المتحكمات الصغرية والنظم المضمنة

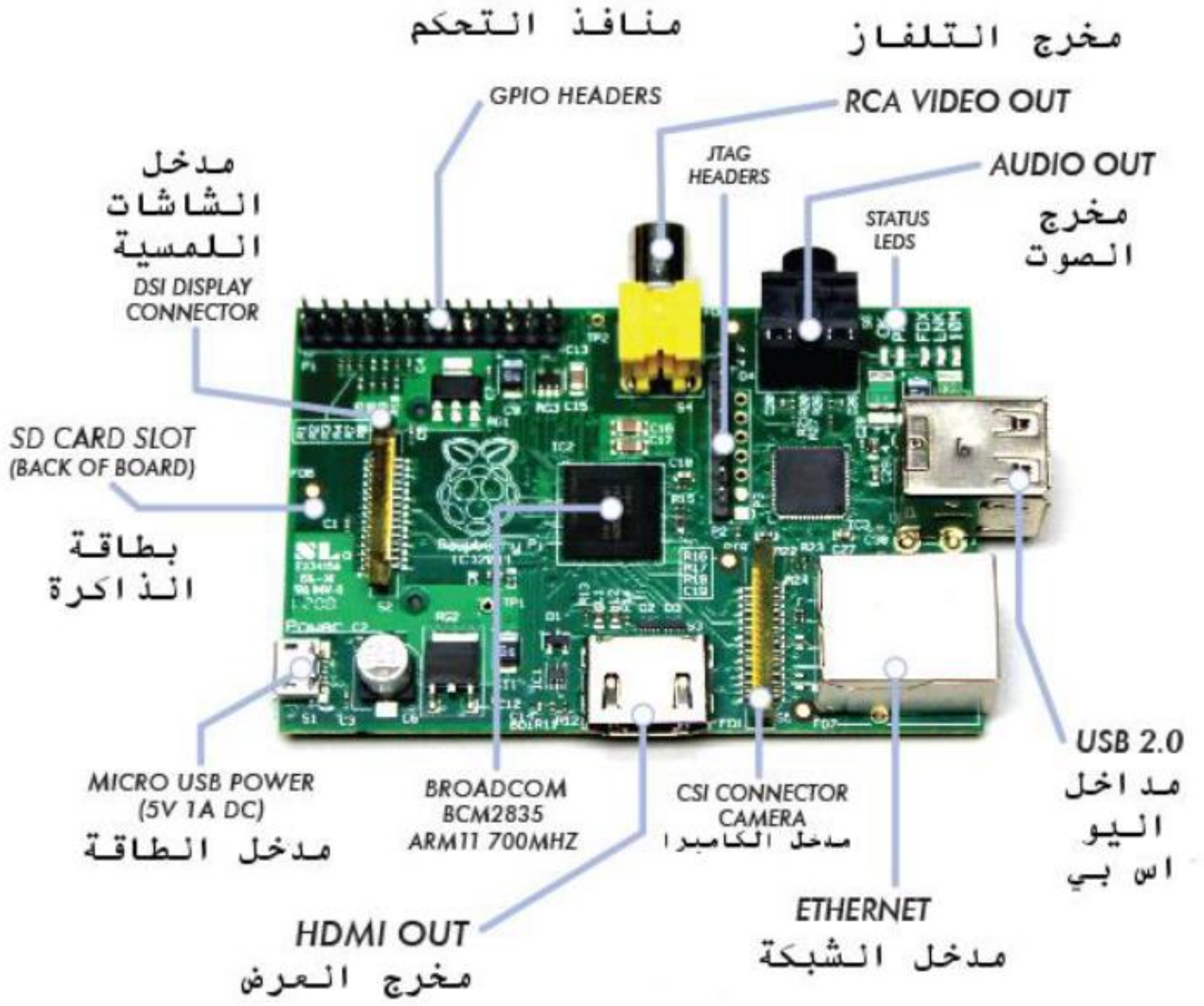
محاضرة 6

Raspberry Pi

د. فادي متوج

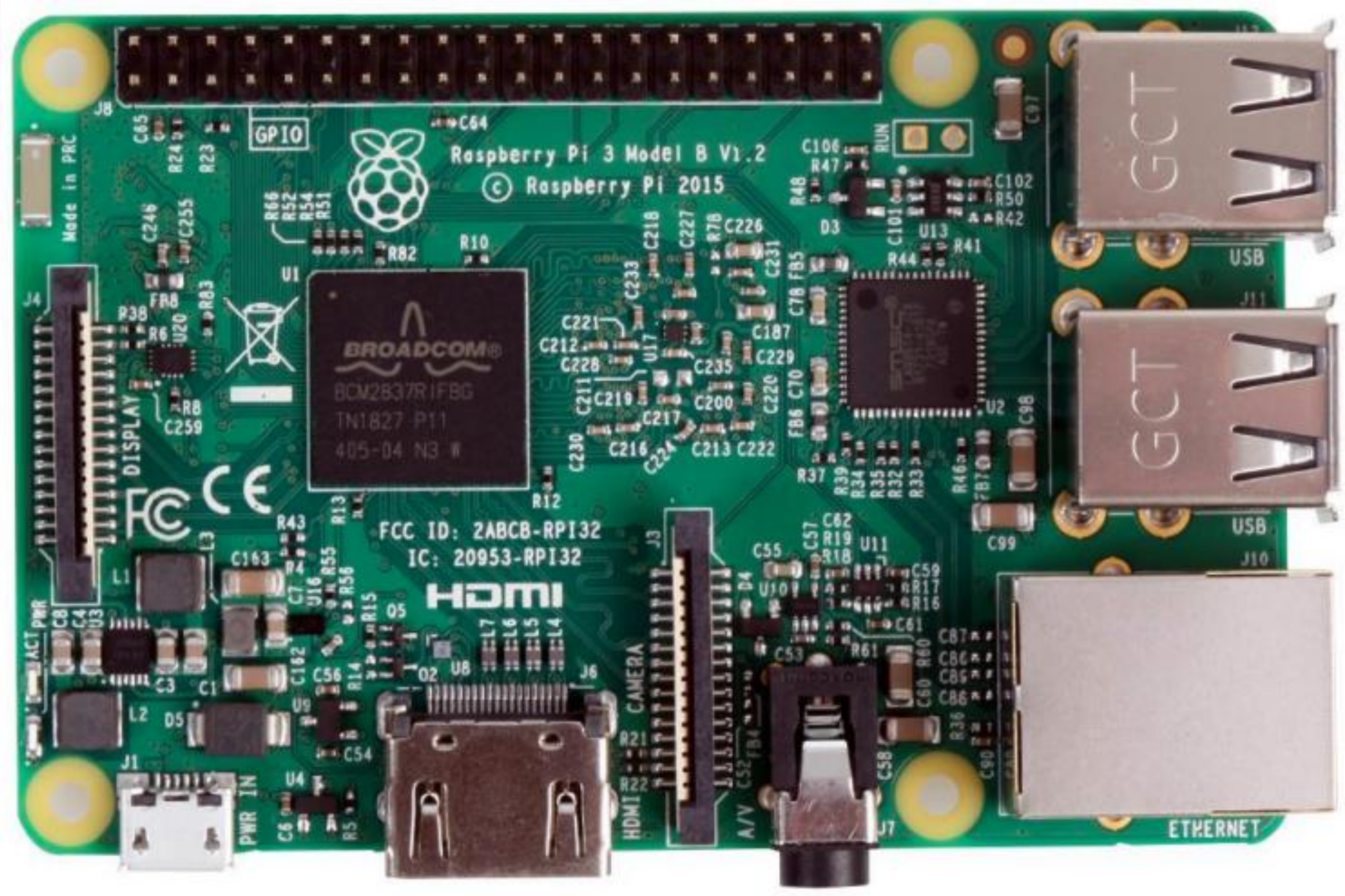
- تمثل لوحة الراسبيري حاسب آلي متكامل ومصنوع من شريحة الكترونية واحدة تحتوي مكونات الحاسب التقليدي وهي معالج البيانات CPU، معالج الرسومات GPU قادر على تشغيل الأفلام عالية الدقة HD و الألعاب ثلاثية الأبعاد مع ذاكرة عشوائية RAM، بالإضافة إلى مخارج تحكم رقمية يمكنها التحكم في القطع الإلكترونية و الكهربائية المختلفة. كل هذه الإمكانيات على رقاقة صغيرة فقط وهو ما يعرف باسم "النظام على شريحة SoC: System on Chip" ويتم تشغيل هذا الحاسوب الصغير بأنظمة تشغيل لينكس مفتوحة المصدر.

مكونات لوحة الراسبيري



جامعة المنارة
MANARA UNIVERSITY

Raspberry Pi 3 Model B



نظام التشغيل لينكس

- تعمل لوحة الراسبيري بنظام التشغيل لينكس وهو نظام تشغيل مجاني ومفتوح المصدر، مما يعني أنه يمكننا استخدامه مجانا ولنا كامل الحرية في التصرف به كما نشاء، أي يمكننا نسخه، توزيعه، وحتى تعديله من الصفر وذلك بفضل أن النظام مفتوح المصدر مما يجعل الشفرة المصدرية الخاصة بكامل النظام متاحه للجميع ويمكن لأي مبرمج أن يشارك في تطويره أو يشتق نسخة خاصة من النظام ونسبها إلى نفسه.

