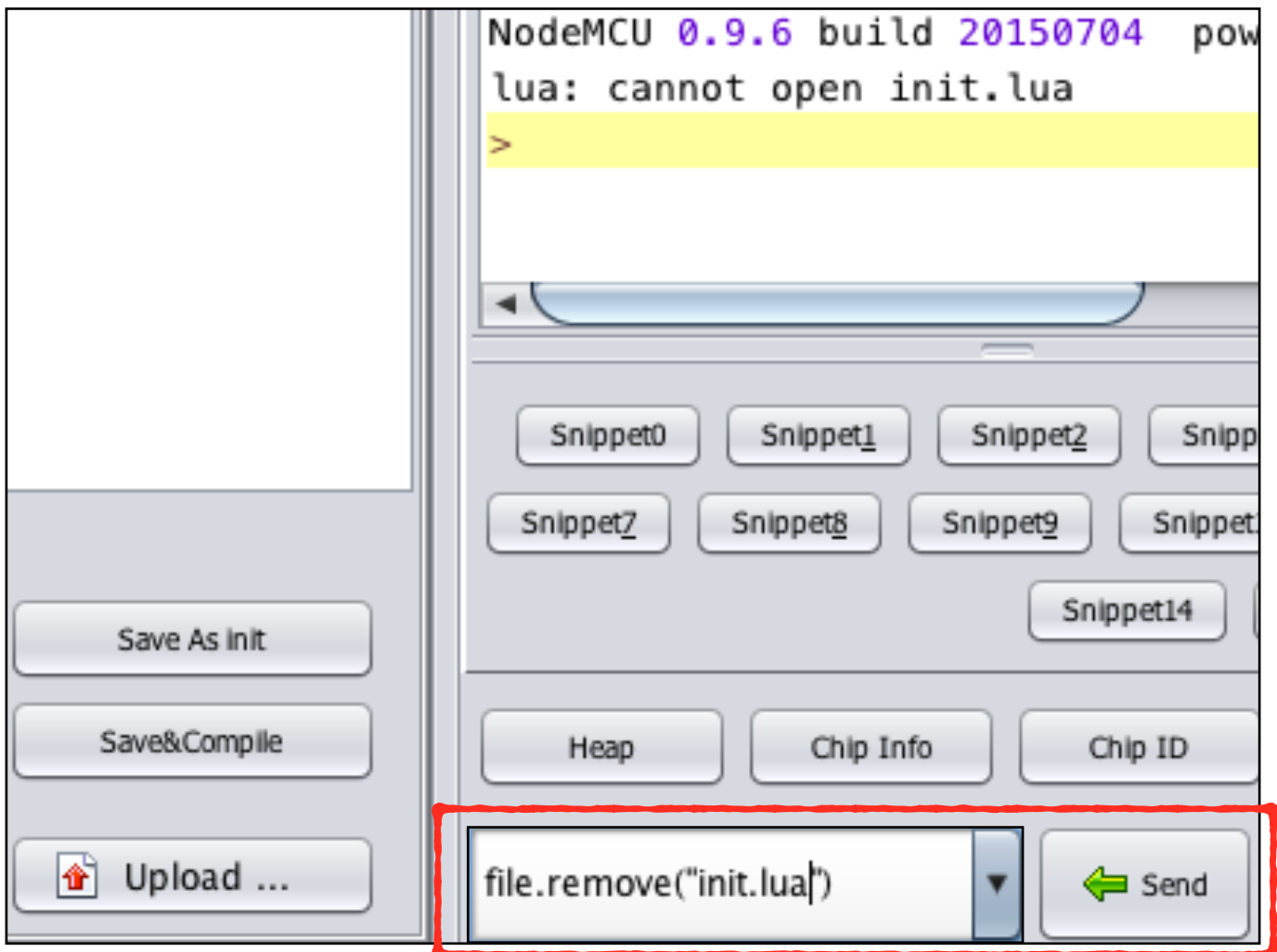
ملاحظات قبل تشغيل المشروع:

- يفضل دائماً إعادة تشغيل لوحة NodeMCU بعد رفع البرنامج الجديد.
لأنه من غير إعادة تشغيل فإنه سيتم تشغيل البرنامج القديم.
- حتى نستطيع الدخول الى صفحة خادم الشبكة يجب ان تكون شبكة الواي فاي (wifi) المتصلة بشريحة ESP8266 هي نفس الشبكة المتصلة بالجهاز الذي نريد التحكم من خلاله.



حذف البرنامج المحفوظ على شريحة ESP8266:

لا يشترط عمل هذه الخطوة كلما أردنا رفع برنامج جديد. فنستطيع رفع برنامج جديد على شريحة ESP8266 دون حذف البرامج القديمة. ولكن إذا أردنا حذف البرنامج نهائياً لتصبح شريحة ESP8266 خالية منه فنقوم بكتابة (`file.remove("init.lua")`) في الخانة الموضحة في الصورة ثم سنضغط على زر `send` وبعدها سيتم حذف البرنامج. أما إذا أردنا حذف جميع البرامج من شريحة ESP8266 فنقوم بكتابة (`file.format()`)





المشروع السادس (تنبيه البريد الإلكتروني):

فكرة المشروع:

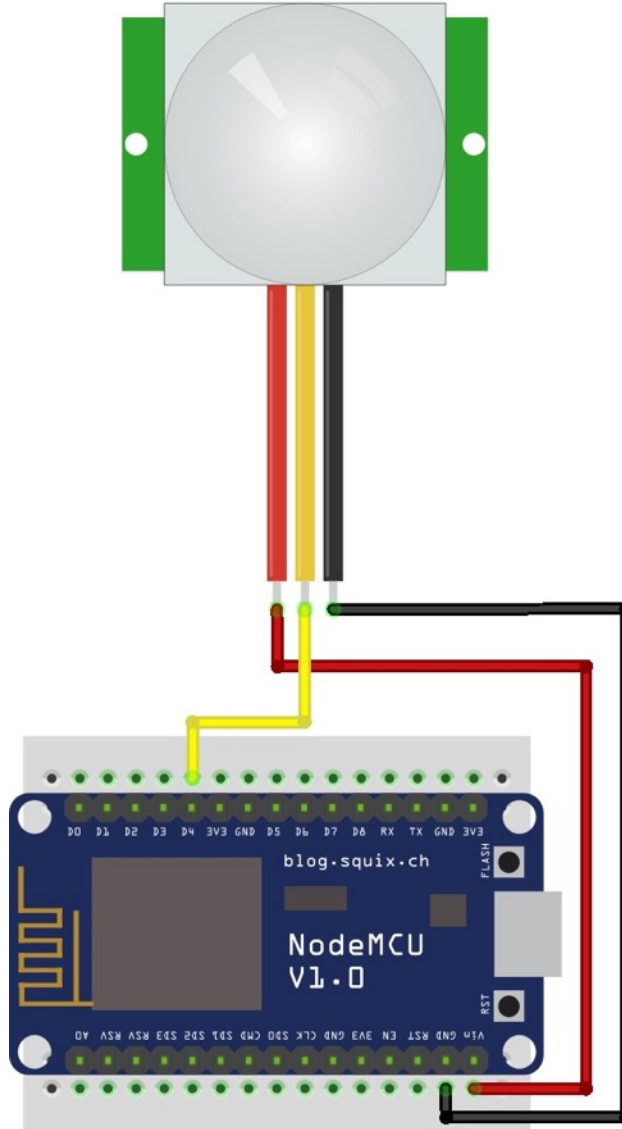
سنستخدم حساس كاشف حركة الاجسام لمعرفة ما إذا كان هناك شخص او جسم قد مر من المنطقة الموجود بها الحساس من خلال رسائل تنبيه بالبريد الالكتروني.

الأدوات المستخدمة:

- لوحة NodeMCU
- حامل لوحة NodeMCU
- حساس كاشف حركة الاجسام (PIR: Passive infrared sensor)
- لوحة تثبيت القطع الالكترونية (Breadboard)



الدائرة الالكترونية:



قمنا بتوصيل الطرف الموجب في 5v. والطرف السالب متصل بالارضي (GND). والطرف الخرج (output) متصل في المدخل D4 في لوحة NodeMCU.

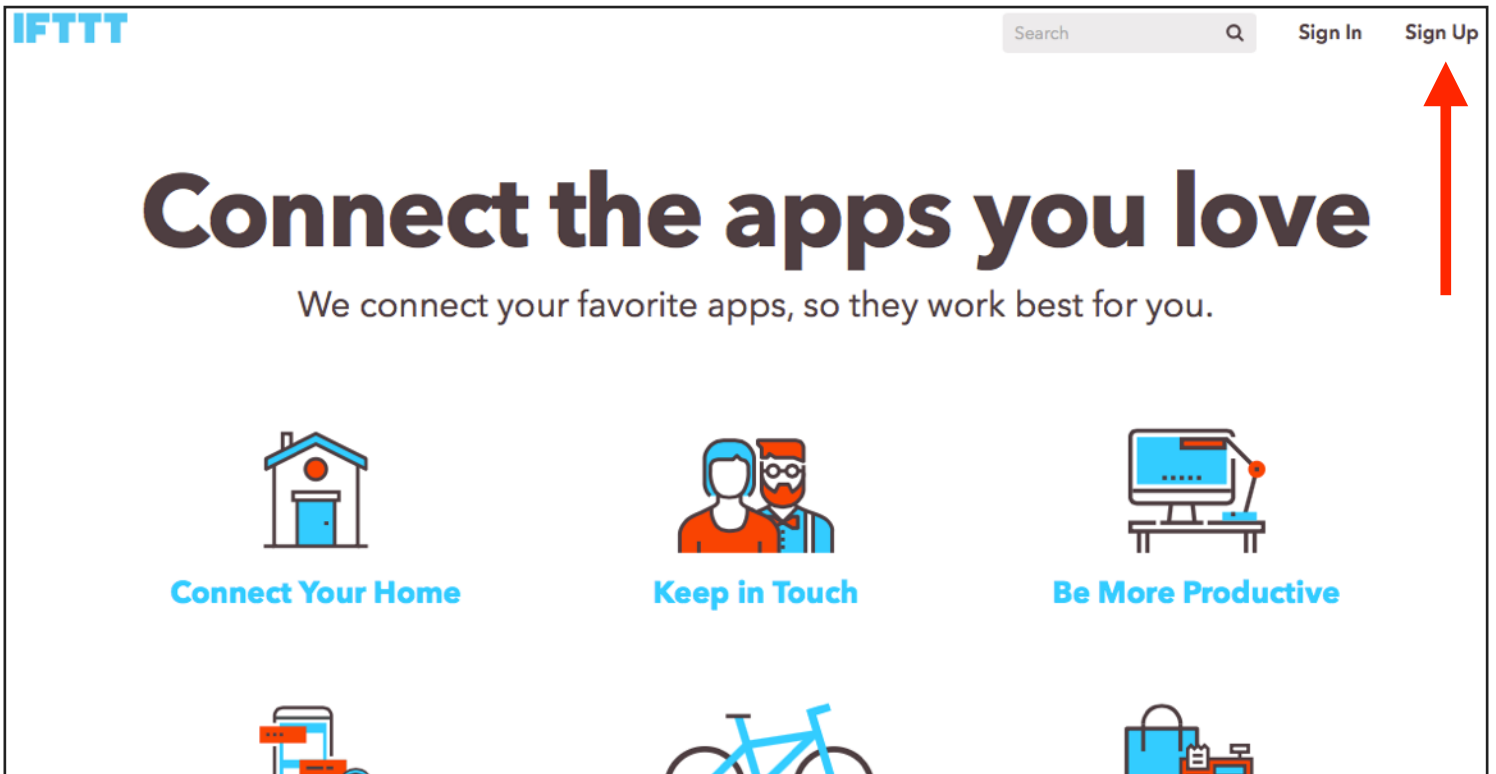


إنشاء حساب في موقع IFTTT (If This Then That):

هذا الموقع يقدم العديد من أدوات التحكم الذكي والتي تسمى بـ (Recipe) ومن ضمنها أداة إرسال تنبيهات بالبريد الإلكتروني التي سنستخدمها في هذا المشروع.

سنقوم بالدخول على الموقع من خلال الرابط التالي:

<https://ifttt.com>



بعد ذلك سنضغط على عبارة (Sign Up) الموجودة في الركن الأيمن العلوي وستظهر لنا صفحة التسجيل في الموقع كما هو موضح في الصورة التالية:



IFTTT



Browse Recipes

Sign in

Create your free account

You're only seconds away from doing more with the products you love.

Your Email

Choose a Password

Create account

في هذه الصفحة سنقوم بإدخال البريد الإلكتروني والرقم السري ثم بعد ذلك سنضغط على (Create account). وبعدها ستظهر لنا الصفحة التالية:



Make powerful connections with a simple phrase:

if this then that

Click this to get started.



