**Using Sensors**

|  |
| --- |
| The official documentation is [**HERE**](http://python-ev3dev.readthedocs.io/en/latest/sensors.html) and [**HERE**](http://www.ev3dev.org/docs/sensors/). See also the [**Sensor Modes page**](https://sites.google.com/site/ev3python/learn_ev3_python/using-sensors/sensor-modes).  **EV3 Python is compatible with all the standard EV3 and NXT sensors except the NXT light sensor cannot be used to detect colors (as of September 2016).**  EV3 and/or NXT sensors can be attached to any of the EV3's four sensor ports but I will adhere to the useful convention that they should be attached as follows:  **port 1 = touch, port 2 = gyro, port 3 = color, port 4 = infrared or ultrasonic**.  This convention is useful because it means you don't have to reconnect the sensors as you move from one EV3 tutorial to another.  if you don’t attach more than one sensor of a particular type, then it will not matter which port a sensor is plugged in to. You don't need to include any reference on your code to the sensor port number - your program will just work. However, if you connect more than one sensor of a given type then you need to specify the port number with 'in1', 'in2'etc or look at the port\_name attribute.  As usual, I only mention what I think is most likely to be useful to you as a beginner. For sensors, that would include InfraredSensor(), TouchSensor(), UltrasonicSensor, ColorSensor(), LightSensor(), GyroSensor(), RemoteControl(), SoundSensor(), value(), mode() and units. Note that the EV3 'color' sensor corresponds to the NXT 'light' sensor.  As in the standard Lego EV3 software, many sensors can be used in different modes. Information about the different modes available for different Lego sensors can be found in the Lego section of the sensor-specific-resources section of [**THIS PAGE**](http://www.ev3dev.org/docs/sensors/#sensor-specific-resources) and selected information from that page is [**HERE**](https://sites.google.com/site/ev3python/learn_ev3_python/using-sensors/sensor-modes).To set the desired mode for the sensor you are using, use mode as in the first example below. For example, if a single IR sensor is attached to any sensor port and named ir then you can set it to proximity mode with ir.mode = 'IR-PROX' . The touch sensor does not have multiple modes, of course, but for the other sensors it is important to always set the mode of the sensor before it is used.  The reading from every sensor in every mode (except the color sensor in RGB-RAW mode) is obtained with the value() method (a 'method' is a special kind of function). Don't forget to include the parentheses.  **Example 1 (touch sensor & IR sensor)**  The example below uses the touch sensor and the IR sensor. It continuously displays the distance value detected by the IR sensor, along with the corresponding units (**pct**, meaning percent,for the IR sensor). It turns the left LED from green to red when an object is brought close to the IR sensor, and turns it back to green when no object is close. The while loop runs as long as the touch sensor button is NOT pressed. When the touch sensor is pressed the loop is exited, a beep is sounded and the left LED is set to green (it might have been red when the loop was exited).  The example uses the assert statement to check that the sensors are attached. If the assertion fails then the program closes with a useful error message (in this case a text message). Assert is a statement and not a function, so it should be used without parentheses as here.  #!/usr/bin/env python3  # so that script can be run from Brickman  from ev3dev.ev3 import \*  # Connect infrared and touch sensors to any sensor ports  # and check they are connected.  