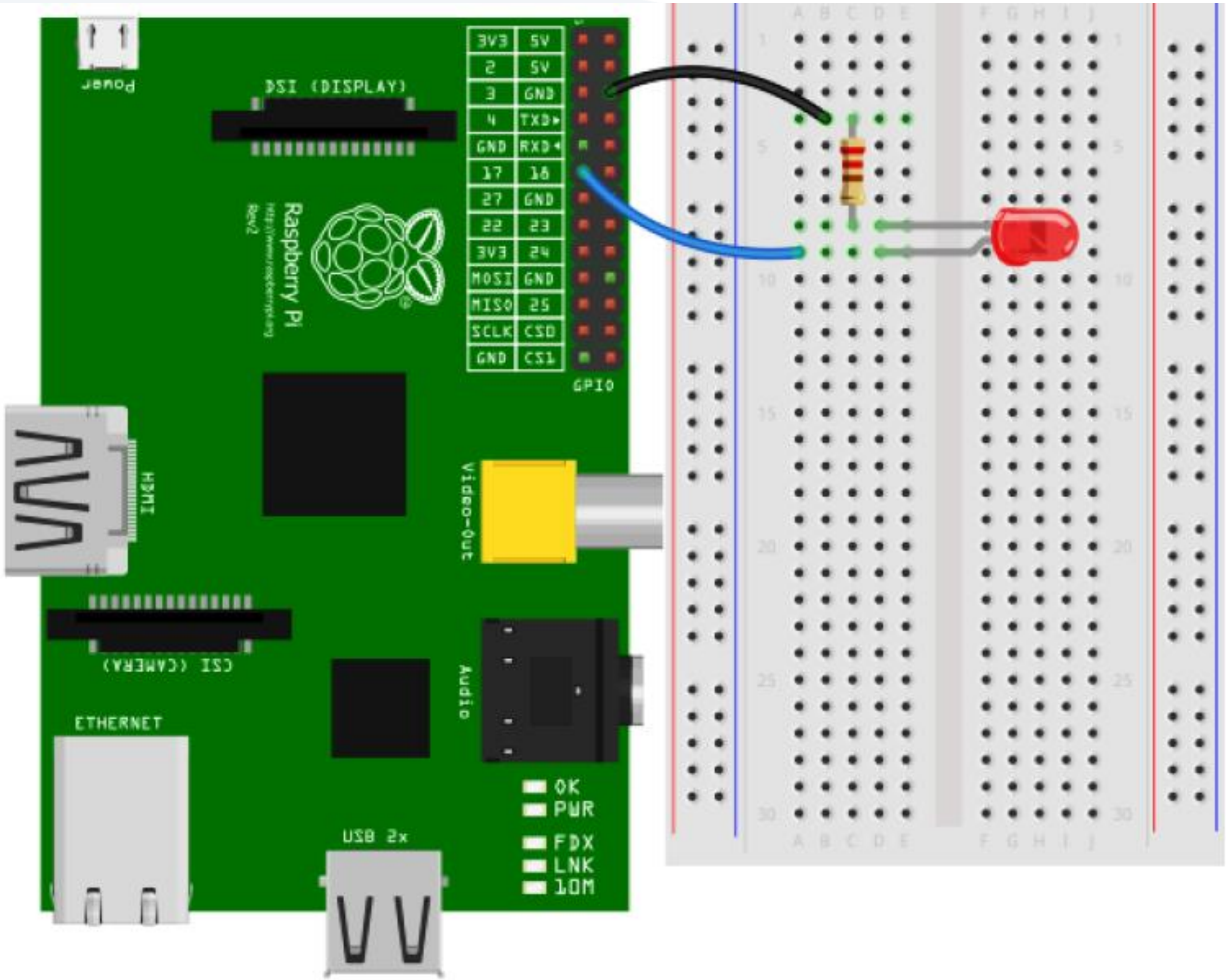
مميزات نظام لينكس

- مجاني ومفتوح المصدر.
- نظام آمن وموثوق به، فعند استخدامنا لهذا النظام لا داعي للقلق من الفيروسات.
- تحديثات سريعة جداً وذلك بفضل المجتمع العملاق من المطورين الذين يعملون على تطويره يومياً.
- وجود توزيعات وإصدارات مناسبة لآلاف الأجهزة الإلكترونية بدءاً من الحواسيب الخارقة إلى الحواسيب فائقة الصغر.

مخارج التحكم GPIO

- هي اختصار لعبارة General-purpose input/output (الأرجل العامة التي تستخدم كدخل أو خرج)، تمتلك لوحة الراسبيري عدد من أرجل التوصيل الكهربائية تسمى GPIO بعضها يستخدم في التحكم الإلكتروني (outputs) والبعض في الاستشعار (input) والبعض في امداد الطاقة والبعض مخصص لبروتوكولات الاتصالات بين المكونات والأجهزة الإلكترونية المختلفة.

المثال الأول: تشغيل وإطفاء ليد Blinking Led



الكود البرمجي

```
import time
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BOARD)
GPIO.setup(11, GPIO.OUT)

while True:
        GPIO.output(11,0)
        time.sleep(1)
        GPIO.output(11,1)
        time.sleep(1)
```

المثال الأول: شرح الكود

```
import time
import RPi.GPIO as GPIO
```

أمر استدعاء المكتبات: يمثل هذا الجزء بداية أي برنامج في معظم لغات البرمجة وهو إضافة المكتبة البرمجية time المسؤولة عن قياس الزمن و أو تنفيذ أمر معين لفترة محددة من الزمن، المكتبة الثانية هي RPi.GPIO و هي مكتبة التحكم في الـ GPIO الخاصة بالراسبيري باي.

```
GPIO.setmode(GPIO.BOARD)
```

هذا الأمر يستخدم في تفعيل جميع منافذ الـ GPIO و يجعلها جاهزة لاستقبال أوامر التحكم، كما يرتب المنافذ بناء على مكانها على لوحة الراسبيري باي.

```
GPIO.setup(11, GPIO.OUT)
```

هذا الأمر يحدد وظيفة أي Pinعلى الـ GPIO و يحدد هل ستعمل كمخرج OUT أم كمدخل IN ويمثل رقم 11 رقم الـ Pin التي نريد التحكم بها، مع ملاحظة انه في حالة استخدام اكثر من Pin يجب كتابة كيفية تشغيلها IN أو OUT في بداية البرنامج، على سبيل المثال نريد تشغيل الـ pin رقم 11 كمخرج و الـ pin رقم 23 كمدخل اذا سنكتب:

```
GPIO.setup(11, GPIO.OUT)
GPIO.setup(23, GPIO.IN)
```

```
while True:
```

عند كتابة الأمر while True فهذا يعني أن جميع الأوامر التي تكتب بعدها ستنفذ إلى ما لا نهاية أو حتى يتم اغلاق البرنامج أو اغلاق الراسبيري باي نفسها.

```
while True:
        command to do
        another thing to do
        another thing to do
```

يجب ترك مسافة قبل كل سطر نريد إدخاله داخل عملية التكرار و ذلك عن طريق الضغط على زر Tabفي الجانب الأيسر من لوحة المفاتيح.

المثال الأول: شرح الكود

GPIO.output(11,0)

تشغيل و اغلاق المخارج:يستخدم الأمر GPIO.output(pin, status) في تشغيل او إغلاق أي منفذ GPIO حيث نستبدل pin برقم المخرج المراد تشغيلة أو اطفائه و نستبدل status بحالة التشغيل و هي اما =1 و تعني تشغيل المنفذ (فرق الجهد = 3.3فولت) و اما =0 .صفر و تعني اغلاق المنفذ(فرق الجهد = صفر).

Time.sleep(1)

التحكم في زمن التشغيل و الإغلاق: يستخدم الأمر time.sleep(time) في تحديد زمن تنفيذ الأمر الذي يسبقة، فمثلاً اذا كان الأمر الذي يسبقة يشغل المنفذ رقم 11 و كتبنا time.sleep(5) فهذا يعني أن المنفذ رقم 11 سيظل يعمل لمدة 5 ثواني.