من هذه الصفحة سنقوم بالضغط على عبارة (this). وبعدها ستظهر لنا الصفحة التالية:

if  then that

Tomorrow's
forecast calls
for rain

من هذه الصفحة سنقوم بالضغط على عبارة (that). وبعدها ستظهر لنا الصفحة التالية:



if  **then** 

Tomorrow's forecast calls for rain

Send me an email

Continue

سنضغط على (Continue). وبعدها ستظهر لنا الصفحة التالية:

RECIPE

if  **then** 

Get an email if there will be rain in your area tomorrow

by [solomakhin](#) 86k 2.9k

TRIGGER CHANNEL

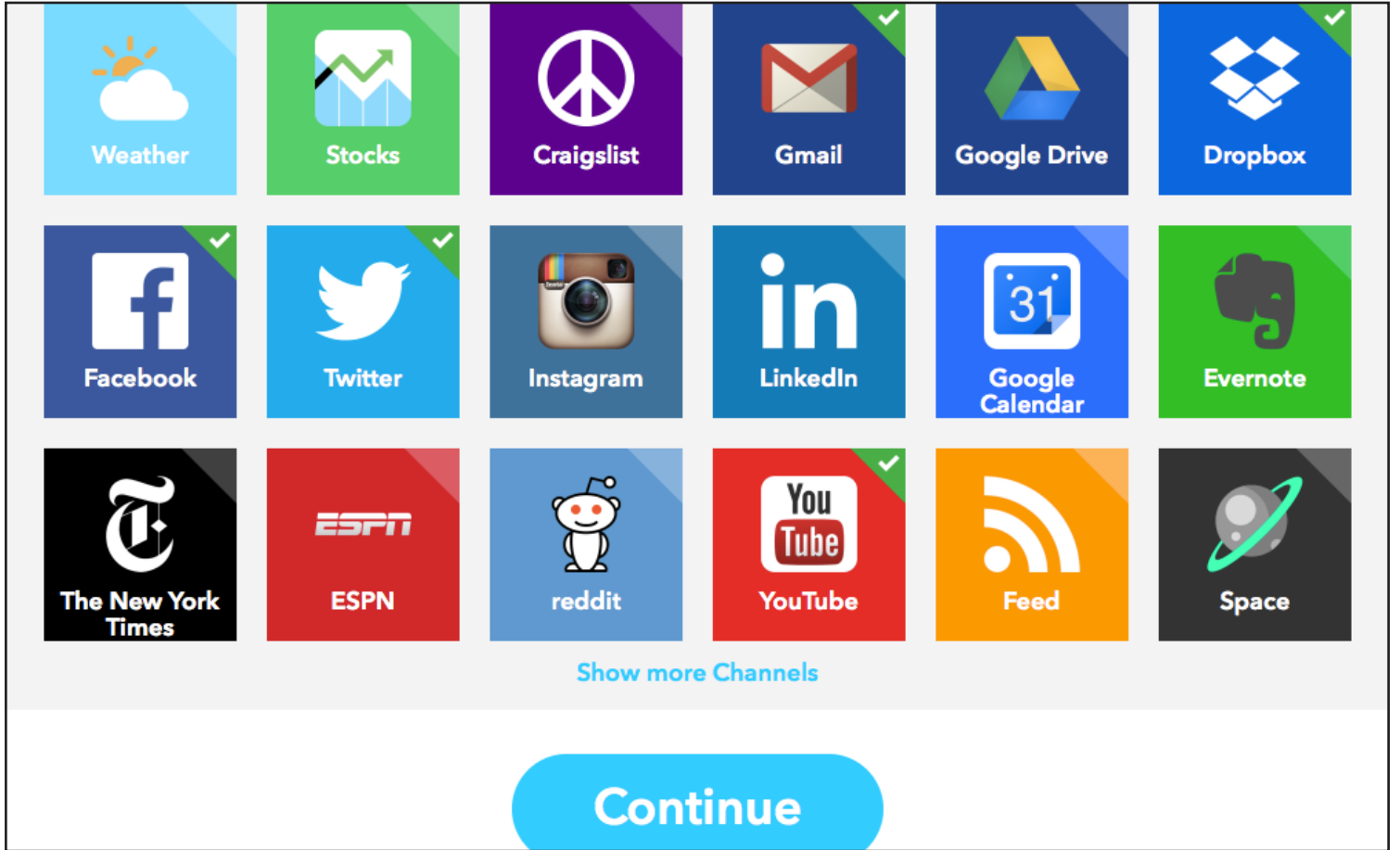
ACTION CHANNEL

Continue

سنضغط على (Continue). وبعدها ستظهر لنا الصفحة التالية:



NodeMCU



في هذه الصفحة سيقوم كل شخص بإختيار القنوات أو المواقع التي يهتم بها كثيرا. ثم بعد ذلك سنضغط على (Continue).

بعد ذلك سنقوم بإختيار الأداة (Recipe) الخاصة بمشروعنا من خلال الرابط التالي:

<https://ifttt.com/recipes/315426-esp8266-email-notifier>

بعد الدخول على الرابط ستظهر لنا الصفحة التالية:



Notes: Your ESP8266 makes a request to sends an email when motion is detected with your PIR Motion Sensor.

Connect these Channels first



Maker Channel

Connect

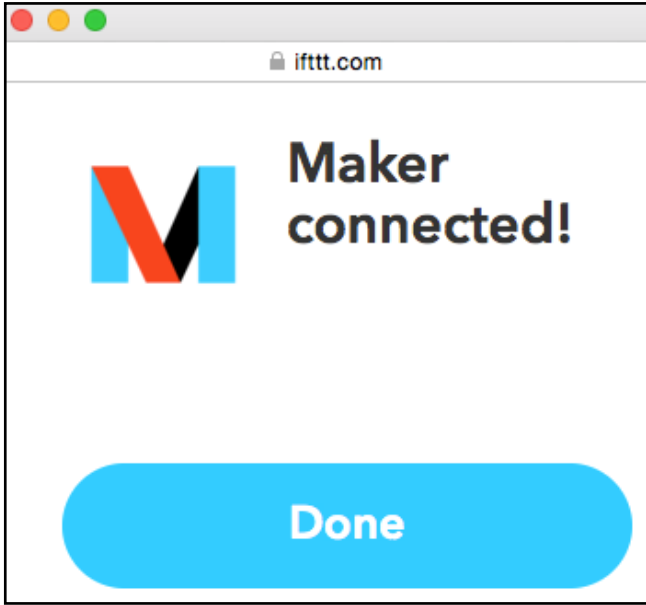


Gmail Channel

Connect

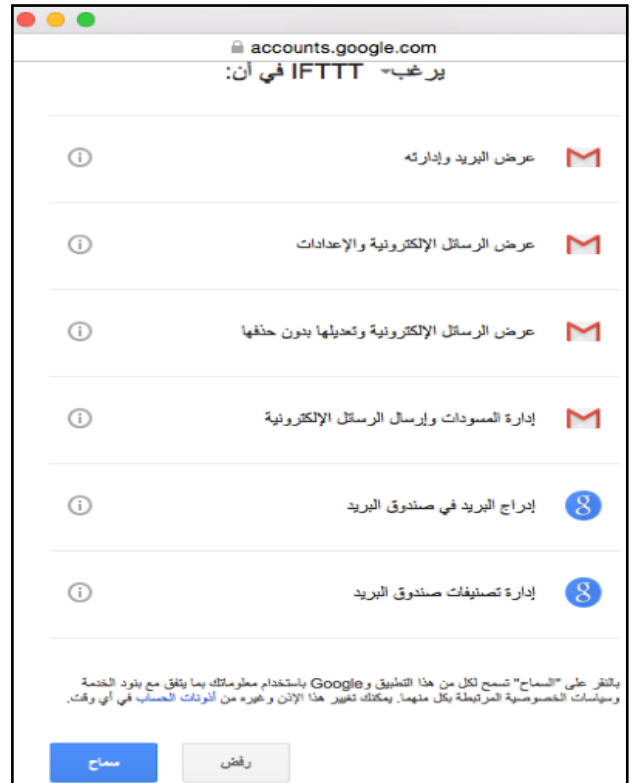
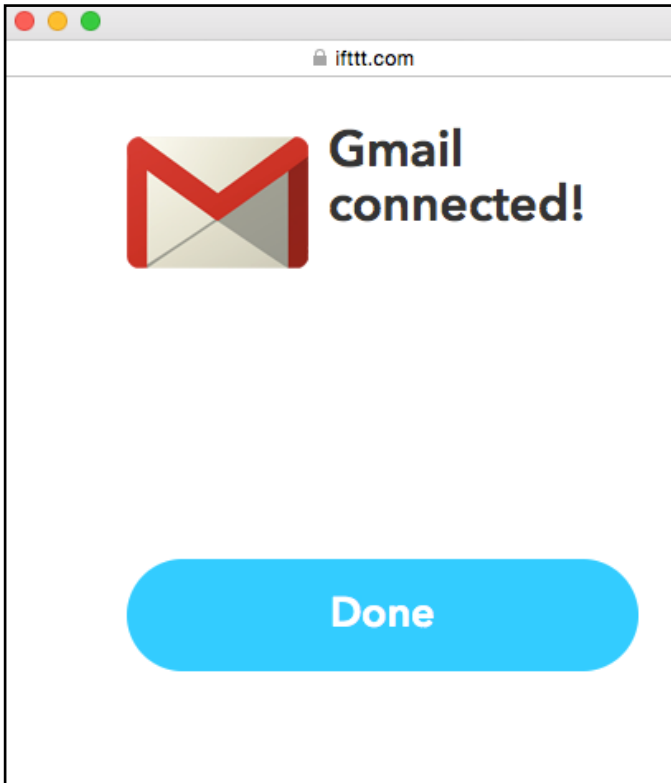
ملاحظة: حتى نستخدم هذه الأداة لابد أن يكون البريد إلكتروني الذي سنستقبل عليه التنبيهات من نوع Gmail.

من هذه الصفحة سنقوم بالضغط على (Connect) في قناة Maker و في قناة Gmail وستظهر لنا الصفحات التالية:



عند الضغط على (Connect) في قناة Maker ستظهر لنا هذه الصفحة و سنقوم بالضغط على (Done).

عند الضغط على (Connect) في قناة Gmail ستظهر لنا هاتين الصفحتين و سنقوم بالضغط على (سماح) و بالضغط على (Done).





بعد ذلك سنقوم بكتابة العبارة (motion_detected) في الخانة الأولى.
وفي الخانة الثانية سنقوم بكتابة البريد الإلكتروني الذي سنستقبل عليه
التنبيهات كما هو موضح في الصورة التالية:

M Your event name: motion_detected

The name of the event, like "button_pressed" or "front_door_opened"

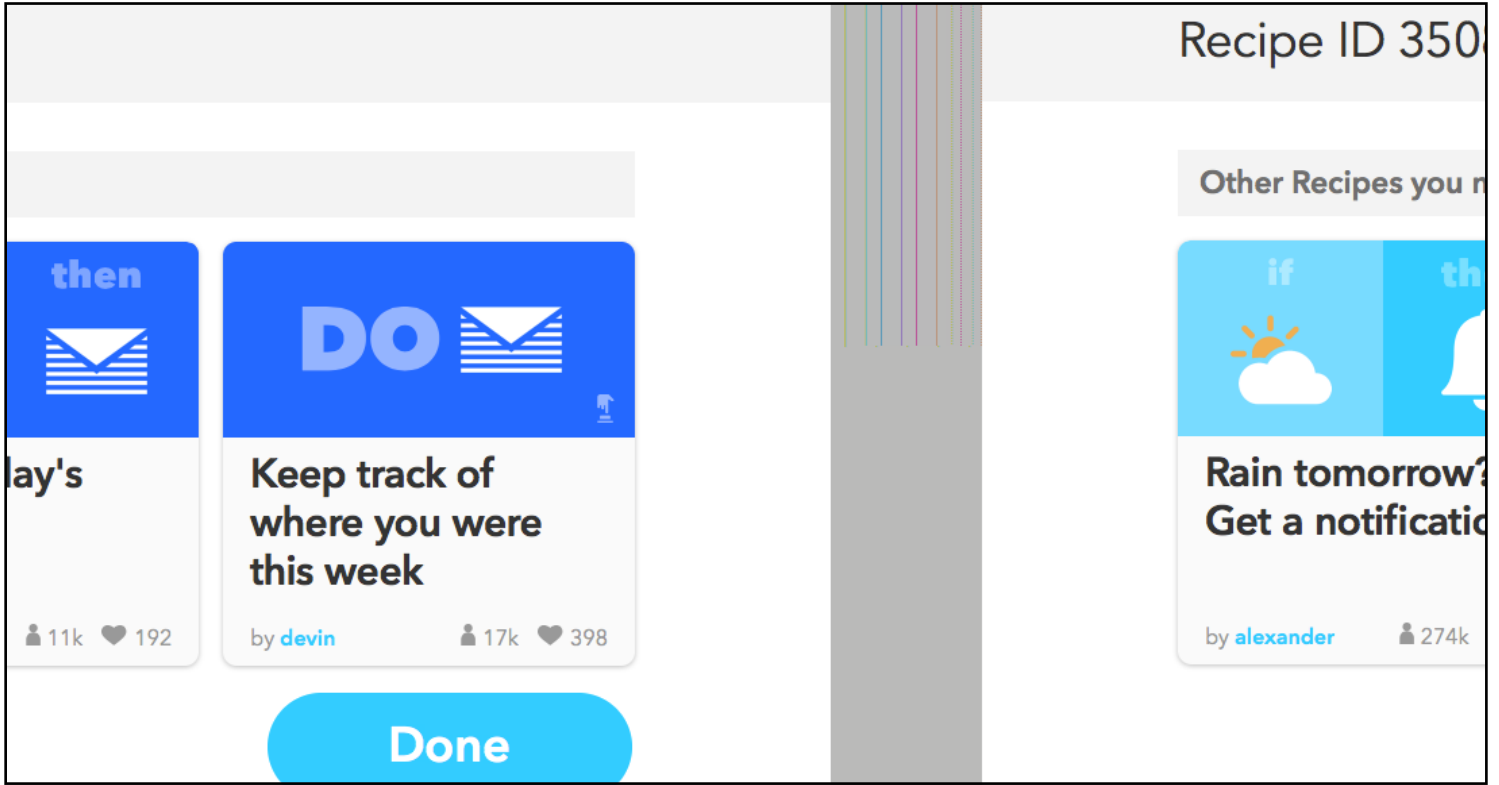
M To address

Accepts up to five email addresses, comma-separated

Receive notifications when this Recipe runs

Add

و بعد ذلك سنضغط على (Add). وبعدها ستظهر لنا الصفحة التالية:



من هذه الصفحة سنقوم بالضغط على (Done).