ir = InfraredSensor()  assert ir.connected, "Connect a single infrared sensor to any sensor port"  ts = TouchSensor();    assert ts.connected, "Connect a touch sensor to any port"  # can have 2 statements on same line if use semi colon  # Put the infrared sensor into proximity mode.  ir.mode = 'IR-PROX'  while not ts.value():    # Stop program by pressing touch sensor button      # Infrared sensor in proximity mode will measure distance to the closest      # object in front of it.      distance = ir.value()      if distance < 60:          Leds.set\_color(Leds.LEFT, Leds.RED)      else:          Leds.set\_color(Leds.LEFT, Leds.GREEN)  Sound.beep()  Leds.set\_color(Leds.LEFT, Leds.GREEN)  #make sure left led is green before exiting  **Example 2 (touch sensor and EV3 ultrasonic sensor)**  An ultrasonic sensor is included with the educational version of the EV3 kit but not with the home version (it can be bought separately). If you have a single EV3 ultrasonic sensor attached to the EV3 then you can modify the code of Example 1 to make it work with the EV3 US sensor as shown below. When you set the EV3 US sensor mode to US-DIST-CM as below you expect the sensor to return a value in cm, of course, and in a sense it does but it is one of several modes (see the [**sensor modes page**](https://sites.google.com/site/ev3python/learn_ev3_python/using-sensors/sensor-modes)) that also **includes a first decimal place (but no decimal point)** in the returned value. That means the value returned really represents mm rather cm. For example, if the sensor returns the value 90 cm it actually means 9.0 cm (or 90 mm) since the value includes one decimal place! Obviously, you need to take great care handling the values returned by these modes. To deal with this problems, I have included a division by ten in the code below to convert the distance from mm to cm. Thus the code below will make the color of the left LED change to red when an object is within 60.0 cm (600 mm) of the US sensor.  Similarly, if you set the mode to US-DIST-IN  you expect the value to be in inches but in fact it is in *tenths*of an inch so you need to divide the value by 10 to get inches. Or, put another way, the value *is* in inches but includes the first decimal place of the value. Confusing, isn't it? If you use inches you will probably want to use a distance less than 60.0 inches for the led to switch to red. Note that the problem with the EV3 ultrasonic sensor is NOT shared by the NXT ultrasonic sensor, which simply gives the reading as an integer (zero decimal places).  #!/usr/bin/env python3  # so that script can be run from Brickman  from ev3dev.ev3 import \*  # Connect ultrasonic and touch sensors to any sensor port  # and check they are connected.  **us = UltrasonicSensor()**  **assert us.connected, "Connect a single US sensor to any sensor port"**  ts = TouchSensor();    assert ts.connected, "Connect a touch sensor to any port"  # can have 2 statements on same line if use semi colon  **# Put the US sensor into distance mode.**  **us.mode='US-DIST-CM'**  **units = us.units**  **# reports 'cm' even though the sensor measures 'mm'**  while not ts.value():    # Stop program by pressing touch sensor button      # US sensor will measure distance to the closest      # object in front of it.  **distance = us.value()/10  # convert mm to cm**      print(str(distance) + " " + units)      if distance < 60:  #This is an inconveniently large distance          Leds.set\_color(Leds.LEFT, Leds.RED)      else:          Leds.set\_color(Leds.LEFT, Leds.GREEN)  Sound.beep()  Leds.set\_color(Leds.LEFT, Leds.GREEN)  #set left led green before exiting  **Example 3 (two touch sensors)**  Usually you will not have two sensors of the same type plugged into the EV3 and you will not have to specify or determine where the sensors are attached. But in this example let's assume we have TWO touch sensors plugged into the EV3. We will SPECIFY which ports the sensors are plugged into using 'in1' and 'in2' for sensor (input) ports 1 and 2. In this program the left led is set to green if the touch sensor on port 1 is not pressed and red if it is pressed (more specifically, if the touch sensor is not pressed then it returns a value of zero and the code then uses item zero from the tuple). In a similar way the touch sensor on port 2 controls the right led. To stop this program you will need to press Ctrl-C to interrupt the while loop.  #!/usr/bin/env python3  # so that script can be run from Brickman  from ev3dev.ev3 import \*  from time   import sleep  # Connect TWO touch sensors to BOTH sensor ports 1 and 2  # and check they are both connected.  