- يمكن كتابة الأمر GPIO.output(pin, status) على صورة True أو False بحيث تُمثل كلمة True تشغيل المخرج (بدلاً من 1) و تمثل كلمة False إغلاق المخرج (بدلاً من 0)، على سبيل المثال يمكننا تعديل البرنامج ليصبح كالتالي:

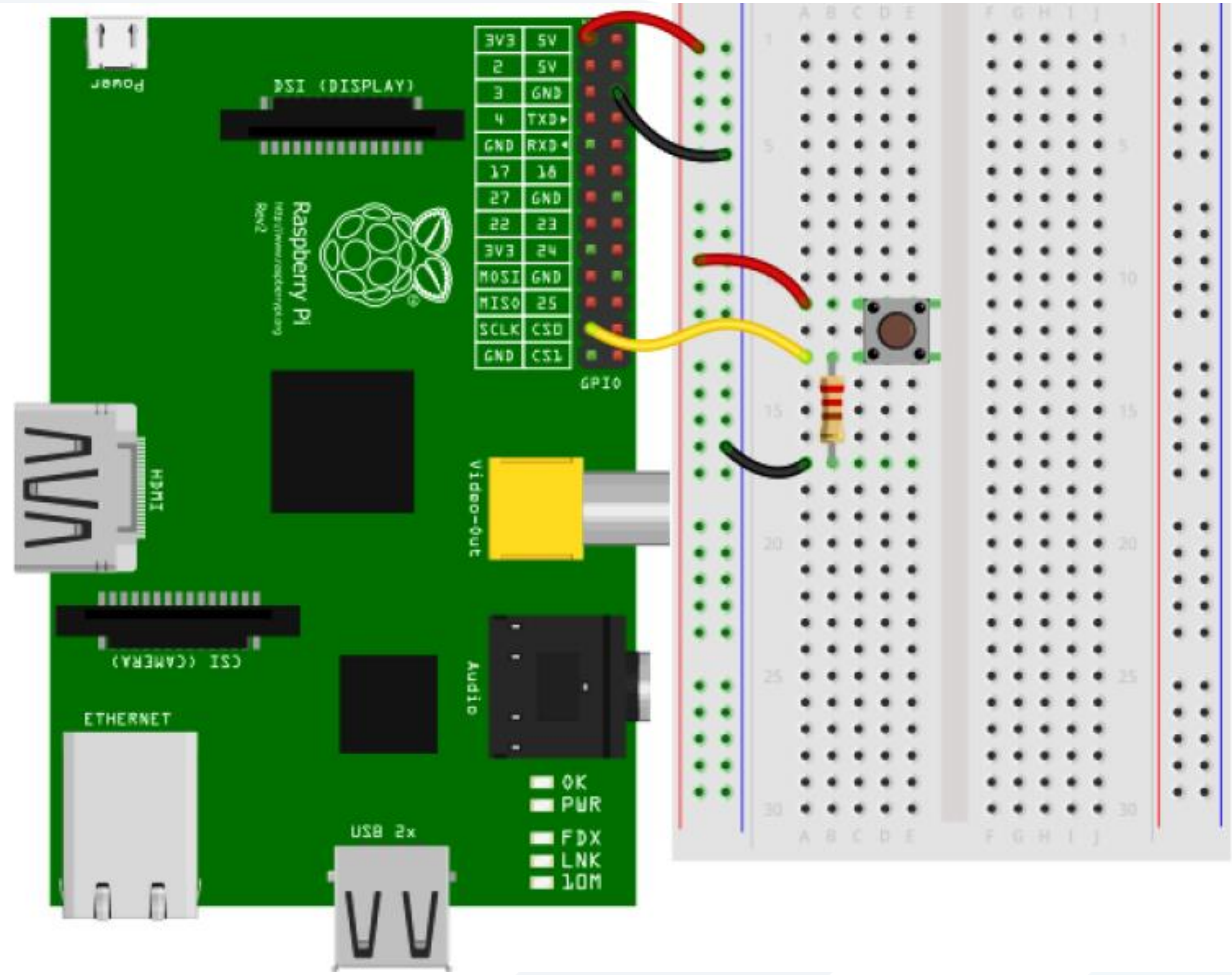
```
import time
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BOARD)
GPIO.setup(11, GPIO.OUT)
while True:
        GPIO.output(11, False)
        time.sleep(1)
        GPIO.output(11, True)
        time.sleep(1)
```

تطوير المثال الأول

- سنقوم بتطوير المثال الأول لكي يعرض رسالة على الشاشة تخبرنا بأن الليد يعمل الآن أو الليد مغلق، لعمل هذا التعديل سنضيف الأمر print مع الرسالة التي نريد عرضها ليصبح الكود كالتالي:

```
import time
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BOARD)
GPIO.setup(11, GPIO.OUT)
while True:
        GPIO.output(11,False)
        print "Led on Pin 11 is now OFF (output = zero volt)"
        time.sleep(1)

        GPIO.output(11,True)
        print "Led on Pin 11 is now ON (output = 3.3 volt)"
        time.sleep(1)
```