بعد ذلك سنقوم بالدخول على الرابط التالي:

<https://ifttt.com/maker>

بعد الدخول على الرابط ستظهر لنا الصفحة التالية:



Maker Channel

◀ All Channels

1 Personal Recipe



The Maker Channel allows you to connect IFTTT to your personal DIY projects. With Maker, you can connect a Recipe to any device or service that can make or receive a web request (aka webhooks). See how others are using the Maker Channel, or share your own experience at hackster.io.

Connected as: [jihadbassuni](#)

Reconnect Channel

Disconnect

[How to Trigger Events](#)

Your key is:

[cnaIpXPkSACc_ZSUincyfQ](#)

في هذه الصفحة سيظهر لنا المفتاح أو الرقم السري.

وبالنسبة لي فإن المفتاح السري هو: `cnaIpXPkSACc_ZSUincyfQ`

سنقوم بنسخه لإستخدامه لاحقاً.



NodeMCU



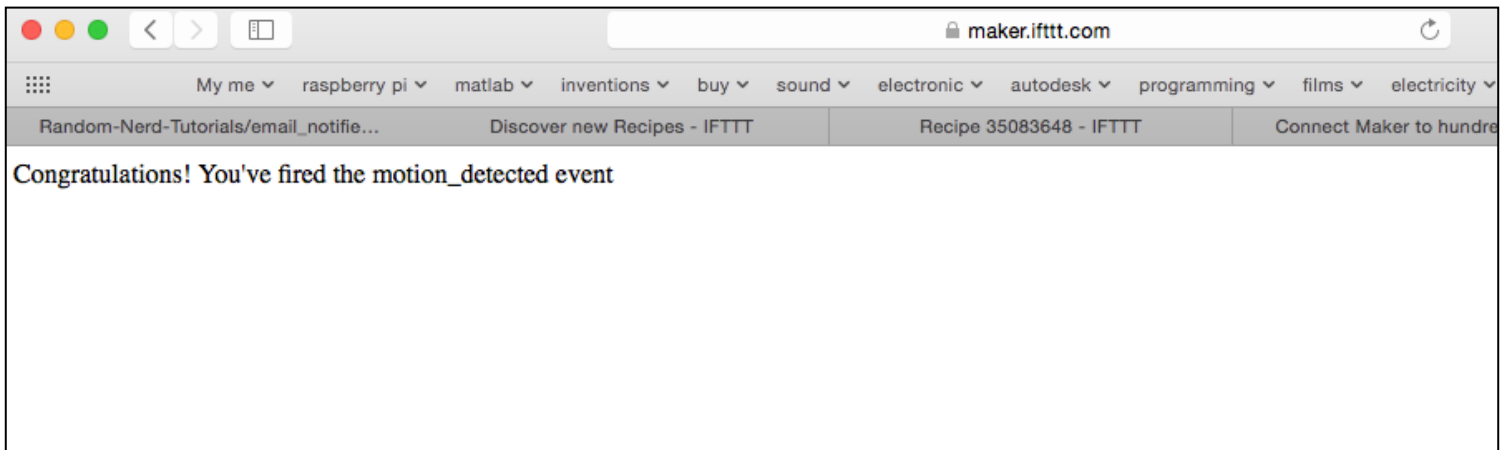
بعد ذلك سنقوم بعمل تجربة لنختبر هل تم تفعيل الأداة بنجاح أو لا من خلال كتابة الرابط التالي في المتصفح:

```
https://maker.ifttt.com/trigger/motion_detected/with/  
key/YOUR_API_KEY
```

نستبدل **YOUR_API_KEY** بالمفتاح السري الذي حصلنا عليه سابقا كما هو موضح في الصورة السابقة. وبالنسبة لي سيصبح الرابط كالتالي:

```
https://maker.ifttt.com/trigger/motion_detected/with/  
key/cnaIpXPkSACc_ZSUincyfQ
```

بعد كتابة الرابط في المتصفح وفتحه ستظهر لنا الصفحة التالية:



سنلاحظ وجود العبارة التالية في الصفحة:

Congratulations!You've fired the motion_detected event



بعد ذلك سنذهب إلى البريد الإلكتروني وسنلاحظ وجود تنبيه كما هو موضح في الصورة التالية:



وصول التنبيه الى البريد الإلكتروني يشير إلى أن الأداة تعمل بشكل جيد. **ملاحظة:** إن لم تجد رسالة التنبيه في مجلد البريد الوارد، فمن الممكن أن تكون في مجلد البريد الغير هام.

وبعد أن أنهينا من تفعيل أداة التنبيه سننتقل إلى جزء البرمجة.



كتابة الشيفرة البرمجية وشرحها:

الشيفرة البرمجية لهذا المشروع موجودة على الرابط التالي:

www.github/email-pir.lua

بعد ذلك سنقوم بفتح برنامج ESPlorer لكتابة الشيفرة البرمجية.

```
1 wifi.setmode(wifi.STATION)
2 wifi.sta.config("ssid", "pass")
3 pin = 4
4 gpio.mode(pin, gpio.INT)
```

السطر	الشرح
1	إعداد شريحة ESP8266 لإستقبال اشارة الواي فاي (wifi). أي أنه سيتم تشغيلها كعميل (client).
2	للإتصال بشبكة الواي فاي (wifi). وتحتوي على خانتين: ssid: سنكتب اسم الشبكة التي نريد الاتصال بها pass: سنكتب الرقم السري الخاص بالشبكة
3	قمنا بتعريف متغير بإسم pin وهو يشير الى المدخل D4
4	تهيئة المدخل D4 المتصل بحساس كاشف حركة الاجسام كقاطع (interrupt).



القاطع (interrupt): هو أمر برمجي وظيفته قطع البرنامج الفعال الرئيسي لتشغيل برنامج آخر. ويتم هذا القطع اعتمادا على القدح (Triggering).
القدح هو إنتقال المدخل (pin) من حالة إلى أخرى فمثلا إنتقاله من High إلى Low أو العكس.

وللتعرف أكثر على القاطع (interrupt) يفضل الدخول على الرابط التالي:

<http://www.genotronex.com/2013/03/interrupt.html>



```
6 function onChange ()
7
8   print('Motion Detected')
9   conn = nil
10  conn=net.createConnection(net.TCP, 0)
11  conn:on("receive", function(conn, payload) end)
12  conn:connect(80,"maker.ifttt.com")
13  conn:on("connection", function(conn, payload)
14    conn:send("POST /trigger/motion_detected/with/key/YOUR_API_KEY HTTP/1.1\r\n
```

السطر	الشرح
6	قمنا بإنشاء دالة بإسم onChange()
8	طباعة العبارة Motion Detected
9	المنفذ 80 (port 80) يستخدم للاتصال بين الخادم و العميل.
12	
13	سيتم الدخول على الرابط الذي سيقوم بتفعيل أداة التنبيه وإرسال تنبيه
14	إلى البريد الإلكتروني بأن الحساس قد اكتشف مرور جسم من أمامه

ملاحظة في السطر 14: يجب أن نستبدل العبارة **YOUR_API_KEY** بالمفتاح السري الذي حصلنا عليه سابقا. وبالنسبة لي سيصبح الأمر كالتالي:

```
conn:send("POST /trigger/motion_detected/with/key/  
cnalpXPkSACc_ZSUincyfQ HTTP/1.1\r\nHost: maker.ifttt.com  
\r\nConnection: keep-alive\r\nAccept: */*\r\n\r\n") end)
```



```
15   conn:close()
16   print('Email Sent')
17 end
18 gpio.trig(pin, 'up', onChange)
```

السطر	الشرح
15	إغلاق الإتصال
16	طباعة العبارة Email sent
17	نهاية الدالة onChange()
18	سيختبر ما إذا كان هناك قذح تتحول فيه حالة المدخل من Low إلى High. فإذا كان هناك قذح أي أن الحساس قد أكتشف مرور جسم أمامه فإنه سيتم تنفيذ الدالة onChange() التي تحتوي على أمر إرسال تنبيه للبريد الإلكتروني.

الأمر (gpio.trig (pin, 'up', onChange) يحتوي على 3 عناصر:

pin: يشير الى المدخل D4 الذي سيستخدم كقاطع (interrupt).

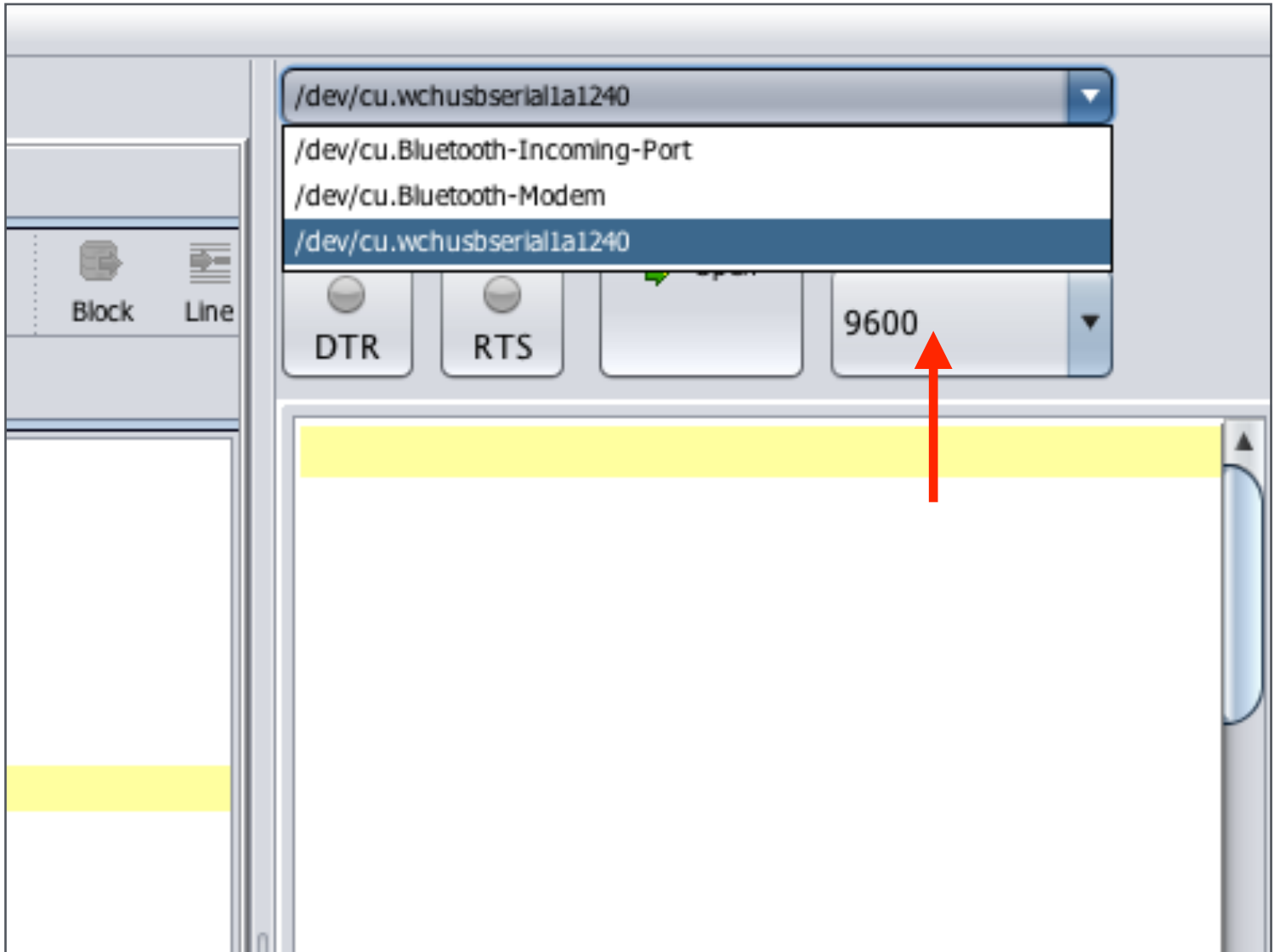
up: يدل على أن عملية إنتقال حالة المدخل ستكون من Low إلى High.

onChange(): اسم الدالة التي سيتم تنفيذها عندما تتم عملية القذح.



رفع البرنامج على لوحة NodeMCU وتشغيل المشروع:

بعد الانتهاء من كتابة الشيفرة البرمجية سنقوم بتوصيل لوحة NodeMCU بالحاسب. وبعدها سنقوم بإختيار المنفذ المتصل بلوحة NodeMCU. وأيضا سنختار 9600 كسرعة نقل البيانات كما هو موضح في الصورة التالية:



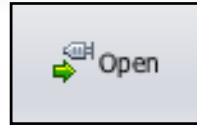
إن لم تجد المنفذ ضمن الخيارات فعليك بالضغط على زر التحديث وبعدها سيظهر معك المنفذ بإذن الله.