ts**1** = TouchSensor('in**1**')  assert ts1.connected, "Connect a touch sensor to sensor port 1"  ts**2** = TouchSensor('in**2**')  assert ts2.connected, "Connect a touch sensor to sensor port 2"  while True:    # Stop this program with Ctrl-C      Leds.set\_color(Leds.**LEFT**, (Leds.GREEN, Leds.RED)[ts**1**.value()])      Leds.set\_color(Leds.**RIGHT**, (Leds.GREEN, Leds.RED)[ts**2**.value()])  # Stop program with Ctrl-C  **Example 4 (gyro sensor and touch sensor)**  The gyro sensor is included with the educational version of the EV3 kit but not with the home version (it can be bought separately). It measures the angle in degrees, giving the value of zero degrees to the orientation of the sensor when the program is started. Note that it is vital that the gyro sensor be absolutely still when the program is launched otherwise the reading will wander later even when the sensor is held still.  The sensor can also measure the *rate of change* of the angle, in degrees per second, but as a beginner you are likely to find that less useful.  The program below prints the measured angle to the console, then plays for one second a tone whose frequency depends on the angle, then waits for 0.5 second before repeating the cycle. Press the touch sensor button for at least a second to stop the program.  #!/usr/bin/env python3  # so that script can be run from Brickman  from ev3dev.ev3 import \*  from time import sleep  # Connect gyro and touch sensors to any sensor port  # and check they are connected.  gy = GyroSensor()  assert gy.connected, "Connect a single gyro sensor to any sensor port"  ts = TouchSensor();    assert ts.connected, "Connect a touch sensor to any port"  # can have 2 statements on same line if use semi colon  # Put the gyro sensor into ANGLE mode.  gy.mode='GYRO-ANG'  units = gy.units  #reports 'deg' meaning degrees  while not ts.value():    # Stop program by pressing touch sensor button      angle = gy.value()      print(str(angle) + " " + units)      Sound.tone(1000+angle\*10, 1000).wait()      sleep(0.5)  Sound.beep()  **Example 5: Color sensor in COL-REFLECT mode**  In this script the color sensor emits light and measures the reflected light intensity, returning twice every second a value which is theoretically between 0 and 100. For best results, place the sensor about 3 mm from the reflecting surface. Using that separation, I get a value of about 80 with normal white paper and about 5 with a typical black surface.  #!/usr/bin/env python3  from ev3dev.ev3 import \*  from time import sleep  # Connect EV3 color sensor and check connected.  cl = ColorSensor()  assert cl.connected, "Connect a color sensor to any sensor port"  # Put the color sensor into COL-REFLECT mode  # to measure reflected light intensity.  # In this mode the sensor will return a value between 0 and 100  cl.mode='COL-REFLECT'  while True:      print(cl.value())      sleep(0.5)    # I get max 80 with white paper, 3mm separation  # and 5 with black plastic, same separation  See also lesson 6 on [**this page**](https://sites.google.com/site/ev3python/learn_ev3_python/basics)**.**  **Example 6: EV3 color sensor in COL-AMBIENT mode, touch sensor and large motor**  In this program the color sensor measures ambient light intensity, returning a value between 0 and 100. This value is used to control the speed of a large motor attached to port B, so the brighter the ambient light the faster the motor will turn. Press the touch sensor button to stop the program. To make this program work with *reflected*light intensity rather than *ambient*light intensity, just replace COL-AMBIENT with COL-REFLECT.  #!/usr/bin/env python3  # so that script can be run from Brickman  from ev3dev.ev3 import \*  from time import sleep  # Connect EV3 color sensor and touch sensor  # and check they are connected.  cl = ColorSensor()  assert cl.connected, "Connect an EV3 color sensor to any sensor port"  ts = TouchSensor()  assert ts.connected, "Connect a touch sensor to any sensor port"  # Connect a large motor to port B and check it is connected.  m = LargeMotor('outB')  assert m.connected, "Connect a large motor to port B"  # Put the color sensor into COL-AMBIENT mode  # to measure ambient light intensity.  # In this mode the sensor will return a value between 0 and 100  cl.mode='COL-AMBIENT'  # run\_forever command will allow us to vary motor  # performance on the fly by adjusting speed\_sp attribute.  