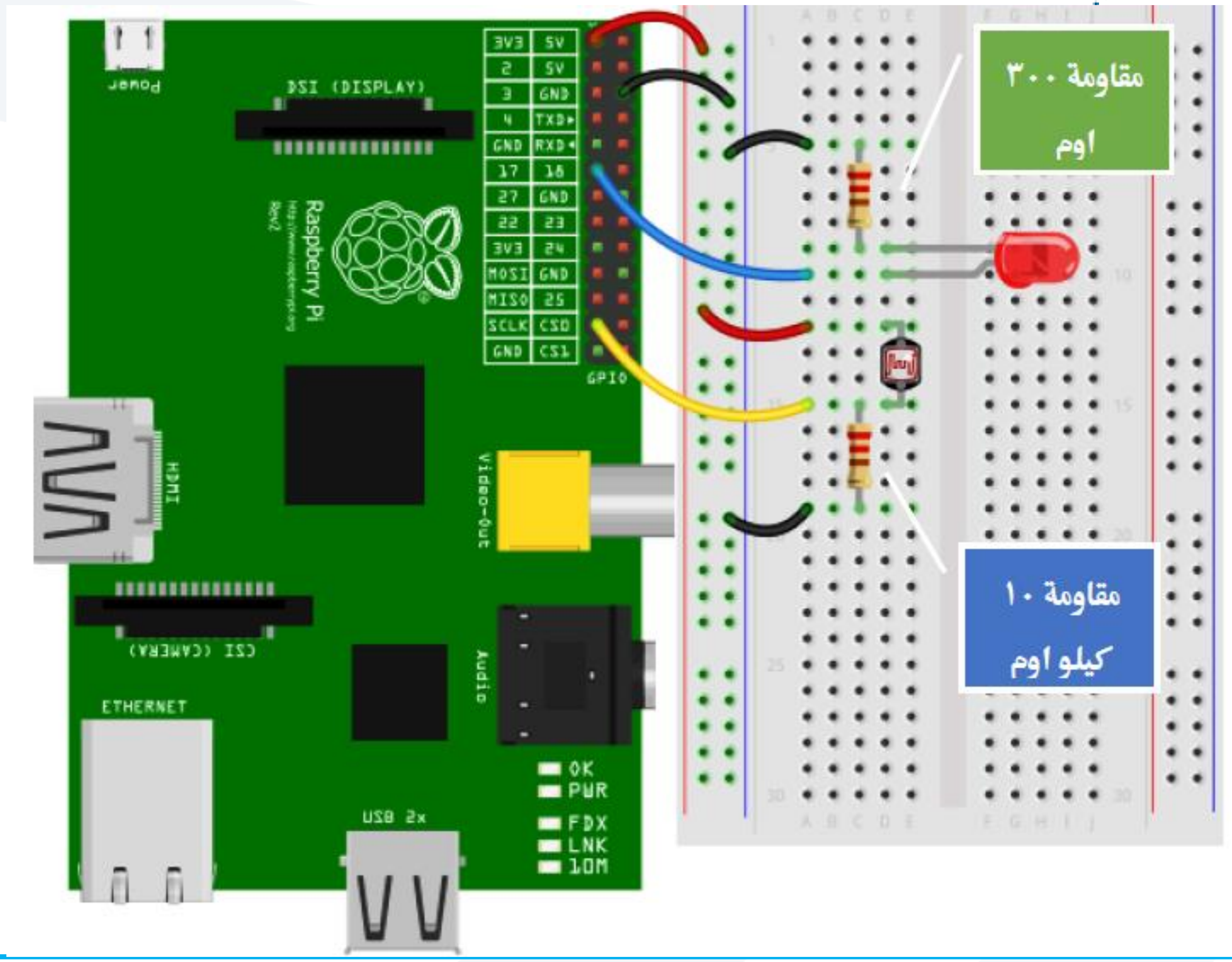
المثال الثاني: قراءة دخل من مفتاح (سويتش)



الكود البرمجي

```
import time
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BOARD)
GPIO.setup(23, GPIO.IN)
while True:
	if (GPIO.input(23)==True):
		print "'Input is True (3.3 volt)"
	else:
		print "'Input is False (zero volt)"
	time.sleep(1)
```

المثال الثالث: تشغيل LED مع المقاومة الضوئية



الكود البرمجي:

```
import time
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BOARD)
GPIO.setup(23, GPIO.IN)
GPIO.setup(11, GPIO.OUT)

while True:
	if (GPIO.input(23) == True):
		GPIO.output(11,0)
	else:
		GPIO.output(11,1)
```

تطوير للمثال الثالث

تطوير 1

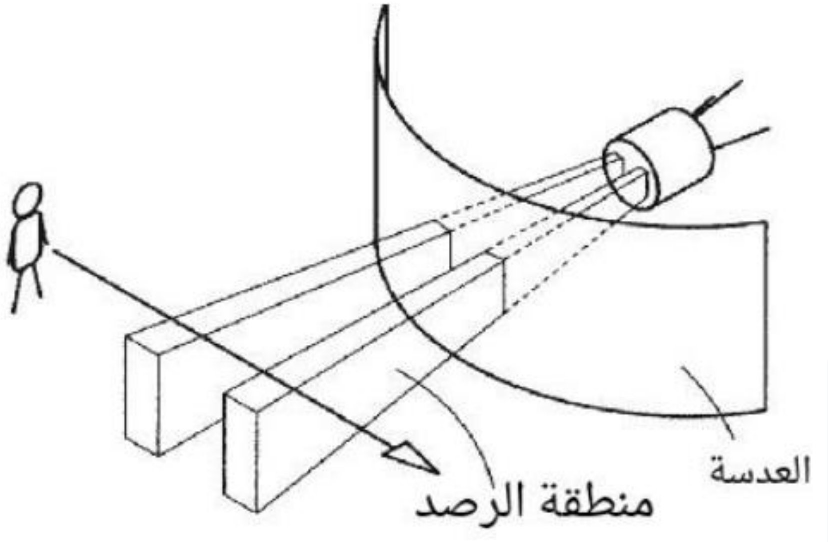
```
import time
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BOARD)
GPIO.setup(23, GPIO.IN)
GPIO.setup(11, GPIO.OUT)

while True:
        if (GPIO.input(23) == False):
                GPIO.output(11,0)
        else:
                GPIO.output(11,1)
```