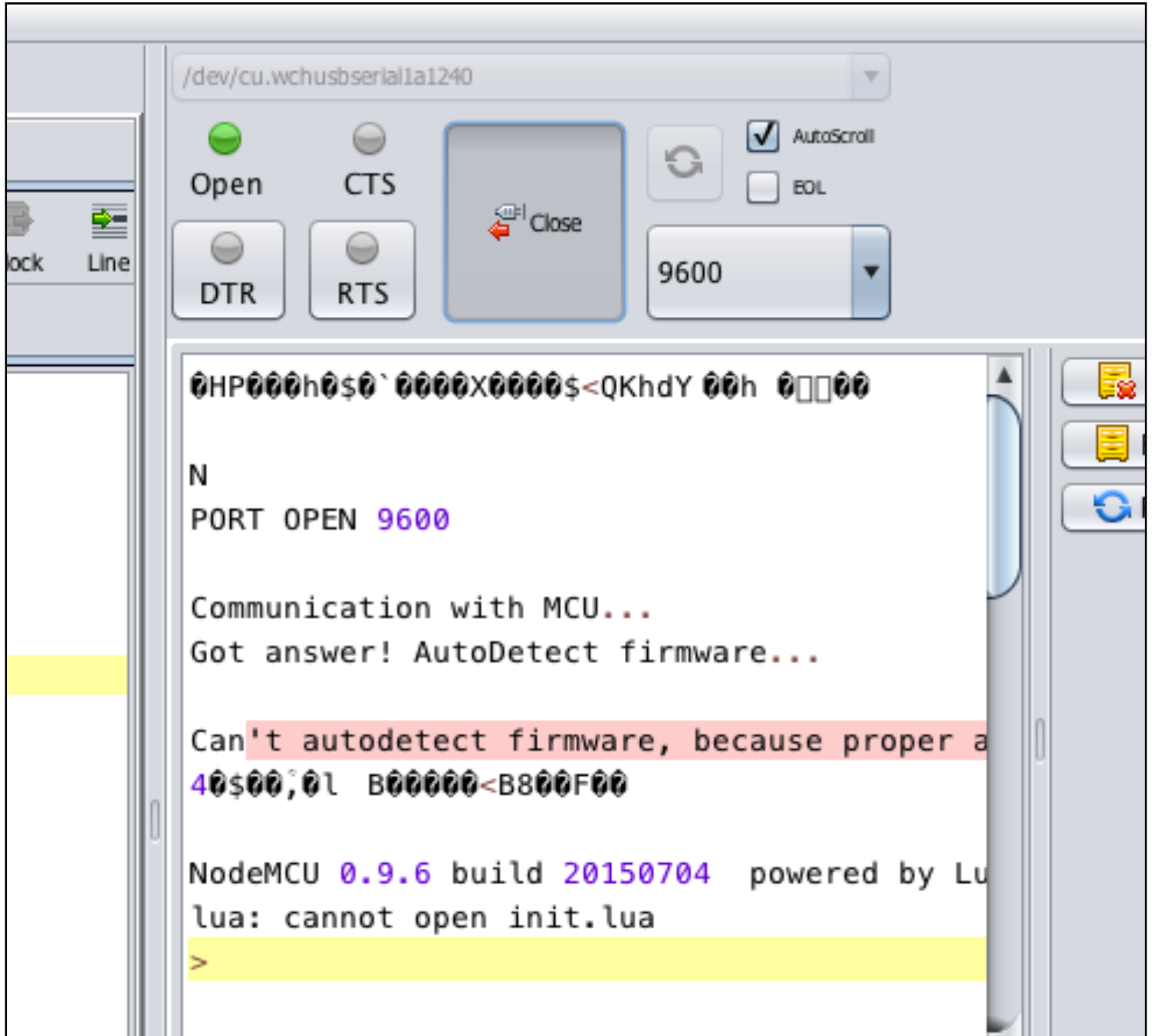
NodeMCU

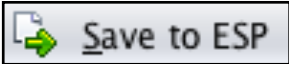


وسيداً عملية الاتصال كما

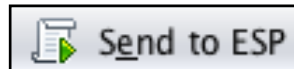


ثم سنضغط على زر تفعيل الاتصال
هو موضح في الصورة التالية:





بعد ذلك سنرفع البرنامج بالضغط على زر (save to ESP)



أو بالضغط على زر (send to ESP)

والفرق بين الاثنين هو:

- زر (save to ESP): يقوم بحفظ البرنامج داخل شريحة ESP8266 فحتى عند فصل اللوحة من الحاسب وتشغيلها باستخدام مصدر خارجي للتغذية (بطاريات) فإن المشروع سيعمل لأن البرنامج محفوظ داخل شريحة ESP8266.

- زر (send to ESP): يقوم فقط بقراءة البرنامج دون حفظه ولا يمكن تشغيل المشروع عند فصل اللوحة عن الحاسب وإستخدام مصدر خارجي للتغذية (بطاريات) لأن البرنامج غير محفوظ داخل شريحة ESP8266.

سأقوم بالضغط على زر (save to ESP) و سيطلب منا أن نقوم بحفظ الشيفرة البرمجية بإسم وسوف أسميه بـ `init.lua` كما هو موضح في الصورة التالية:

File Name:

Files of Type:

Save Cancel



NodeMCU



بعد الانتهاء من حفظ الشيفرة بإسم سيتم رفع البرنامج كما هو موضح في الصورة التالية:

The screenshot displays the NodeMCU IDE interface. On the left, the 'init.lua' script is visible, containing Lua code for setting up a WiFi station and a GPIO pin. A red box highlights the progress bar and status area, which shows '68%' completion and a 'BUSY' status. The right side of the IDE shows the serial monitor with the upload progress and a 'Send' button.

```
1 wifi.setmode(wifi.STATION)
2 wifi.sta.config("HUAWEI-E5330","fgm3fb9q")
3 pin = 4
4 gpio.mode(pin, gpio.INT)
5
6 function onChange ()
7
8     print('Motion Detected')
9     conn = nil
10    conn=net.createConnection(net.TCP, 0)
11    conn:on("receive", function(conn, payload) end)
12    conn:connect(80,"maker.ifttt.com")
13    conn:on("connection", function(conn, payload)
14        conn:send("POST /trigger/motion_detected/with/key/cn
15    conn:close()
16    print('Email Sent')
17 end
18 gpio.trig(pin, 'up', onChange)
```

68%

BUSY /Users/inventor/Desktop/pir/init.lua

Save&Run Save&Compile Save&Compile&RunLC Save As Init

Save&Compile All View on ESP View on ESP Save&Compile

Cancel Send to ESP Run Upload ...

Open CTS Close AutoScroll EOL

DTR RTS 9600

```
> file.remove("init.lua");
> file.open("init.lua","w+");
> w = file.writeline
> w([[wifi.setmode(wifi.STATION)]]);
> w([[wifi.sta.config("HUAWEI-E5330","fgm3fb9q")]]);
> w([[pin = 4 ]]);
> w([[gpio.mode(pin, gpio.INT)]]);
> w([[ ]]);
> w([[function onChange ()]]);
> w([[ ]]);
> w([[ print('Motion Detected')]]);
> w([[ conn = nil]]);
> w([[ conn=net.createConnection(net.TCP, 0)]]);
> w([[ conn:on("receive", function(c
> w([[ conn:on("connection", fu
```

Snippet0 Snippet1 Snippet2 Snippet3 Snippet4 Snippet5 Snippet6 Snippet7 Snippet8 Snippet9 Snippet10 Snippet11 Snippet12 Snippet13 Snippet14 Snippet15

Heap Chip Info Chip ID Flash

=node.heap() Send CR LF

رابط فيديو لمشاهدة تشغيل المشروع:

<https://www.youtube.com/watch?v=ItWVZUcArB0>



ملاحظة حول تسمية ملفات الشيفرة البرمجية:

حتى يعمل البرنامج المحفوظ على شريحة ESP8266 تلقائياً من غير مشاكل، فإنه يفضل تسمية جميع ملفات الشيفرة البرمجية بإسم `.init.lua`.
وأفضل طريقة هي أن يتم عمل مجلد خاص لكل شيفرة بحيث يتم تسمية المجلد بإسم المشروع الذي نريده وأما ملف الشيفرة فسيكون بإسم `init.lua` والمثال التالي يوضح المعنى:



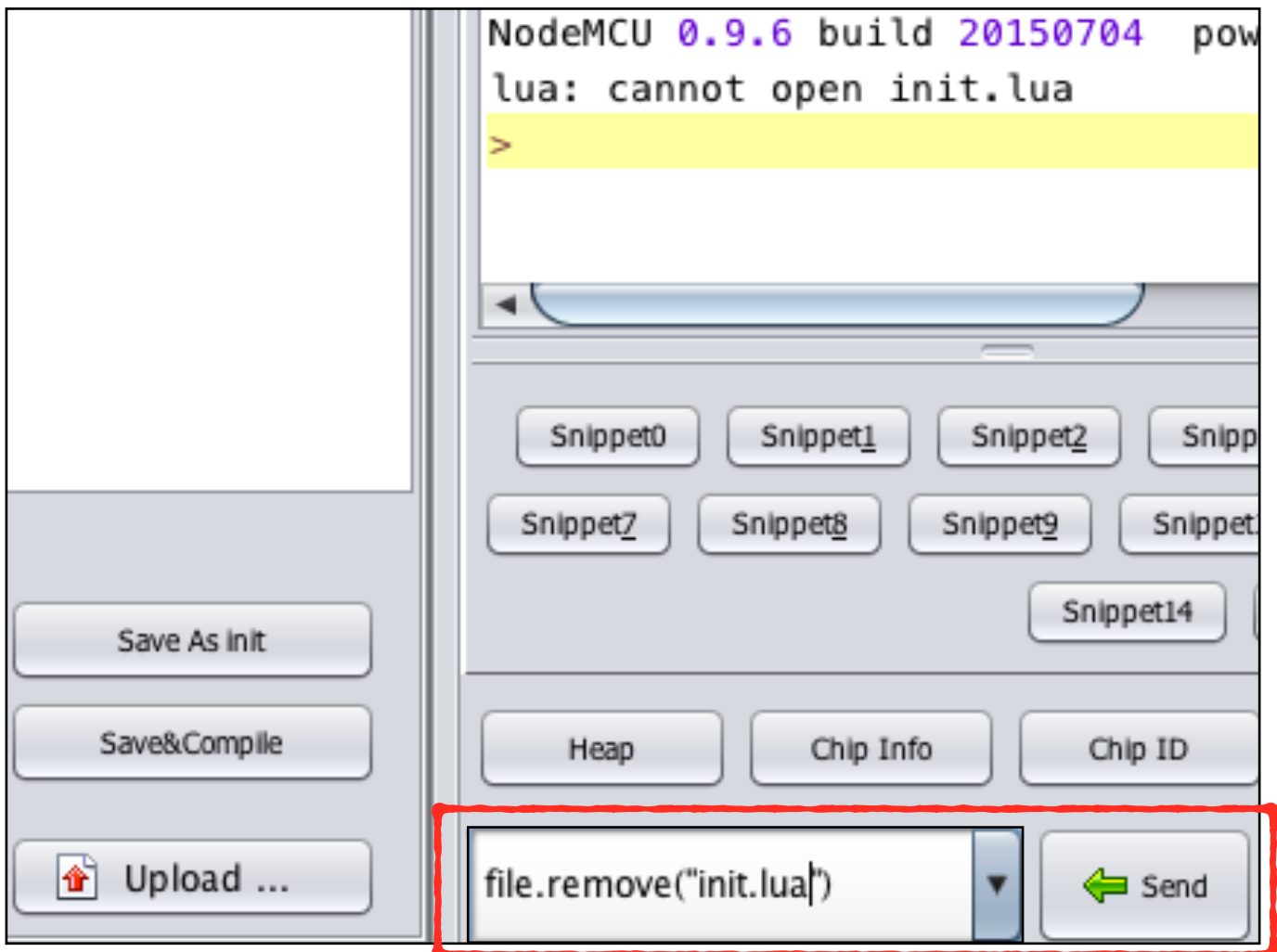
ملاحظات قبل تشغيل المشروع:

يفضل دائماً إعادة تشغيل لوحة NodeMCU بعد رفع البرنامج الجديد. لأنه من غير إعادة تشغيل فإنه سيتم تشغيل البرنامج القديم.



حذف البرنامج المحفوظ على شريحة ESP8266:

لا يشترط عمل هذه الخطوة كلما أردنا رفع برنامج جديد. فنستطيع رفع برنامج جديد على شريحة ESP8266 دون حذف البرامج القديمة. ولكن إذا أردنا حذف البرنامج نهائياً لتصبح شريحة ESP8266 خالية منه فنقوم بكتابة (`file.remove("init.lua")`) في الخانة الموضحة في الصورة ثم سنضغط على زر `send` وبعدها سيتم حذف البرنامج. أما إذا أردنا حذف جميع البرامج من شريحة ESP8266 فنقوم بكتابة (`file.format()`)





الفصل الثالث

المشاريع

(برنامج *Arduino*)



المشروع الأول (واجهة تشغيل و إطفاء):

فكرة المشروع:

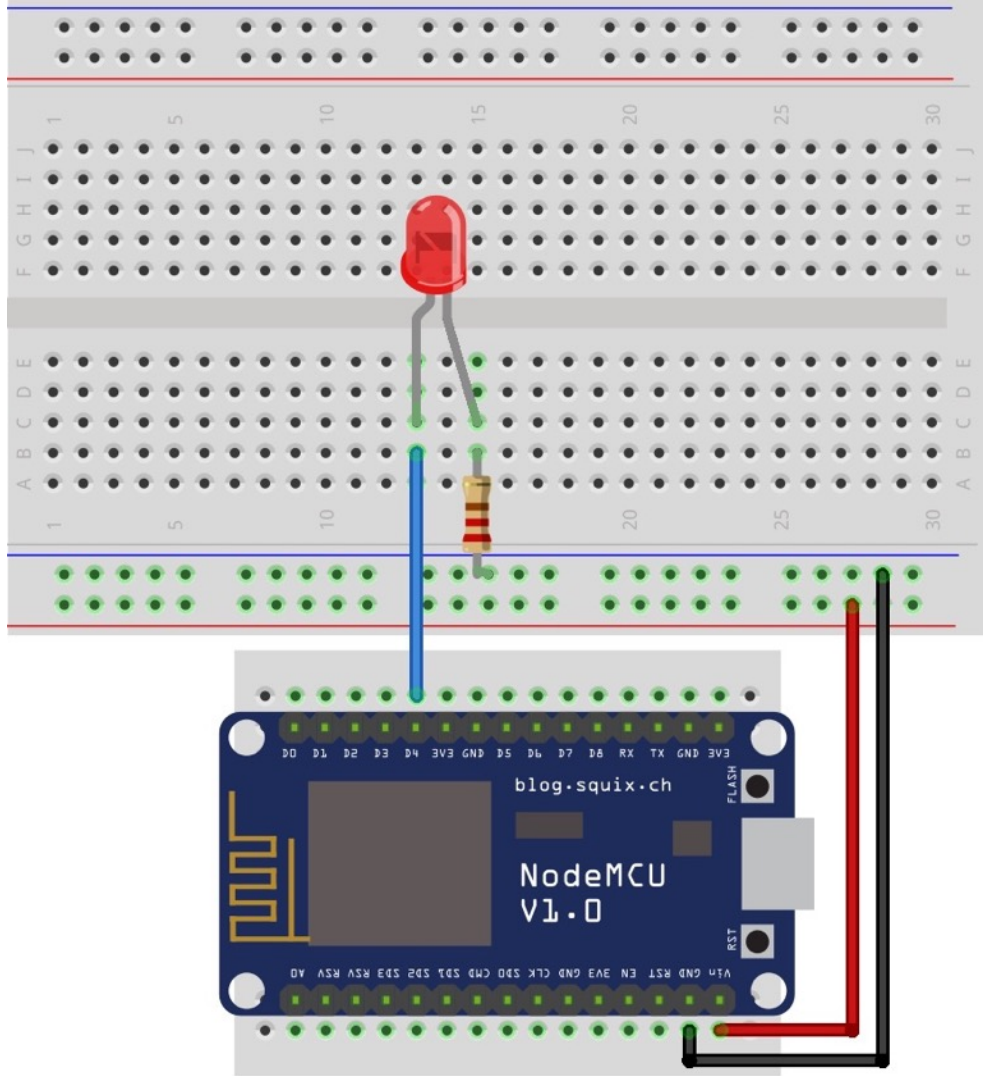
سنقوم بإنشاء صفحة خادم الشبكة تحتوي على مفتاح لتشغيل الدايود الضوئي ومفتاح آخر لإطفائه.