m.run\_forever(speed\_sp = 0)  while not ts.value():    # Stop program by pressing touch sensor button  # set the motor's speed set point to be equal to  # the measured ambient light intensity value      m.speed\_sp = cl.value()  Sound.beep()  **Example 7: EV3 color sensor in COL-COLOR mode and touch sensor**  When the EV3 color sensor is in COL-COLOR mode it tries to recognise the color of standard Lego bricks placed about 5-6mm in front of the sensor (the distance is critical) and returns a corresponding integer value between 0 (unknown) and 7 (brown). The program below reads the integer once per second, converts it into the corresponding text string using a tuple and displays the string in the console. Press the touch sensor button for at least a second to stop the program.  To make the program speak the colors as well as displaying their text strings, uncomment the line highlighted in blue. This will slightly increase the time between measurements as I included a wait() function to ensure that the speech was not interrupted.  #!/usr/bin/env python3  # so that script can be run from Brickman  from ev3dev.ev3 import \*  from time   import sleep  # Connect EV3 color sensor to any sensor port  # and check it is connected.  cl = ColorSensor()  assert cl.connected, "Connect a single EV3 color sensor to any sensor port"  # Connect touch sensor to any sensor port  # and check it is connected.  ts = TouchSensor();    assert ts.connected, "Connect a touch sensor to any port"  # you can have 2 statements on the same line if you use a semi colon  # Put the color sensor into COL-COLOR mode.  cl.mode='COL-COLOR'  colors=('unknown','black','blue','green','yellow','red','white','brown')  while not ts.value():    # Stop program by pressing touch sensor button      print(colors[cl.value()])      #Sound.speak(colors[cl.value()]).wait()      sleep(1)  Sound.beep()  **Example 8: EV3 color sensor in RGB-RAW mode and touch sensor**  This mode is special since it returns 3 values simultaneously, representing the amounts of red, green and blue reflected light (any color can be obtained by mixing the correct amounts of red, green and blue light). These values can be referred to in code as value(0), value(1) and value(2) respectively. The following program prints the three values detected once every second. Press the touch sensor button for at least a second to stop the program. For more information see [**THIS PAGE**](https://sites.google.com/site/ev3python/learn_ev3_python/using-sensors/sensor-modes/color-sensor-mode-4-rgb).  #!/usr/bin/env python3  # so that script can be run from Brickman  from ev3dev.ev3 import \*  from time   import sleep  # Connect EV3 color sensor to any sensor port  # and check it is connected.  cl = ColorSensor()  assert cl.connected, "Connect a single EV3 color sensor to any sensor port"  # Connect touch sensor to any sensor port  # and check it is connected.  ts = TouchSensor();  assert ts.connected, "Connect a touch sensor to any port"  # you can have 2 statements on the same line if you use a semi colon  # Put the color sensor into RGB mode.  cl.mode='RGB-RAW'  while not ts.value():    # Stop program by pressing touch sensor button      red = cl.value(0)      green=cl.value(1)      blue=cl.value(2)      print("Red: " + str(red) + ", Green: " + str(green) + ", Blue: " + str(blue))      sleep(1)  Sound.beep() |

**Usando sensores**

|  |
| --- |
| La documentación oficial está [**AQUÍ**](http://python-ev3dev.readthedocs.io/en/latest/sensors.html)  y [**AQUÍ**](http://www.ev3dev.org/docs/sensors/) . Ver también la [**página de modos de sensor**](https://sites.google.com/site/ev3python/learn_ev3_python/using-sensors/sensor-modes) .  **EV3 Python es compatible con todos los sensores estándar EV3 y NXT, excepto que el sensor de luz NX T no se puede utilizar para detectar colores (a partir de septiembre de 2016) .**  Los sensores EV3 y / o NXT se pueden conectar a cualquiera de los cuatro puertos del sensor del EV3, pero cumpliré la convención práctica de que deben adjuntarse de la siguiente manera:  **puerto 1 = contacto, puerto 2 = giroscopio, puerto 3 = color, puerto 4 = infrarrojo o ultrasónico** .  