تطوير 2

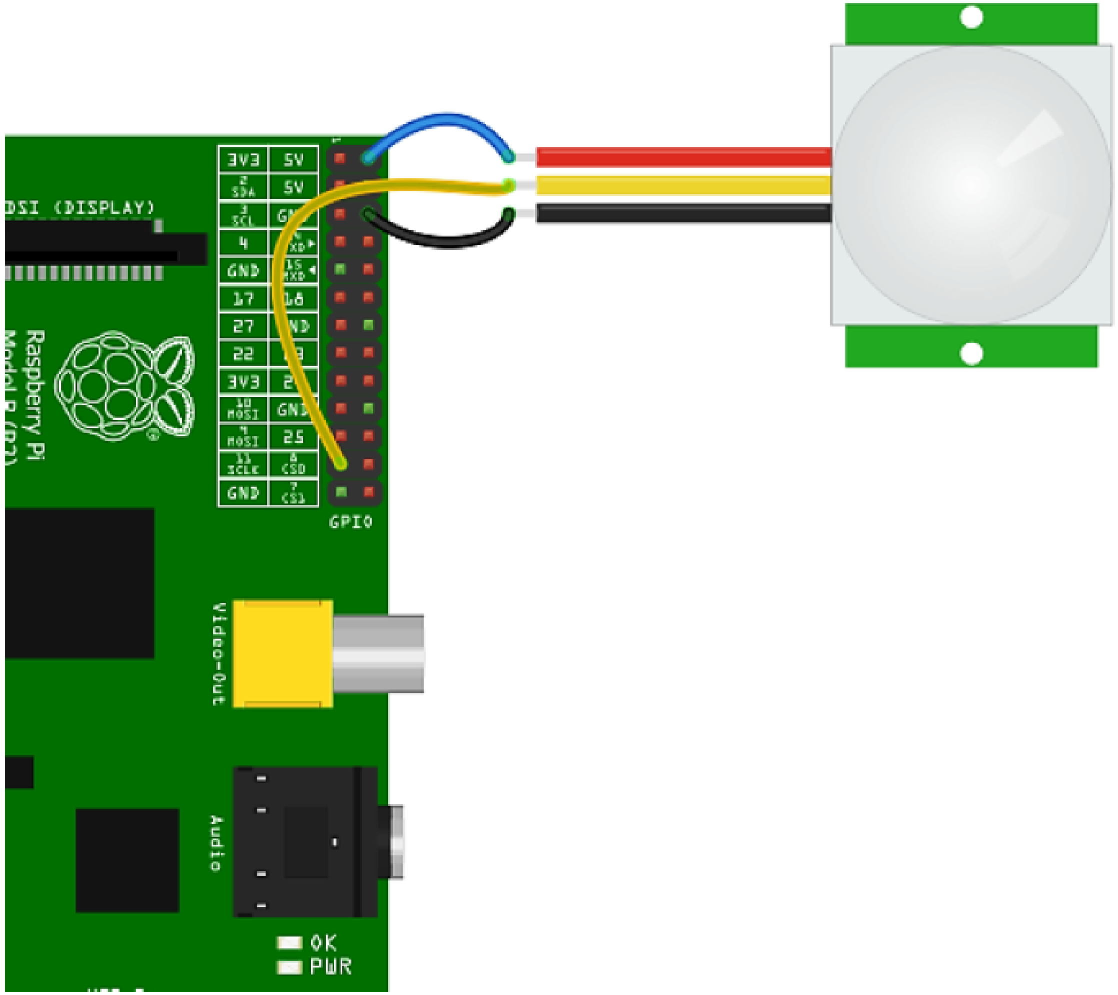
```
import time
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BOARD)
GPIO.setup(23, GPIO.IN)
GPIO.setup(11, GPIO.OUT)

while True:
        if (GPIO.input(23) == False):
                GPIO.output(11,0)
                print " Led on pin 11 is now OFF "
        else:
                GPIO.output(11,1)
                print " Led on pin 11 is now ON "

time.sleep(1)
```

المثال الرابع: راصد الحركة PIR Motion Detector

- تعتبر راصدات الحركة من أهم الحساسات المستخدمة في الحياة العملية لعدة أغراض أشهرها الحماية والأبواب الذكية، فنراها دائماً في المحلات التجارية والمولات الكبيرة وتستخدم أيضاً على أبواب الكراجات، و هذا ما يفسر كيف تفتح الأبواب الزجاجية للمحلات التجارية بمجرد أن نقترب منها؟؟السر يكمن في استخدام راصدات الحركة والتي تستطيع أن تستشعر قدوم شخص ما ورصد تحركاته في نطاق محدد يصل إلى عدة مترات وفي بعض الراصدات المتطورة يمكن استشعار الحركة من على بعد مئات المترات.