الأدوات المستخدمة:

- لوحة NodeMCU
- حامل لوحة NodeMCU
- لوحة تثبيت القطع الالكترونية (Breadboard)
- مقاومة 220 اوم
- دايود ضوئي (LED)



الدائرة الالكترونية:



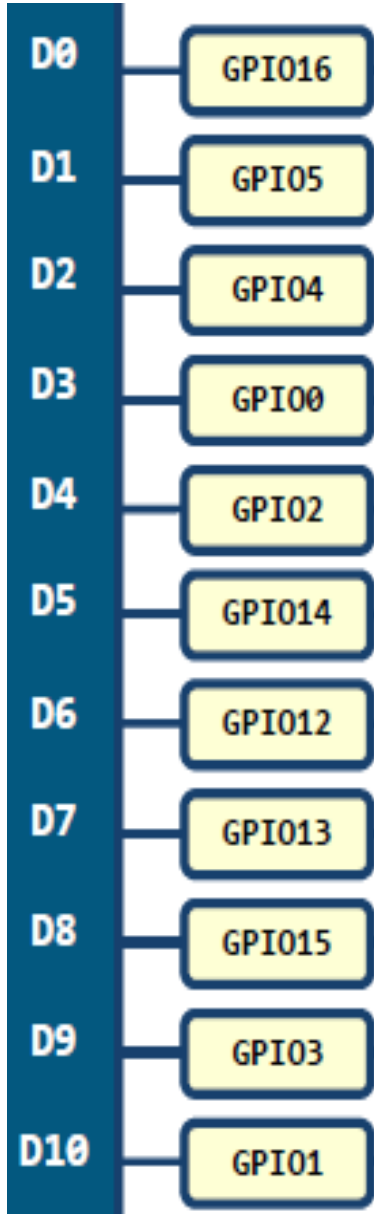
قمنا بتوصيل الطرف الموجب للديود الضوئي (LED) في المدخل D4 في لوحة NodeMCU والطرف السالب للديود الضوئي (LED) متصل بالارضي (GND) من خلال المقاومة 220.



ملاحظة حول تسمية مداخل لوحة NodeMCU:

الصورة المجاورة هي مقطع من مخطط لوحة NodeMCU الموجود في الصفحة رقم: (14) من هذا الكتاب.

نلاحظ من هذه الصورة أن لكل مدخل من مداخل لوحة NodeMCU تسميتين مختلفتين:



التسمية الثانية	التسمية الاولى
GPIO16 (16)	D0
GPIO5 (5)	D1
GPIO4 (4)	D2
GPIO0 (0)	D3
GPIO2 (2)	D4
GPIO14 (14)	D5
GPIO12 (12)	D6
GPIO13 (13)	D7
GPIO15 (15)	D8
GPIO3 (3)	D9
GPIO1 (1)	D10

فعند البرمجة باستخدام برنامج الاردوينو يجب علينا التعامل مع التسمية الثانية. فعلى سبيل المثال عندما اقوم بتوصيل الدايود الضوئي في المدخل D4 فعند كتابة البرنامج سأكتب أن (LED=2) حسب رقم التسمية الثانية.



كتابة الشيفرة البرمجية وشرحها:

الشيفرة البرمجية لهذا المشروع موجودة على الرابط التالي:

www.github/on-off-button.lua

بعد ذلك سنقوم بفتح برنامج الاردوينو لكتابة الشيفرة البرمجية.

```
1 #include <ESP8266WiFi.h>
2
3 const char* ssid = "your ssid";
4 const char* password = "your pass";
5
6 int led = 2;
7 int value = LOW;
8 WiFiServer server(80);
```

السطر	الشرح
1	مكتبة خاصة بشريحة ESP8266
3	ssid: اسم شبكة الواي فاي
4	pass: الرقم السري لشبكة الواي فاي
6	قمنا بتعريف متغير بإسم led وهو يشير الى المدخل GPIO2
7	قمنا بتعريف متغير بإسم value وأعطيناه القيمة LOW
8	منفذ يستخدم للاتصال بين الخادم و العميل



```
10 void setup(){
11
12 Serial.begin(115200);|
13 delay(10);
14 pinMode(led, OUTPUT);
15 digitalWrite(led, LOW);
16
17 Serial.println();
18 Serial.println();
19 Serial.print("Connecting to ");
20 Serial.println(ssid);
```

السطر	الشرح
10	دالة التهيئة
12	هذا الامر لفتح مراقب السريال عبر المنفذ رقم 115200
13	دالة تأخير بوحدة ملي ثانية. ومدة التأخير هنا هي 10 ملي ثانية
14	تهيئة الدايود الضوئي كخرج (output)
15	اعطاء الدايود الضوئي القيمة LOW. وهذا يعني انه سينطفئ
17	طباعة العبارة (conn...) وطباعة اسم الشبكة (ssid) على مراقب
20	السريال كما سنشاهد فيما بعد



```
22 WiFi.begin(ssid, password);
23 while (WiFi.status() != WL_CONNECTED) {
24   delay(500);
25   Serial.print(".");
26 }
27 Serial.println("");
28 Serial.println("WiFi connected");
29
30 server.begin();
31 Serial.println("Server started");
32 Serial.print("Use this URL to connect: ");
33 Serial.print("http://");
34 Serial.print(WiFi.localIP());
35 Serial.println("/");
```

السطر	الشرح
22	هذا الامر لإتصال ESP8266 بشبكة الواي فاي التي تم اختيارها
23	سيختبر هل تم الاتصال بالشبكة او لا. وفي حال لم يتصل فسيظل داخل دالة
26	while ويطبع (.) على مراقب السريال. اما اذا تم الاتصال فسيخرج من دالة while ويكمل بقية الشيفرة.
27	طباعة العبارة الموضحة بأنه تم الاتصال على مراقب السريال
28	
30	بدأ تشغيل شبكة الخادم للاتصال بها
31	سيتم طباعة العبارات الموضحة على مراقب السريال بالاضافة الى طباعة
35	عنوان IP الذي سنستخدمه للدخول الى صفحة شبكة الخادم



```
39 void loop() {  
40  
41 WiFiClient client = server.available();  
42 if (!client) {  
43 return;  
44 }  
45 Serial.println("new client");  
46  
47 client.println("HTTP/1.1 200 OK");  
48 client.println("Content-Type: text/html");  
49 client.println("");
```

السطر	الشرح
39	الدالة الرئيسية وهي لا نهائية تتكرر باستمرار
41	سيختبر هل هناك عميل متصل بشبكة الخادم ام لا. فإذا كان لا يوجد عميل فإنه سيكرر عملية الاختبار باستخدام الدالة return. أما إذا وجد عميل متصل فإنه سينتقل الى الاوامر التالية.
44	ملاحظة: العميل هو اي جهاز سنستخدمه للتحكم بلوحة NodeMCU
45	طباعة العبارة الموضحة وهي (عميل جديد) على مراقب السريال
47	بروتوكول http و هو جدا مفيد للصفحات الالكترونية حيث يسرع
49	من عملية تحميل بيانات الصفحة



```
51 client.println("<!DOCTYPE HTML>");
52 client.println("<html>");
53 client.println("<head>");
54 client.println("<title>First Project</title>");
55 client.println("<meta charset='utf-8'>");
56 client.println("<meta name='viewport' content='width=device-width, initial-scale=1'>");
57 client.println("<link rel='stylesheet' href='http://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css'>");
58 client.println("<script src='https://ajax.googleapis.com/ajax/libs/jquery/1.12.0/jquery.min.js'></script>");
59 client.println("<script src='http://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js'></script>");
60 client.println("</head>");
61
62 client.println("<body>");
63 client.println("<div class='container'>");
64 client.print("<h1>ESP8266 Web Server</h1>");
65 client.print("<p class='text-danger'>on-off Button:</p>");
66 client.print("<a href='/LED=ON' class='btn btn-success' role='button'>ON</a>");
67 client.print("<a href='/LED=OFF' class='btn btn-danger' role='button'>OFF</a>");
68 client.print("<br><br>");
```

السطر	الشرح
51	هذه الجزئية من الاوامر تستخدم لانشاء الصفحات الالكترونية باستخدام البوتستراب (Bootstrap) وقد تحدثنا عنها في الفصل السابق (صفحة خادم الشبكة).
60	



السطر 64 من الشيفرة البرمجية

السطر 65 من الشيفرة البرمجية

الأسطر من 66 إلى 67 من الشيفرة البرمجية

السطر 70 من الشيفرة البرمجية



للدخول الى صفحة خادم الشبكة سنقوم بكتابة عنوان IP في المتصفح كما هو ظاهر في الصورة السابقة **192.168.1.9** (يختلف من شخص لشخص آخر).

وعند الضغط على مفتاح التشغيل (ON) ذو اللون الأخضر، فإن المتصفح تلقائيا سيذهب الى العنوان **192.168.1.9/LED=ON** وسنلاحظ أن

المتغير LED أصبحت قيمته تساوي .ON



وعند الضغط على مفتاح الاطفاء (OFF) ذو اللون الأحمر، فإن المتصفح تلقائيا سيذهب الى العنوان **192.168.1.9/LED=OFF** وسنلاحظ أن

المتغير LED أصبحت قيمته تساوي .OFF





```
70 client.println("Led pin is now: ");
71 if(value == HIGH) {
72   client.print("On");
73 } else {
74   client.print("Off");
75 }
76
77 client.println("</div>");
78 client.println("</body>");
79 client.println("</html>");
```

السطر	الشرح
70	طباعة العبارة الموضحة على صفحة شبكة الخادم.
71	سيختبر إذا كانت قيمة المتغير value بـ HIGH فإنه سيتم طباعة
75	العبارة (On). وأما إذا كانت قيمة المتغير value بـ LOW فإنه سيتم طباعة العبارة (Off) على صفحة شبكة الخادم.
77	اغلاق الجزئية الخاصة بالبوستتراب (Bootstrap)
79	



```
81 while(!client.available()){
82   delay(1);
83 }
84
85 String request = client.readStringUntil('\r');
86 Serial.println(request);
87 client.flush();
```

السطر	الشرح
81	سيختبر هل قام العميل بإرسال بيانات ام لا. فإذا لم يتم إرسال اي بيانات فإنه سيدخل داخل دالة while ولن يخرج منها إلا إذا قام العميل بإرسال بيانات.
83	ملاحظة: يتم ارسال البيانات (/LED=ON) عند الضغط على المفتاح الاخضر أو (/LED=OFF) عند الضغط على المفتاح الأحمر
85	الأمر client.readStringUntil(\r) يقوم بقراءة البيانات المرسله وتخزينها في المتغير request
86	طباعة قيمة المتغير request
87	لمسح البيانات من الذاكرة المؤقتة (buffer)



```
89 if (request.indexOf('/LED=ON') != -1) {
90   digitalWrite(led, HIGH);
91   value = HIGH;
92 }
93 if (request.indexOf('/LED=OFF') != -1) {
94   digitalWrite(led, LOW);
95   value = LOW;
96 }
```

السطر	الشرح
89	سيختبر هل القيمة المخزنة في المتغير request هي (/LED=ON).
92	إذا كانت هي فسيتم تشغيل الدايود الضوئي. والمتغير value ستصبح قيمته بـ HIGH.
93	سيختبر هل القيمة المخزنة في المتغير request هي (/LED=OFF).
96	إذا كانت هي فسيتم إطفاء الدايود الضوئي. والمتغير value ستصبح قيمته بـ LOW.



```
98 delay(1);
99 Serial.println("Client disconnected");
100 Serial.println("");
101
102 }
```

السطر	الشرح
98	دالة تأخير بوحدة ملي ثانية. ومدة التأخير هنا هي 1 ملي ثانية
99	طباعة العبارات الموضحة على مراقب السريال.
100	
102	نهاية البرنامج



رفع البرنامج على لوحة NodeMCU وتشغيل المشروع:

بعد الانتهاء من كتابة الشيفرة البرمجية سنقوم بتوصيل لوحة NodeMCU بالحاسب. وبعدها سنقوم بإختيار نوع اللوحة المستخدمة كما هو موضح في الصورة التالية:

The screenshot shows the Arduino IDE interface. The top menu bar includes 'مساعدة' (Help), 'ادوات' (Tools), and 'الشيفرة البرمجية' (Code). The 'ادوات' menu is open, showing options like 'تنسيق تلقائي' (⌘T), 'ارشفة الشيفرة البرمجية', 'الترميز و أعد التحميل', and 'مراقب المنفذ التسلسلي "سيريال بورت"' (⇧⌘M). The 'لوحة: "Arduino Uno"' (Board: "Arduino Uno") option is selected. Below the menu, the code editor shows the following code:

```
4 }
5
6 void loop() {
7   // put your main
8
9 }
```

The board selection dropdown menu is open, showing a list of boards. The 'NodeMCU 0.9 (ESP-12 Module)' option is highlighted in blue. Other boards listed include Arduino Uno, Arduino Duemilanove or Diecimila, Arduino Nano, Arduino Mega or Mega 2560, Arduino Mega ADK, Arduino Leonardo, Arduino Micro, Arduino Esplora, Arduino Mini, Arduino Ethernet, Arduino Fio, Arduino BT, LilyPad Arduino USB, LilyPad Arduino, Arduino Pro or Pro Mini, Arduino NG or older, Arduino Robot Control, Arduino Robot Motor, Arduino Gemma, ESP8266 Modules, Generic ESP8266 Module, ESPDuino (ESP-13 Module), Adafruit HUZZAH ESP8266, ESPresso Lite 1.0, ESPresso Lite 2.0, NodeMCU 1.0 (ESP-12E Module), Olimex MOD-WIFI-ESP8266(-DEV), SparkFun ESP8266 Thing, SparkFun ESP8266 Thing Dev, SweetPea ESP-210, and WeMos D1 R2 & mini.