Esta convención es útil porque significa que no tiene que volver a conectar los sensores a medida que pasa de un tutorial de EV3 a otro.  Si no conecta más de un sensor de un tipo particular, entonces no importará a qué puerto está enchufado un sensor. No necesita incluir ninguna referencia en su código al número de puerto del sensor; su programa simplemente funcionará. Sin embargo, si conecta más de un sensor de un tipo determinado, deberá especificar el número de puerto con 'in1' , 'in2', etc. o consultar el atributo port\_name .  Como de costumbre, solo menciono lo que creo que es más útil para usted como principiante. Para los sensores, eso incluiría  InfraredSensor (), TouchSensor (), UltrasonicSensor, ColorSensor (), LightSensor (), GyroSensor (), RemoteControl (), SoundSensor (),  value () , mode () y unidades . Tenga en cuenta que el sensor EV3 'color' corresponde al sensor 'ligero' NXT.  Al igual que en el software Lego EV3 estándar, muchos sensores se pueden usar en diferentes modos. Puede encontrar información sobre los diferentes modos disponibles para diferentes sensores Lego en la sección Lego de la sección de recursos específicos del sensor de [**ESTA PÁGINA**](http://www.ev3dev.org/docs/sensors/#sensor-specific-resources)  y la información seleccionada de esa página está [**AQUÍ**](https://sites.google.com/site/ev3python/learn_ev3_python/using-sensors/sensor-modes) .Para configurar el modo deseado para el sensor que está utilizando, use el modo  como en el primer ejemplo a continuación. Por ejemplo, si un único sensor de IR está conectado a un puerto de sensor y recibe el nombre de ir, entonces puede establecerlo en el modo de proximidad con  ir.mode = 'IR-PROX'  . El sensor táctil no tiene múltiples modos, por supuesto, pero para los otros sensores es importante establecer siempre el modo del sensor antes de usarlo.  La lectura de cada sensor en cada modo (excepto el sensor de color en modo RGB-RAW) se obtiene con el   método de valor () (un "método" es un tipo especial de función). No te olvides de incluir los paréntesis.  **Ejemplo 1 (sensor táctil y sensor IR)**  El siguiente ejemplo usa el sensor táctil y el sensor IR. Muestra continuamente el valor de distancia detectado por el sensor IR, junto con las unidades correspondientes ( **pct** , que significa porcentaje, para el sensor IR). Gira el LED izquierdo de verde a rojo cuando un objeto se acerca al sensor IR y lo vuelve a poner verde cuando no hay ningún objeto cerca. El tiempo bucle se ejecuta mientras el botón del sensor táctil no está presionado. Cuando se presiona el sensor táctil, se sale del ciclo, se emite un pitido y el LED izquierdo se pone en verde (podría haber sido rojo cuando se salió del circuito).  El ejemplo usa la aserción declaración para comprobar que los sensores están unidos.Si la afirmación falla, el programa se cierra con un mensaje de error útil (en este caso, un mensaje de texto). Assert es una declaración y no una función, por lo que debe usarse sin paréntesis como aquí.  #! / usr / bin / env python3  # para que el script pueda ejecutarse desde Brickman  de la importación ev3dev.ev3 \*  # Conecte el infrarrojo y toque los sensores a cualquier puerto de sensor  # y verificar que estén conectados.  ir = InfraredSensor ()  assert ir.connected, "Conecta un solo sensor de infrarrojos a cualquier puerto del sensor"  ts = TouchSensor (); assert ts.connected, "Conecta un sensor táctil a cualquier puerto"  # puede tener 2 instrucciones en la misma línea si usa semi colon  # Ponga el sensor de infrarrojos en modo de proximidad.  ir.mode = 'IR-PROX'  while not ts.value (): # Parar el programa presionando el botón táctil      # El sensor infrarrojo en el modo de proximidad medirá la distancia al más cercano      # objeto en frente de él.      distance = ir.value ()      si la distancia <60:          Leds.set\_color (Leds.LEFT, Leds.RED)      más:          Leds.set\_color (Leds.LEFT, Leds.GREEN)  Sound.beep ()  Leds.set\_color (Leds.LEFT, Leds.GREEN)  # Asegúrese de que el led izquierdo esté verde antes de salir  **Ejemplo 2 (sensor táctil y sensor ultrasónico EV3)**  Se incluye un sensor ultrasónico con la versión educativa del kit EV3 pero no con la versión hogareña (se puede comprar por separado). Si tiene un solo