المثال الرابع: راصد الحركة PIR Motion Detector

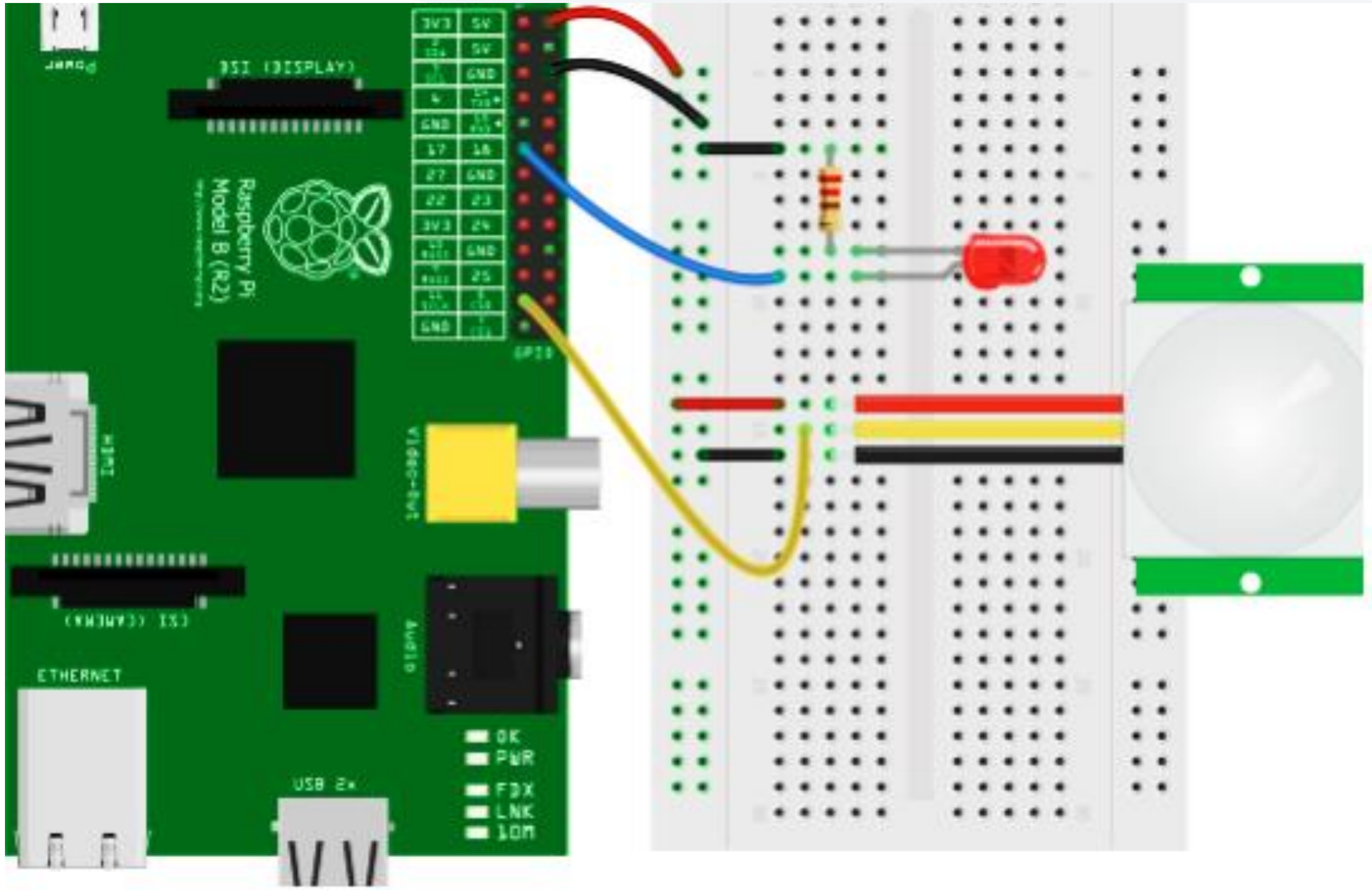


الكود البرمجي:

```
import time
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BOARD)
GPIO.setup(23, GPIO.IN)

while True:
        if (GPIO.input(23) == True):
                pritn " Motion Detected "
        else:
                pritn " There is No Motion "
        time.sleep(1)
```

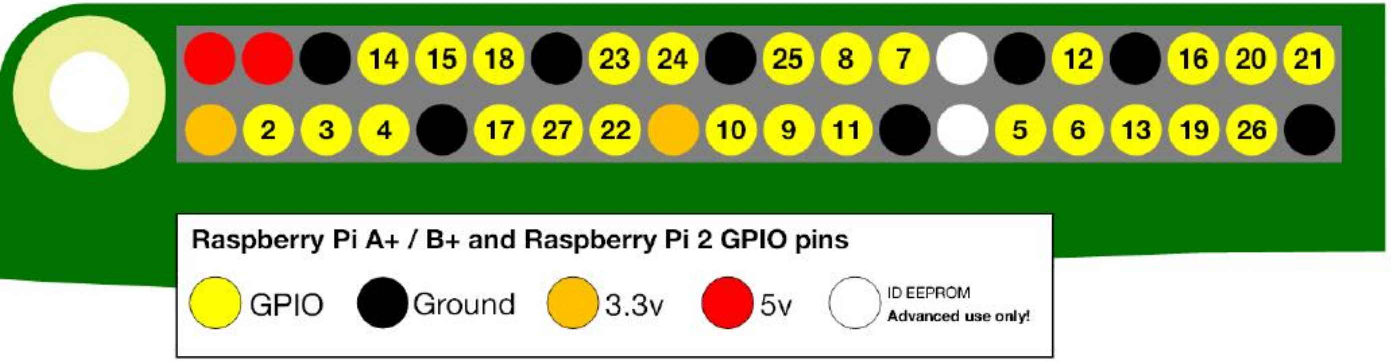
تطوير المثال الرابع

- لتطوير المثال الرابع سنقوم بإضافة LED مع مقاومة و سنقوم بتعديل الكود البرمجي لتشغيل الLED لمدة 3 ثواني عند لقط أي حركة:




```
import time
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BOARD)
GPIO.setup(23, GPIO.IN)
GPIO.setup(11, GPIO.OUT)

while True:
        if (GPIO.input(23) == True):
                GPIO.output(11,1)
                print " Motion Detected, Now Turning On Led "
                time.sleep(3)

        else:
                GPIO.output(11,0)
                print " There is No motion "

time.sleep(1)
```

ترقيم BCM بدلاً من ترقيم المنافذ بالترتيب



- هناك نوعان من ترقيم المنافذ في لوحة راسبيري باي وهما ترقيم المنافذ تبعا لمكانها على اللوحة ويسمى (ترقيم BOARD) وترقيم اللوحات تبعا لشريحة BCM(الشرحية الإلكترونية التي تصنعها Broadcom وتحتوي على أغلب مكونات الراسبيري).
- أين الاختلاف؟ الاختلاف الوحيد بين الترقيمين سيكون في طريقة استدعاء مكتبة GPIO في لغة بايثون، فمثلا اذا اردنا تشغيل منافذ التحكم للراسبيري بترتيب GPIO نستخدم الأمر التالي:

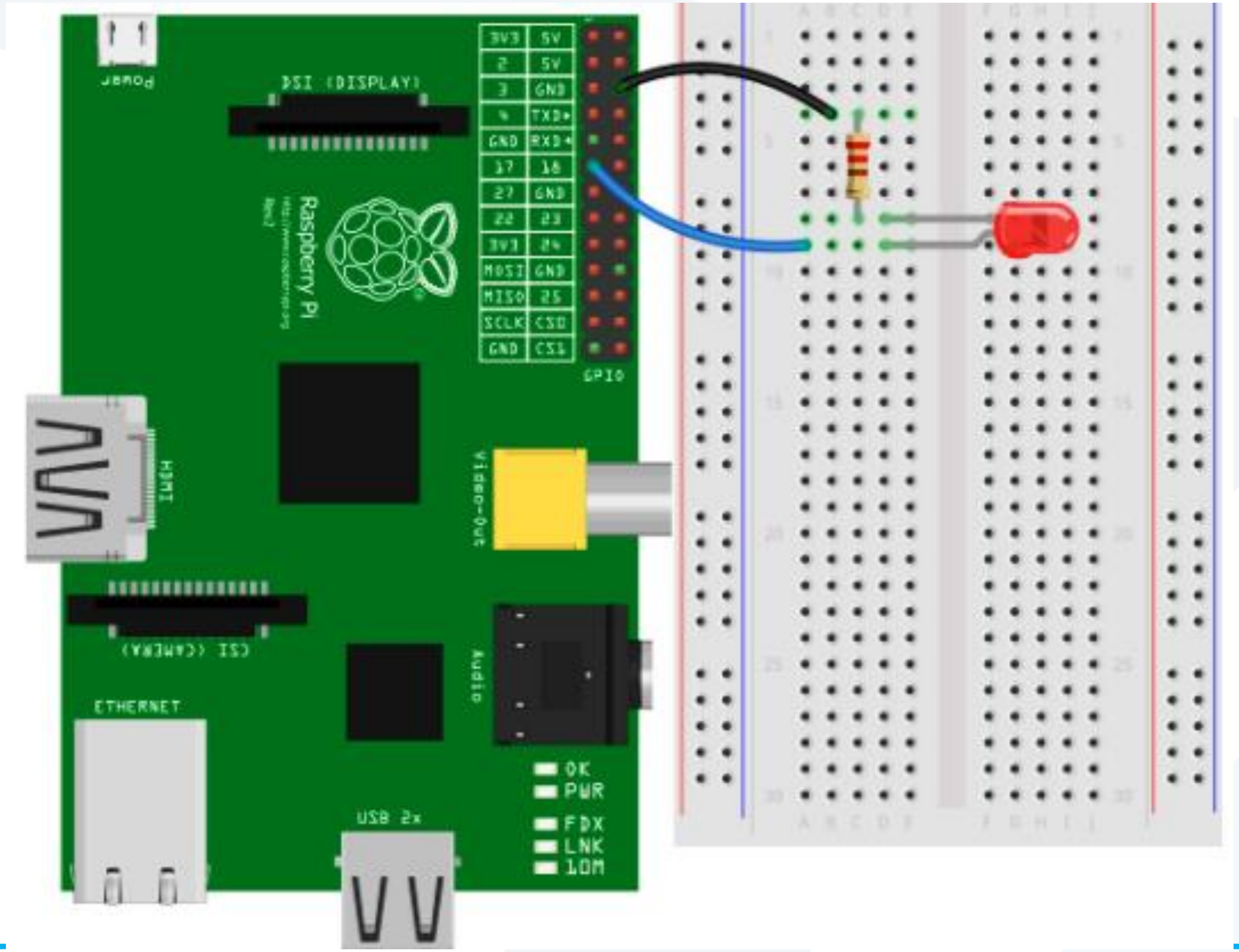
GPIO.setMode(GPIO.BOARD)

- واذا أردنا تشغيل المنافذ بترقيم BCM نستخدم الأمرالتالي: GPIO.setmode(GPIO.BCM)

تقنية تعديل عرض النبضة PWM

- تدعم لوحة راسبيري باي نوعين من الـ PWM وهما Hardware PWM و Software PWM النوع الأول هو استخدام الأرجل المعدة خصيصاً لتوليد نبضات PWM، والطريقة الثانية هي عبر برمجة أي رجل GPIO عن طريق لغة بايثون ومكتبة التحكم في المنافذ، حيث نستطيع أن نحول هذه الرجل من مخرج تقليدي الى مخرج PWM .

المثال الخامس: التحكم في إضاءة ليد عن طريق PWM



الكود البرمجي:

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BOARD)
GPIO.setup(11, GPIO.OUT)

p = GPIO.PWM(11, 50)
p.start(5)

p.ChangeDutyCycle(10)
time.sleep(3)
p.ChangeDutyCycle(40)
time.sleep(3)
p.ChangeDutyCycle(80)
time.sleep(3)
p.ChangeDutyCycle(100)
time.sleep(3)

p.stop()
GPIO.cleanup()
```

المثال الخامس: شرح الكود

في الأسطر الأربعة الأولى استخدمنا نفس الأوامر كما في الأمثلة السابقة لاستدعاء مكتبة التحكم في الـ GPIO ومكتبة قياس الزمن time، ثم استخدمنا مجموعة من الأوامر للتحكم في الـ PWM كالتالي:

```
p = GPIO.PWM(11, 50)
```

في هذا الأمر نخبر لوحة الراسبيري بأن تقوم بتفعيل الرجل رقم 11 في وضع الـ PWM وبتردد 50 هرتز

كما سنسمي هذا المنفذ بالاسم p في ذات الوقت، وهذا يعطينا القدرة على التحكم في هذا المخرج وخصائصه بكتابة p بدلا من رقم المخرج نفسة.

```
p.start(5)
```

هذا الأمر يعطي إشارة للراسبيري بأن تبدأ تشغيل المخرج وبقدرة Duty Cycle = 5%

```
p.ChangeDutyCycle(10)
time.sleep(3)
```

- يستخدم الأمر (10) p.ChangeDutyCycle في تغيير ال Duty Cycle لنبضات الـ PWM و الذي يعني تحديد قيمة فرق الجهد الخارجة من المنفذ وفي حالتنا 10 تعني 10% ثم يأتي الأمر (3) time.sleep و الذي يعني أن الراسبيري ستظل تولد نبضات PWM ب 10% Duty cycle لمدة 3 ثواني.

المثال الخامس: شرح الكود

```
p.ChangeDutyCycle(40)
time.sleep(3)
p.ChangeDutyCycle(80)
time.sleep(3)
p.ChangeDutyCycle(100)
time.sleep(3)
```

هذه الأوامر مثل السابقة بالضبط والاختلاف هنا هو زيادة الـ Duty Cycleبالتدريج من 10٪ الى 40٪ إلى 80٪ ثم إلى 100٪ والجدول التالي يوضح اختلاف إضاءة الدايو الضوئي في كل من هذه الحالات.

```
p.stop()
GPIO.cleanup()
```

يستخدم الأمر p.stop() في إيقاف تشغيل الـ PWM والأمر GPIO.cleanup() في اغلاق المنافذ كلها.