NodeMCU

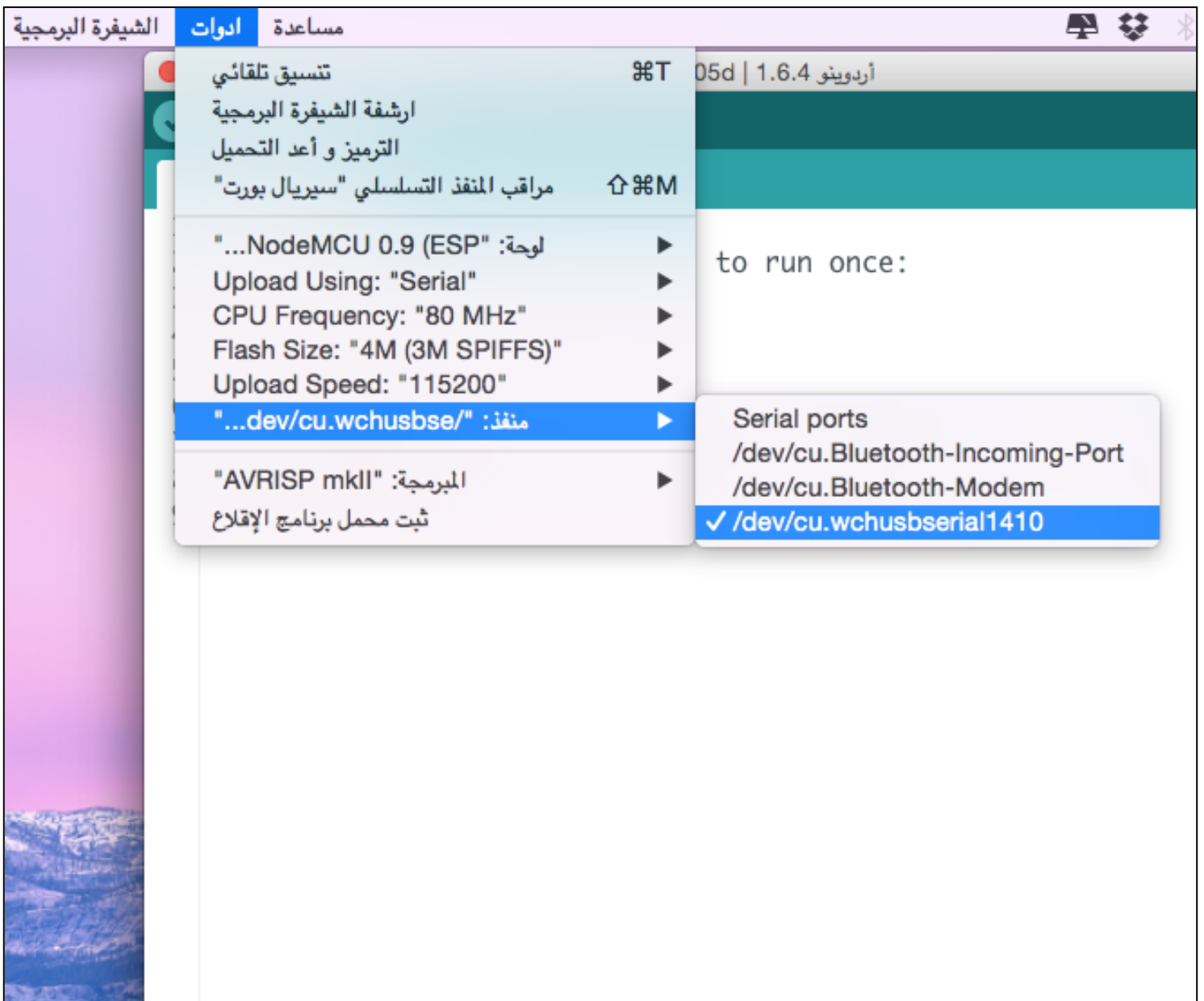


بعد ذلك سننتقل الى المسار التالي:

أدوات (Tools) ← منفذ (port)

وسنلاحظ وجود عدة منافذ وسنختار منها المنفذ المتصل بلوحة

NodeMCU كما هو موضح في الصورة التالية:

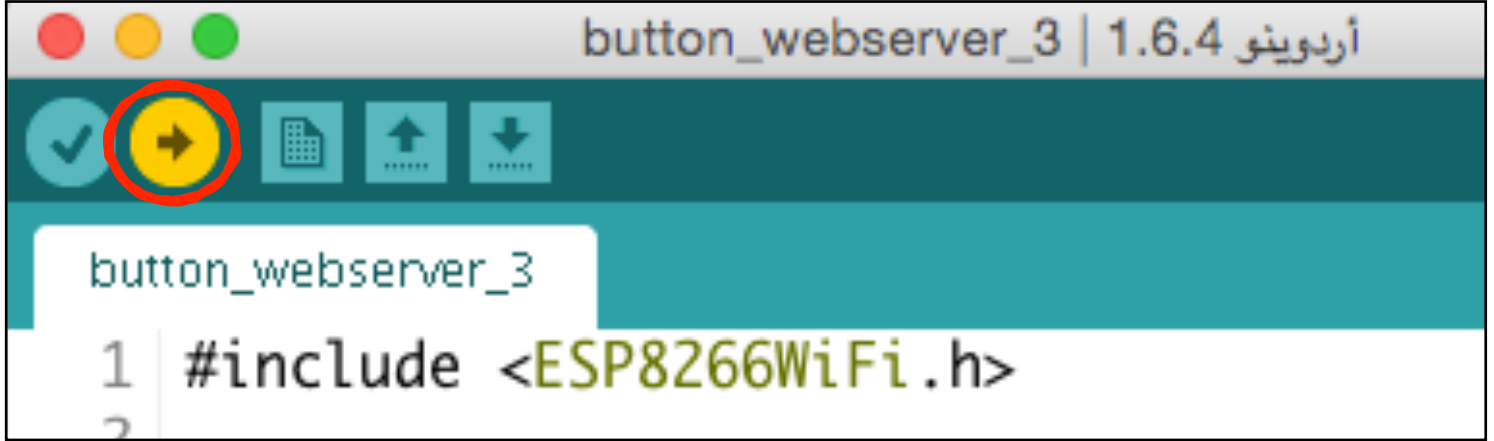




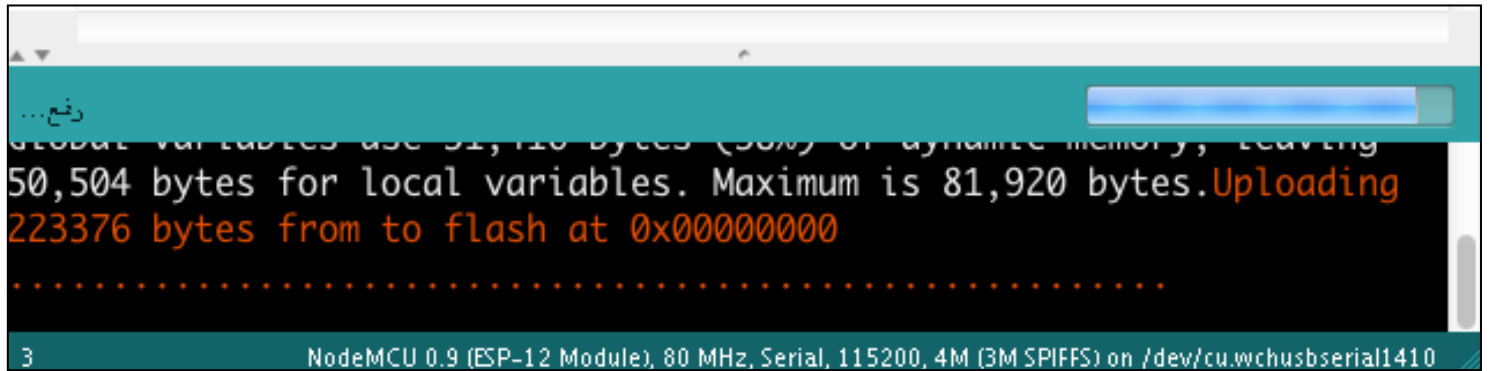
NodeMCU



بعد ذلك سنضغط على زر الرفع كما هو موضح في الصورة التالية:



وسنلاحظ أن البرنامج بدأ في عملية الرفع كما هو موضح في الصورة التالية:

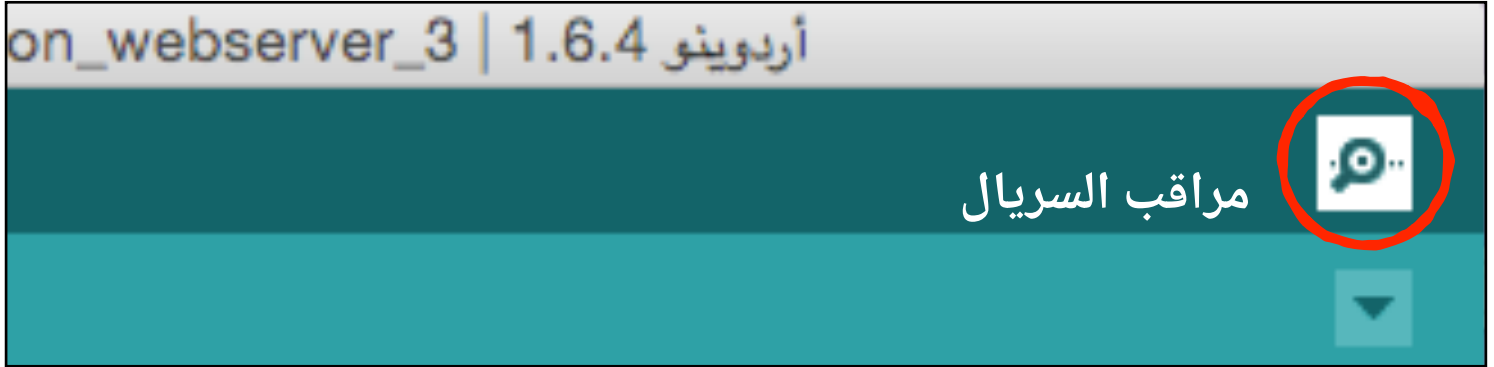




NodeMCU



بعد ذلك سنفتح مراقب السريال لمعرفة IP كما هو موضح في الصور التالية:



رابط فيديو لمشاهدة تشغيل المشروع:

<https://www.youtube.com/watch?v=449FbBCfKcA>



المشروع الثاني (عرض بيانات حساس الحرارة):

فكرة المشروع:

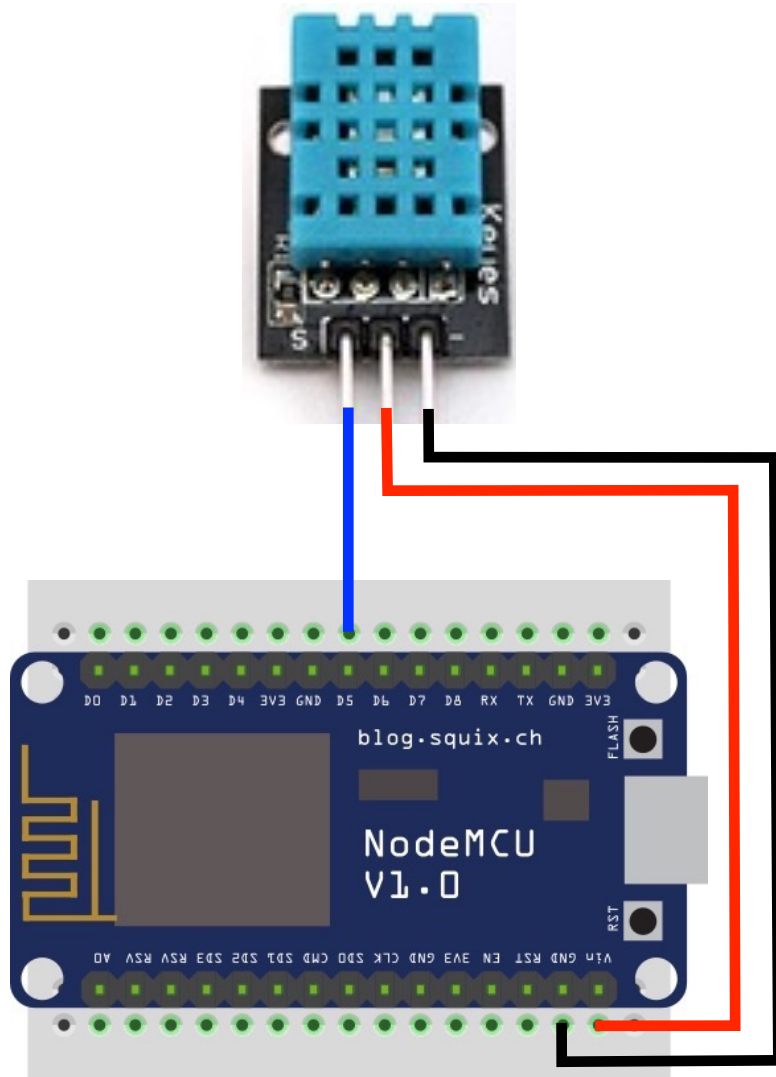
سنقوم بعرض التغير في درجة الحرارة والرطوبة على صفحة خادم الشبكة