sensor ultrasónico EV3 conectado al EV3, entonces puede modificar el código del Ejemplo 1 para que funcione con el sensor EV3 US como se muestra a continuación. Cuando configura el modo de sensor US EV3 en  US-DIST-CM  como se muestra a continuación, espera que el sensor devuelva un valor en cm, por supuesto, y en cierto sentido lo hace, pero es uno de varios modos (consulte la [**página de modos**](https://sites.google.com/site/ev3python/learn_ev3_python/using-sensors/sensor-modes) del [**sensor**](https://sites.google.com/site/ev3python/learn_ev3_python/using-sensors/sensor-modes) ) eso también **incluye un primer lugar decimal (pero sin punto decimal)**en el valor devuelto Eso significa que el valor devuelto realmente representa mm en lugar de cm. Por ejemplo, si el sensor devuelve el valor de 90 cm, en realidad significa 9.0 cm (o 90 mm) ya que el valor incluye un lugar decimal. Obviamente, debe tener mucho cuidado al manejar los valores devueltos por estos modos.  Para lidiar con estos problemas, he incluido una división por diez en el siguiente código para  convertir la distancia de mm a cm. Por lo tanto, el siguiente código hará  que  el color del LED izquierdo  cambie a rojo cuando un objeto se encuentre  dentro de los 60.0 cm (600 mm) del sensor de EE. UU.  De forma similar , si configura el modo en  US-DIST-IN   , espera que el valor esté en pulgadas, pero  de hecho  está en *décimas*de pulgada, por lo que debe dividir el valor entre 10 para obtener pulgadas. O, dicho de otra manera, el valor *está* en pulgadas pero incluye el primer lugar decimal del valor. Confuso, ¿verdad? Si usa pulgadas, probablemente desee utilizar una distancia inferior a 60.0 pulgadas para que el LED cambie a rojo. Tenga en cuenta que el problema con el sensor ultrasónico EV3 NO es compartido por el sensor ultrasónico NXT, que simplemente da la lectura como un número entero (cero lugares decimales).  #! / usr / bin / env python3  # para que el script pueda ejecutarse desde Brickman  de la importación ev3dev.ev3 \*  # Conecte sensores ultrasónicos y táctiles a cualquier puerto del sensor  # y verificar que estén conectados.  **us = Sensor ultrasónico ()**  **assert.connected, "Conecta un único sensor de EE. UU. a cualquier puerto de sensor"**  ts = TouchSensor (); assert ts.connected, "Conecta un sensor táctil a cualquier puerto"  # puede tener 2 instrucciones en la misma línea si usa semi colon  **# Ponga el sensor de EE. UU. En modo de distancia.**  **us.mode = 'US-DIST-CM'**  **units = us.units**  **# informa 'cm' a pesar de que el sensor mide 'mm'**  while not ts.value (): # Parar el programa presionando el botón táctil      El sensor # US medirá la distancia al más cercano      # objeto en frente de él.  **distance = us.value () / 10 # convert mm a cm**      imprimir (str (distancia) + "" + unidades)      si la distancia <60: # Esta es una distancia inconvenientemente grande          Leds.set\_color (Leds.LEFT, Leds.RED)      más:          Leds.set\_color (Leds.LEFT, Leds.GREEN)  Sound.beep ()  Leds.set\_color (Leds.LEFT, Leds.GREEN) #set left led green antes de salir  **Ejemplo 3 (dos sensores táctiles)**  Por lo general, no tendrá dos sensores del mismo tipo enchufados en el EV3 y no tendrá que especificar ni determinar dónde están conectados los sensores. Pero en este ejemplo supongamos que tenemos dos sensores táctiles conectados al EV3. ESPECIFICAREMOS en qué puertos se enchufan los sensores usando ' in1' y ' in2' para los puertos 1 y 2 del sensor (entrada). En este programa, el led izquierdo se pone verde si el sensor táctil en el puerto 1 no está presionado y rojo si se presiona (más específicamente, si el sensor táctil no se presiona, entonces devuelve un valor de cero y el código utiliza el elemento cero de la tupla). De manera similar, el sensor táctil en el puerto 2 controla el led derecho. Para detener este programa tendrá que pulsar Ctrl-C para interrumpir el tiempo lazo.  #! / usr / bin / env python3  # para que el script pueda ejecutarse desde Brickman  de la importación ev3dev.ev3 \*  desde el tiempo de importación de sueño  # Conecte DOS sensores táctiles a AMBOS puertos de sensor 1 y 2  # y verificar que ambos estén conectados.  