الأدوات المستخدمة:

- لوحة NodeMCU
- حامل لوحة NodeMCU
- حساس الحرارة والرطوبة (DHT11)
- لوحة تثبيت القطع الالكترونية (Breadboard)



الدائرة الالكترونية (أ):

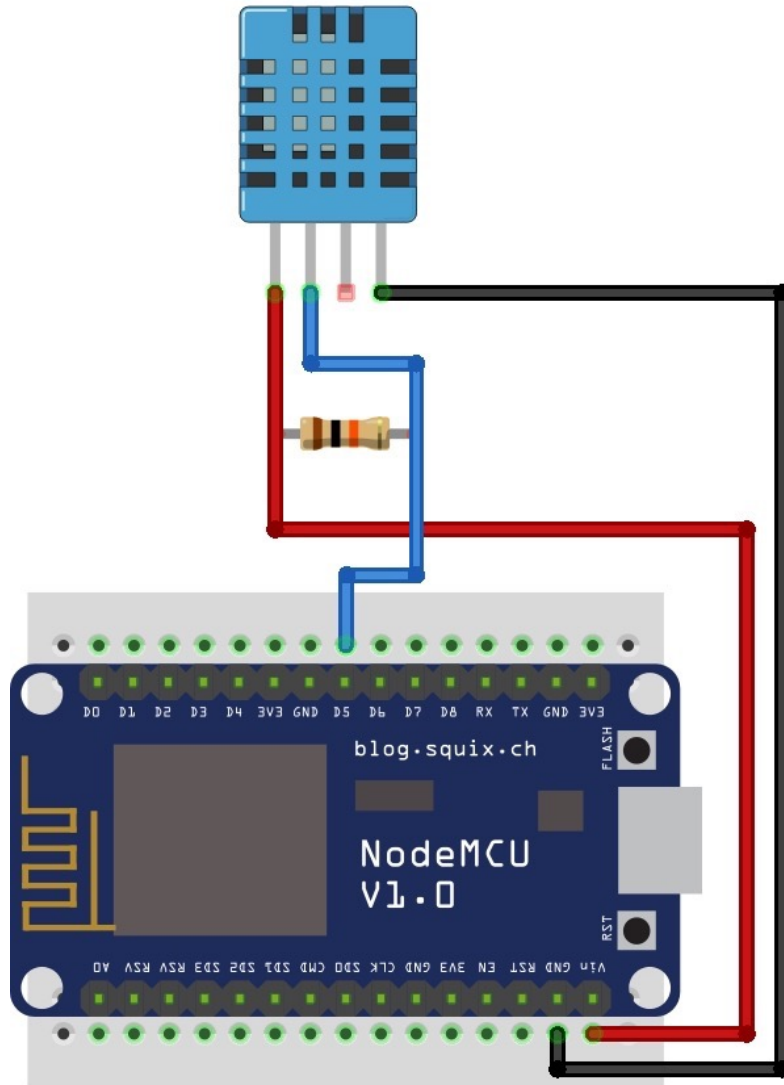


إذا كان الحساس مثبت على دائرة إلكترونية، فلن نحتاج إلى إضافة مكونات أخرى.

قمنا بتوصيل الطرف الموجب للحساس في المصدر 5v. والطرف السالب للحساس متصل بالأرضي (GND). والطرف الخرج (output) متصل في المدخل D5 في لوحة NodeMCU.



الدائرة الالكترونية (ب):



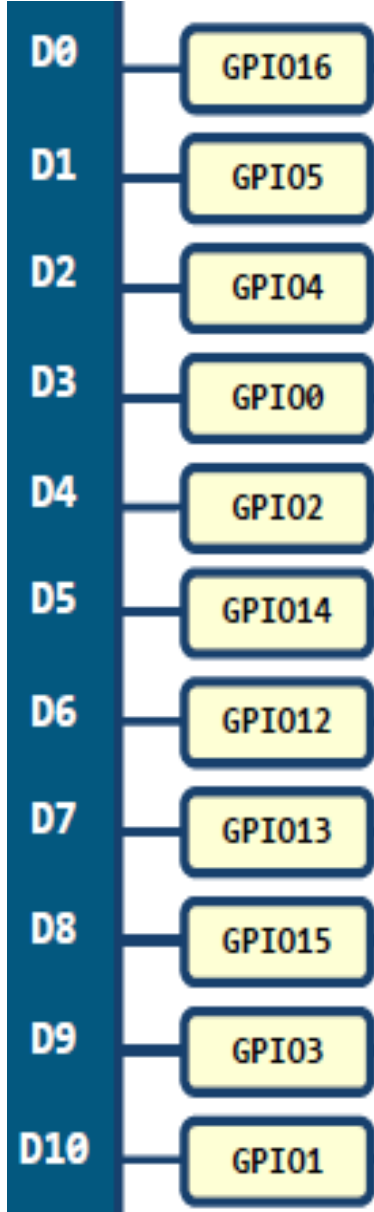
إذا كان الحساس غير مثبت على دائرة إلكترونية، فسنحتاج إلى إضافة مقاومة بقيمة 10 كيلوم تكون متصلة بين طرف الخرج (output) وطرف المصدر 5v. قمنا بتوصيل الطرف الموجب للحساس في المصدر 5v. والطرف السالب للحساس متصل بالارضى (GND). والطرف الخرج (output) متصل في المدخل D5 في لوحة NodeMCU.



ملاحظة حول تسمية مداخل لوحة NodeMCU:

الصورة المجاورة هي مقطع من مخطط لوحة NodeMCU الموجود في الصفحة رقم: (14) من هذا الكتاب.

نلاحظ من هذه الصورة أن لكل مدخل من مداخل لوحة NodeMCU تسميتين مختلفتين:



التسمية الثانية	التسمية الاولى
GPIO16 (16)	D0
GPIO5 (5)	D1
GPIO4 (4)	D2
GPIO0 (0)	D3
GPIO2 (2)	D4
GPIO14 (14)	D5
GPIO12 (12)	D6
GPIO13 (13)	D7
GPIO15 (15)	D8
GPIO3 (3)	D9
GPIO1 (1)	D10

فعند البرمجة باستخدام برنامج الاردوينو يجب علينا التعامل مع التسمية الثانية. فعلى سبيل المثال عندما اقوم بتوصيل الدايود الضوئي في المدخل D4 فعند كتابة البرنامج سأكتب أن (LED=2) حسب رقم التسمية الثانية.



كتابة الشيفرة البرمجية وشرحها:

الشيفرة البرمجية لهذا المشروع موجودة على الرابط التالي:

www.github/dht.lua

بعد ذلك سنقوم بفتح برنامج الاردوينو لكتابة الشيفرة البرمجية.

```
1 #include <ESP8266WiFi.h>
2 #include "DHT.h"
3 #define DHTPIN 14
4
5 #define DHTTYPE DHT11
6 // #define DHTTYPE DHT22
7 // #define DHTTYPE DHT21
```

السطر	الشرح
1	مكتبة خاصة بشريحة ESP8266
2	مكتبة حساس الحرارة DHT
3	قمنا بتعريف متغير بإسم DHTPIN وهو يشير الى المدخل GPIO14
5	سنختار نوع الحساس المستخدم عن طريق حذف علامة الملاحظة
7	(//) من أمام تعريف النوع المستخدم. وهنا سنستخدم DHT11



NodeMCU



```
9  const char* ssid = "your ssid";
10 const char* password = "your pass";
11
12 DHT dht(DHTPIN, DHTTYPE);
13 WiFiServer server(80);
```

السطر	الشرح
9	ssid: اسم شبكة الواي فاي
10	pass: الرقم السري لشبكة الواي فاي
12	أمر داخل مكتبة حساس الحرارة
13	منفذ يستخدم للاتصال بين الخادم و العميل



```
15 void setup(){
16
17   Serial.begin(115200);
18   delay(10);
19
20   Serial.println("DHTxx test!");
21   dht.begin();
22
23   Serial.println();
24   Serial.println();
25   Serial.print("Connecting to ");
26   Serial.println(ssid);
```

السطر	الشرح
15	دالة التهيئة
17	هذا الامر لفتح مراقب السريال عبر المنفذ رقم 115200
18	دالة تأخير بوحدة ملي ثانية. ومدة التأخير هنا هي 10 ملي ثانية
20	طباعة العبارة (DHTxx test!) على مراقب السريال
21	أمر داخل مكتبة الحساس ليبدأ الحساس حساب درجة الحرارة
23	طباعة العبارة (connecting to) وطباعة اسم الشبكة (ssid)
26	على مراقب السريال كما سنشاهد فيما بعد



```
28 WiFi.begin(ssid, password);
29 while (WiFi.status() != WL_CONNECTED) {
30   delay(500);
31   Serial.print(".");
32 }
33 Serial.println("");
34 Serial.println("WiFi connected");
35
36 server.begin();
37 Serial.println("Server started");
38 Serial.print("Use this URL to connect: ");
39 Serial.print("http://");
40 Serial.print(WiFi.localIP());
41 Serial.println("/");
```

السطر	الشرح
28	هذا الامر لإتصال ESP8266 بشبكة الواي فاي التي تم اختيارها
29	سيختبر هل تم الاتصال بالشبكة او لا. وفي حال لم يتصل فسيظل داخل
32	دالة while ويطبع (.) على مراقب السريال. اما اذا تم الاتصال فسيخرج من دالة while ويكمل بقية الشيفرة.
33	طباعة العبارة الموضحة بأنه تم الاتصال على مراقب السريال
34	
36	بدأ تشغيل شبكة الخادم للاتصال بها
37	سيتم طباعة العبارات الموضحة على مراقب السريال بالاضافة الى طباعة
41	عنوان IP الذي سنستخدمه للدخول الى صفحة شبكة الخادم



```
45 void loop() {
46
47   delay(1000);
48
49   float h = dht.readHumidity();
50   float t = dht.readTemperature();
51   float f = dht.readTemperature(true);
52
53   if (isnan(h) || isnan(t) || isnan(f)) {
54     Serial.println("Failed to read from DHT sensor!");
55     return;
56   }
```

السطر	الشرح
45	الدالة الرئيسية وهي لا نهائية تتكرر باستمرار
47	دالة تأخير بوحدة ملي ثانية. ومدة التأخير هنا هي 1000 ملي ثانية
49	حساب قيمة الرطوبة وتخزينها في المتغير h
50	حساب درجة الحرارة بالدرجة المئوية وتخزينها في المتغير t
51	حساب درجة الحرارة بالفهرنهايت وتخزينها في المتغير f.
53	سيختبر هل الحساس لم يستطع قياس الرطوبة أو الحرارة. فإن لم
56	يستطع القياس فسيتم طباعة العبارة (Failed...). ولن يكمل بقية البرنامج حتى يتمكن الحساس من قراءة الرطوبة أو الحرارة.



```
58 float hif = dht.computeHeatIndex(f, h);
59 float hic = dht.computeHeatIndex(t, h, false);
60
61 WiFiClient client = server.available();
62 if (!client) {
63   return;
64 }
65 Serial.println("new client");
66
67 client.println("HTTP/1.1 200 OK");
68 client.println("Content-Type: text/html");
69 client.println("");
```

السطر	الشرح
58	حساب قيمة مؤشر الحرارة بالفهرنهايت وتخزينها في المتغير hif.
59	حساب قيمة مؤشر الحرارة بالدرجة المئوية وتخزينها في المتغير hic
61	سيختبر هل هناك عميل متصل بشبكة الخادم ام لا. فإذا كان لا يوجد عميل فإنه سيكرر عملية الاختبار باستخدام الدالة return. أما إذا وجد عميل متصل فإنه سينتقل الى الاوامر التالية.
64	ملاحظة: العميل هو اي جهاز سنستخدمه للتحكم بلوحة NodeMCU
65	طباعة العبارة الموضحة وهي (عميل جديد) على مراقب السريال
67	بروتوكول http و هو جدا مفيد للصفحات الالكترونية حيث يسرع من
69	عملية تحميل بيانات الصفحة



```
71 client.println("<meta http-equiv=\"refresh\" content=\"3\">");
72 client.println("<!DOCTYPE html>");
73 client.println("<html xmlns='http://www.w3.org/1999/xhtml'>");
74 client.println("<head>\n<meta charset='UTF-8'>");
75 client.println("<title>ESP8266 Temperature & Humidity DHT11 Sensor</title>");
76 client.println("</head>\n<body>");
77 client.println("<H2>ESP8266 & DHT11 Sensor</H2>");
78 client.println("<H3>Humidity / Temperature</H3>");
79 client.println("<pre>");
80 client.print("Humidity: ");
81 client.print((float)h); client.println("%");
82 client.print("Temperature (Celsius): ");
83 client.print((float)t); client.println("°C");
84 client.print("Temperature (Fahrenheit): ");
85 client.print((float)f); client.println("°F");
86 client.print("HeatIndex (Celsius): ");
87 client.print((float)hic); client.println("°C");
88 client.print("HeatIndex (Fahrenheit): ");
89 client.print((float)hif); client.println("°F");
90 client.println("</pre>");
91 client.print("</body>\n</html>");
92 }
```

السطر	الشرح
71	هذه الجزئية من الاوامر تستخدم لانشاء الصفحات الالكترونية
91	باستخدام البوتستراب (Bootstrap) وقد تحدثنا عنها في الفصل السابق (صفحة خادم الشبكة).
92	نهاية البرنامج



```
14% 1:51 ص mobility ●●●●●
192.168.8.101
ESP8266 & DHT11 Sensor
Humidity / Temperature
Humidity: 41.00%
Temperature (Celsius): 26.00°C
Temperature (Fahrenheit): 78.80°F
HeatIndex (Celsius): 25.73°C
HeatIndex (Fahrenheit): 78.31°F
```

السطر 77 من الشيفرة البرمجية

السطر 87 من الشيفرة البرمجية

الأسطر من 79 إلى 90 من
الشيفرة البرمجية

ملاحظة:

سيتم تحديث معلومات الصفحة كل 3 ثواني. والأمر البرمجي المسؤول عن ذلك يوجد في السطر 71 من الشيفرة البرمجية.