ts **1** = TouchSensor ('en **1** ')  afirme ts1.connected, "Conecte un sensor táctil al puerto del sensor 1"  ts **2** = TouchSensor ('en **2** ')  afirme ts2.connected, "Conecte un sensor táctil al puerto del sensor 2"  while True: # Detener este programa con Ctrl-C      Leds.set\_color (Leds. **LEFT** , (Leds.GREEN, Leds.RED) [ts **1** .value ()])      Leds.set\_color (Leds. **DERECHA** , (Leds.GREEN, Leds.RED) [ts **2** .value ()])  # Parar el programa con Ctrl-C  **Ejemplo 4 (sensor giroscópico y sensor táctil)**  El sensor giroscópico se incluye con la versión educativa del kit EV3 pero no con la versión hogareña (se puede comprar por separado). Mide el ángulo en grados, dando el valor de cero grados a la orientación del sensor cuando se inicia el programa. Tenga en cuenta que es vital que el sensor del giróscopo esté completamente quieto cuando se inicia el programa, de lo contrario la lectura vagará más tarde, incluso cuando el sensor esté detenido.  El sensor también puede medir la *velocidad de cambio* del ángulo, en grados por segundo, pero como principiante es probable que encuentre menos útil.  El siguiente programa imprime el ángulo medido a la consola, luego reproduce por un segundo un tono cuya frecuencia depende del ángulo, luego espera 0,5 segundos antes de repetir el ciclo. Presione el botón del sensor táctil por al menos un segundo para detener el programa.  #! / usr / bin / env python3  # para que el script pueda ejecutarse desde Brickman  de la importación ev3dev.ev3 \*  desde el tiempo de importación de sueño  # Conecte el giroscopio y toque los sensores a cualquier puerto del sensor  # y verificar que estén conectados.  gy = GyroSensor ()  afirme gy.connected, "Conecte un solo sensor giroscópico a cualquier puerto de sensor"  ts = TouchSensor (); assert ts.connected, "Conecta un sensor táctil a cualquier puerto"  # puede tener 2 instrucciones en la misma línea si usa semi colon  # Ponga el sensor giroscópico en modo ANGLE.  gy.mode = 'GYRO-ANG'  units = gy.units  #reports 'deg' que significa grados  while not ts.value (): # Parar el programa presionando el botón táctil      angle = gy.value ()      imprimir (str (ángulo) + "" unidades +)      Sound.tone (1000 + ángulo \* 10, 1000) .wait ()      dormir (0.5)  Sound.beep ()  **Ejemplo 5: sensor de color  en modo COL-REFLECT**  En este script, el sensor de color emite luz y mide la intensidad de la luz reflejada, y devuelve dos veces por segundo un valor teóricamente entre 0 y 100. Para obtener los mejores resultados, coloque el sensor a unos 3 mm de la superficie reflectante. Usando esa separación, obtengo un valor de aproximadamente 80 con papel blanco normal y aproximadamente 5 con una superficie negra típica.  #!/usr/bin/env python3  from ev3dev.ev3 import \*  from time import sleep  # Connect EV3 color sensor and check connected.  cl = ColorSensor()  assert cl.connected, "Connect a color sensor to any sensor port"  # Put the color sensor into COL-REFLECT mode  # to measure reflected light intensity.  # In this mode the sensor will return a value between 0 and 100  cl.mode='COL-REFLECT'  while True:      print(cl.value())      sleep(0.5)    # I get max 80 with white paper, 3mm separation  # and 5 with black plastic, same separation  Ver también la lección 6 en [**esta página**](https://sites.google.com/site/ev3python/learn_ev3_python/basics)**.**  **Ejemplo 6: sensor de color EV3 en modo COL-AMBIENT, sensor táctil y motor grande**  En este programa, el sensor de color mide la intensidad de la luz ambiental, devolviendo un valor entre 0 y 100. Este valor se usa para controlar la velocidad de un motor grande conectado al puerto B, de modo que cuanto más brillante sea la luz ambiental más rápido girará el motor. Presione el botón del sensor táctil para detener el programa. Para que este programa funcione con la intensidad de la luz *reflejada en* lugar de *la* intensidad de *la* luz *ambiente* , simplemente reemplace  COL-AMBIENT  con  COL-REFLECT .  #!/usr/bin/env python3  # so that script can be run from Brickman  from ev3dev.ev3 import \*  from time import sleep  # Connect EV3 color sensor and touch sensor  # and check they are connected.  