رفع البرنامج على لوحة NodeMCU وتشغيل المشروع:

بعد الانتهاء من كتابة الشيفرة البرمجية سنقوم بتوصيل لوحة NodeMCU بالحاسب. وبعدها سنقوم بإختيار نوع اللوحة المستخدمة كما هو موضح في الصورة التالية:

The screenshot shows the Arduino IDE interface. The 'Tools' menu is open, and the 'Board' submenu is selected, displaying a list of boards. The board 'NodeMCU 0.9 (ESP-12 Module)' is highlighted in blue. The code editor shows a simple C++ sketch with a `void loop()` function.

```
4 }
5
6 void loop() {
7   // put your main
8
9 }
```



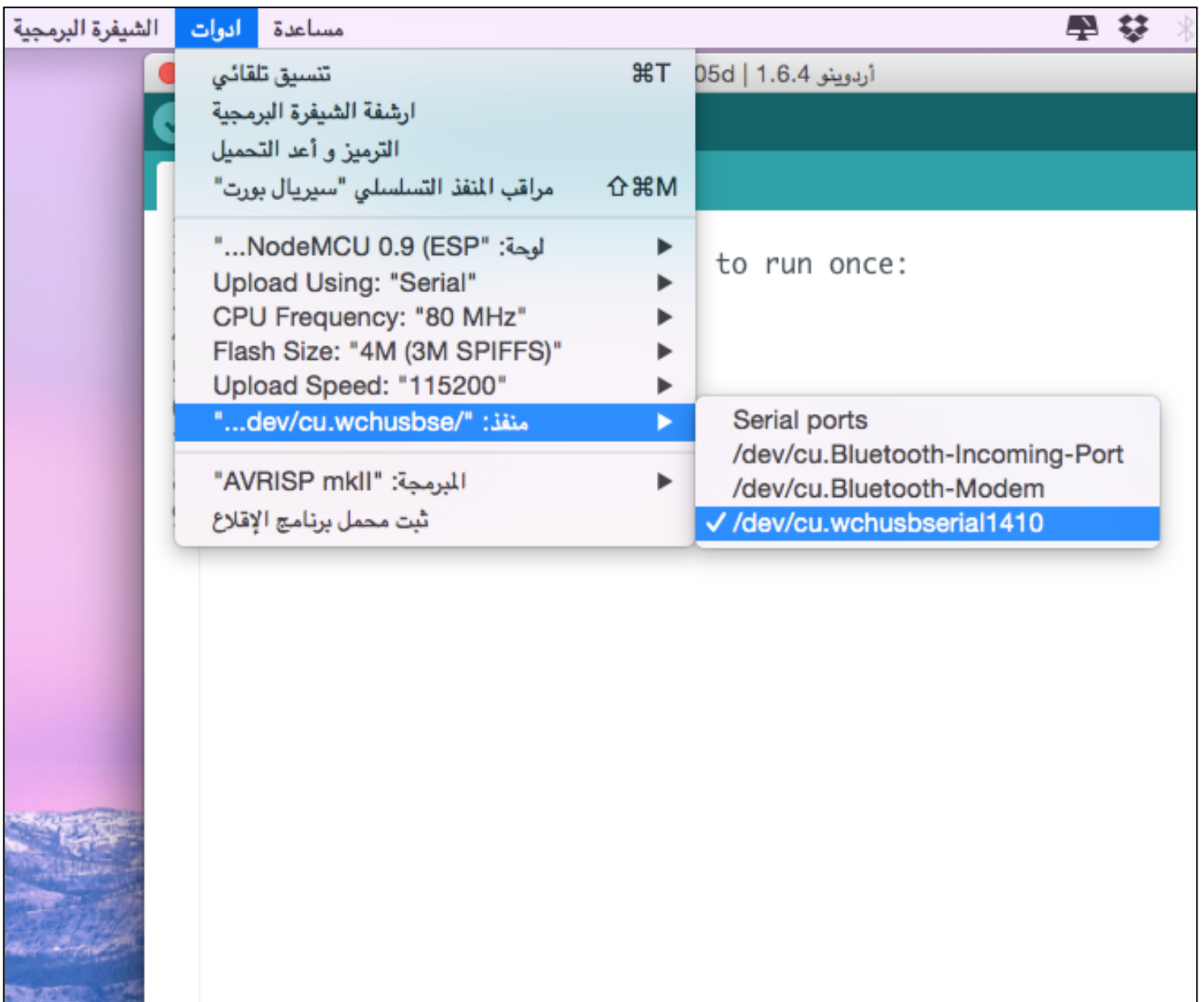
NodeMCU



بعد ذلك سننتقل الى المسار التالي:

أدوات (Tools) ← منفذ (port)

وسنلاحظ وجود عدة منافذ وسنختار منها المنفذ المتصل بلوحة NodeMCU كما هو موضح في الصورة التالية:

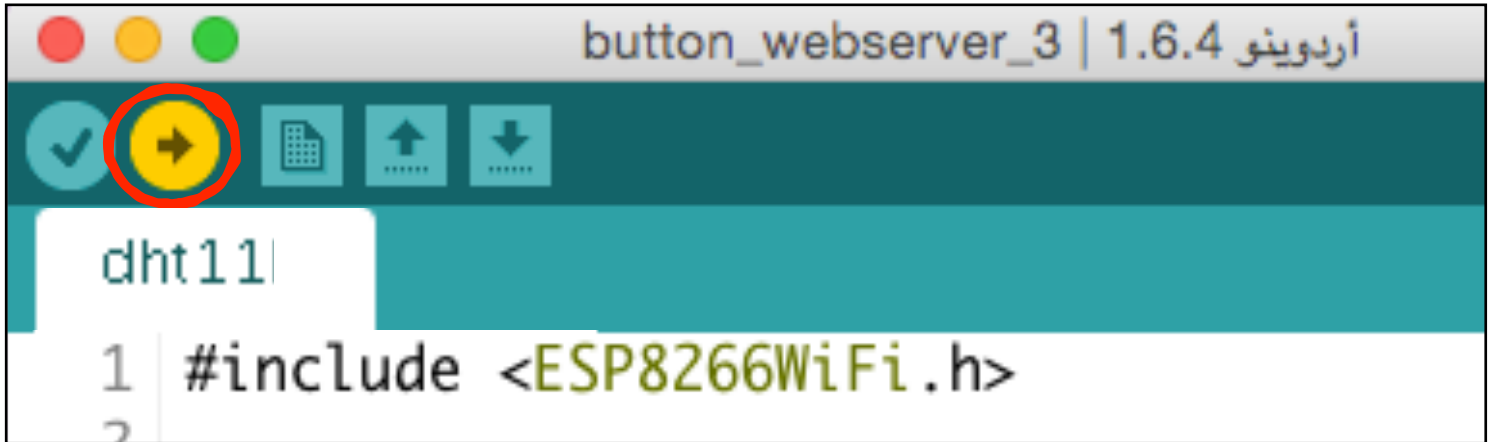




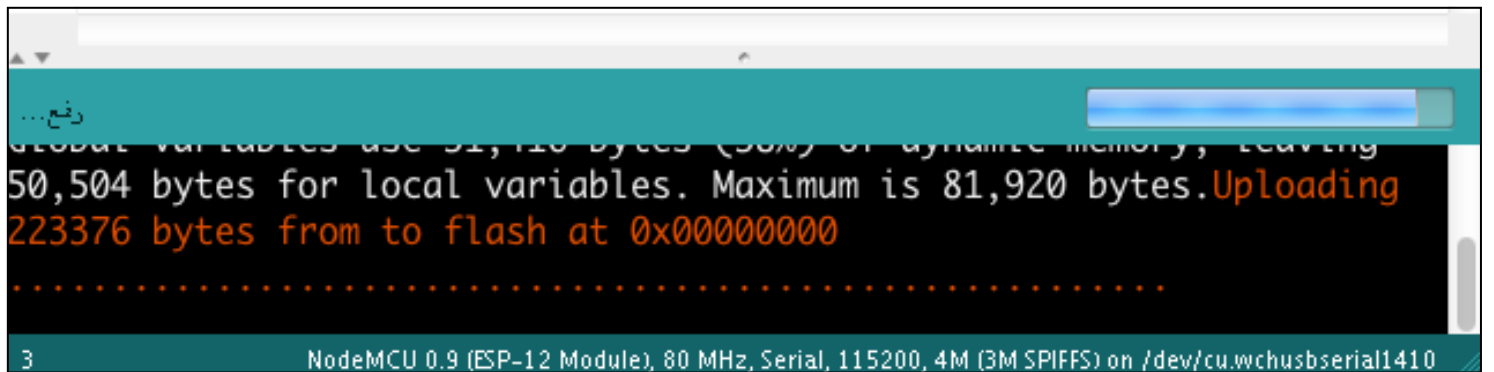
NodeMCU



بعد ذلك سنضغط على زر الرفع كما هو موضح في الصورة التالية:



وسنلاحظ أن البرنامج بدأ في عملية الرفع كما هو موضح في الصورة التالية:

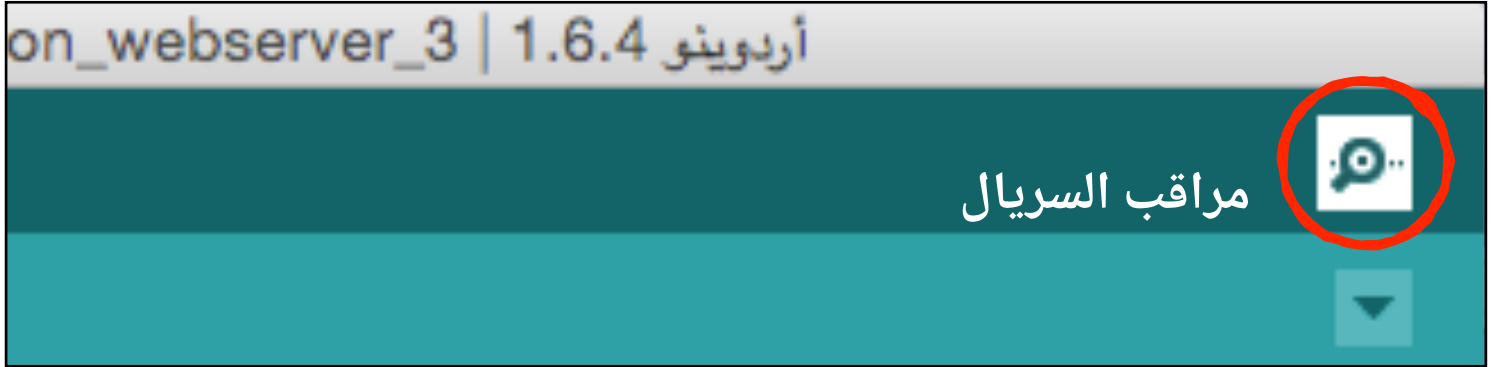




NodeMCU



بعد ذلك سنفتح مراقب السريال لمعرفة IP كما هو موضح في الصور التالية:



رابط فيديو لمشاهدة تشغيل المشروع:

<https://www.youtube.com/watch?v=WfZXg6FIOc>



المراجع



المراجع التي أعتمدت عليها في هذا الكتاب:

● كتاب HOME AUTOMATION USING ESP8266

● كتاب NodeMCU Development Workshop

● الموقع الرسمي للوحة NodeMCU

<http://www.nodemcu.com/docs/index>

● موقع تعليمي للبوستتراب (Bootstrap)

<http://www.w3schools.com/bootstrap/default.asp>



أعمال أخرى للمؤلف



من أعمال المؤلف:

● دورة معمل الاردوينو: الطريق للتحكم بالاشياء

دورة احترافية عملية مجانية. تتحدث عن برمجة القطع الالكترونية باستخدام الاردوينو. تجدونها على الرابط التالي:

<https://www.udemy.com/arduinoworkshop>

ولمشاهدة الدورة يجب أولاً التسجيل في الموقع والإشتراك في الدورة مجاناً.

● كتاب ماتلاب سيميولينك والأردوينو: البرمجة باستخدام البلوكات.
تجدونه على الروابط التالية:

<https://drive.google.com/file/d/>

[0BzFV7oWCcSrbT041S2RGdDZZMk0/view?usp=sharing](https://drive.google.com/file/d/0BzFV7oWCcSrbT041S2RGdDZZMk0/view?usp=sharing)



التواصل مع المؤلف



التواصل مع المؤلف على أحد هذه القنوات:

● البريد الإلكتروني

jihad.basuni@gmail.com

● الفيسبوك

<https://www.facebook.com/Inventor.Jihad>

● تويتر

https://twitter.com/jihad_basuni