cl = ColorSensor()  assert cl.connected, "Connect an EV3 color sensor to any sensor port"  ts = TouchSensor()  assert ts.connected, "Connect a touch sensor to any sensor port"  # Connect a large motor to port B and check it is connected.  m = LargeMotor('outB')  assert m.connected, "Connect a large motor to port B"  # Put the color sensor into COL-AMBIENT mode  # to measure ambient light intensity.  # In this mode the sensor will return a value between 0 and 100  cl.mode='COL-AMBIENT'  # run\_forever command will allow us to vary motor  # performance on the fly by adjusting speed\_sp attribute.  m.run\_forever(speed\_sp = 0)  while not ts.value():    # Stop program by pressing touch sensor button  # set the motor's speed set point to be equal to  # the measured ambient light intensity value      m.speed\_sp = cl.value()  Sound.beep()  **Ejemplo 7: sensor de color EV3 en modo COL-COLOR y sensor táctil**  Cuando el sensor de color EV3 está en modo COL-COLOR, intenta reconocer el color de los ladrillos Lego estándar colocados a unos 5-6 mm delante del sensor (la distancia es crítica) y devuelve un valor entero correspondiente entre 0 (desconocido) y 7 (marrón). El siguiente programa lee el número entero una vez por segundo, lo convierte en la cadena de texto correspondiente usando una tupla y muestra la cadena en la consola. Presione el botón del sensor táctil por al menos un segundo para detener el programa.  Para que el programa diga los colores y muestre sus cadenas de texto, elimine el comentario de la línea resaltada en azul. Esto aumentará ligeramente el tiempo entre mediciones ya que incluí una función de espera ()  para asegurar que la conversación no fue interrumpida.  #! / usr / bin / env python3  # para que el script pueda ejecutarse desde Brickman  de la importación ev3dev.ev3 \*  desde el tiempo de importación de sueño  # Conecte el sensor de color EV3 a cualquier puerto del sensor  # y verificar que esté conectado.  cl = ColorSensor ()  assert cl.connected, "Conecta un solo sensor de color EV3 a cualquier puerto de sensor"  # Conecte el sensor táctil a cualquier puerto del sensor  # y verificar que esté conectado.  ts = TouchSensor (); assert ts.connected, "Conecta un sensor táctil a cualquier puerto"  # usted puede tener 2 declaraciones en la misma línea si usa un semi colon  # Ponga el sensor de color en modo COL-COLOR.  cl.mode = 'COL-COLOR'  colores = ('desconocido', 'negro', 'azul', 'verde', 'amarillo', 'rojo', 'blanco', 'marrón')  while not ts.value (): # Parar el programa presionando el botón táctil      imprimir (colores [cl.value ()])      # Sound.speak (colores [cl.value ()]). Wait ()      dormir (1)  Sound.beep ()  **Ejemplo 8: sensor de color EV3 en modo RGB-RAW y sensor táctil**  Este modo es especial ya que devuelve 3 valores simultáneamente, representando las cantidades de luz reflejada roja, verde y azul (se puede obtener cualquier color mezclando las cantidades correctas de luz roja, verde y azul). Se puede hacer referencia a estos valores en el código como valor (0), valor (1) y valor (2), respectivamente. El siguiente programa imprime los tres valores detectados una vez por segundo. Presione el botón del sensor táctil  por al menos un segundo  para detener el programa. Para más información, vea [**ESTA PÁGINA**](https://sites.google.com/site/ev3python/learn_ev3_python/using-sensors/sensor-modes/color-sensor-mode-4-rgb) .  #! / usr / bin / env python3  # para que el script pueda ejecutarse desde Brickman  de la importación ev3dev.ev3 \*  desde el tiempo de importación de sueño  # Conecte el sensor de color EV3 a cualquier puerto del sensor  # y verificar que esté conectado.  cl = ColorSensor ()  assert cl.connected, "Conecta un solo sensor de color EV3 a cualquier puerto de sensor"  # Conecte el sensor táctil a cualquier puerto del sensor  # y verificar que esté conectado.  ts = TouchSensor (); assert ts.connected, "Conecta un sensor táctil a cualquier puerto"  # usted puede tener 2 declaraciones en la misma línea si usa un semi colon  # Ponga el sensor de color en modo RGB.  cl.mode = 'RGB-RAW'  while not ts.value (): # Parar el programa presionando el botón táctil      rojo = cl.valor (0)      verde = cl.valor (1)      blue = cl.value (2)      print ("Rojo:" + str (rojo) + ", verde:" + str (verde) + ", azul:" + str (azul))      dormir (1)  Sound.beep () |