Introducción de Python para

LEGO® Education

SPIKE™

Pere Manel Verdugo Zamora

Este tutorial es un resumen obtenido de la aplicación Lego SPIKE del Banco del conocimiento

Introducción a Python

Banco de conocimiento

Para empezar

Desde aquí puedes comenzar a programar con Python. En las siguientes partes de esta sección "Para empezar" encontrarás numerosos ejemplos de programas que puedes utilizar mientras descubres las funciones de Python en SPIKE.

Cada vez que veas este icono  , púlsalo para copiar el contenido del recuadro. A continuación deberás pegarlo en la zona de programación. A esta zona la denominamos "panel de programación".

¡Vamos allá!

¿Cómo escribir un programa en Python?

Python es un lenguaje de programación muy intuitivo basado en texto. Es ideal para los principiantes porque es conciso y fácil de leer. También es perfecto para los programadores, porque puede utilizarse para cualquier cosa, desde el desarrollo web el desarrollo de software o para aplicaciones científicas.

Existen solo unas pocas reglas que debes recordar al escribir comandos.

- Cómo importar bibliotecas
- Cómo comentar sobre el código
- Convenciones de nomenclatura

¿Cómo importar bibliotecas?

Al crear proyectos de Python, a menudo deberás importar una biblioteca de funciones. En jerga de programación, una biblioteca consiste esencialmente en todos aquellos posibles "ingredientes" que puedes utilizar para crear tu "receta".

Al utilizar SPIKE, debes "incluir" siempre la biblioteca asociada a los diversos componentes de hardware (por ejemplo motores, Hub, sensores)

```
from spike import PrimeHub,
MotorPair
```

Estas son algunas de las bibliotecas que puedes importar.

- App
- Hub
- Matriz de luces
- Botones
- Altavoz
- Luz de estado del ladrillo
- Sensor de movimiento
- Sensor de distancia
- Sensor de color
- Sensor de fuerza
- Motor
- Pareja de motores
- Operadores

Las bibliotecas importadas se encuentran al principio del archivo .py y deben aparecer solo una vez en el programa.

Si no estás seguro de que biblioteca debes importar, siempre puedes importar todo lo posible mediante:

```
from spike import PrimeHub,
LightMatrix, Button,
StatusLight, ForceSensor,
MotionSensor, Speaker,
ColorSensor, App,
DistanceSensor, Motor, MotorPair
from spike.control import
wait_for_seconds, wait_until,
Timer
```

Comentarios en Python

Cada línea que comience por un símbolo "#" se considera un comentario. Por lo tanto, no se ejecuta como acción.

```
# Esto es un comentario.
# Esto es otro comentario.
```

Sangría

Python distingue las mayúsculas, los espacios y la sangría. Esto hace que sea más sencillo programar una vez que te hayas familiarizado con las reglas. Por ejemplo, hay una distinción entre:

```
x = 0
if x == 1:
print('LEGO')
```

y

```
x = 0
if x == 1:
  print('LEGO')
```

¿Cómo usar este banco de conocimientos?

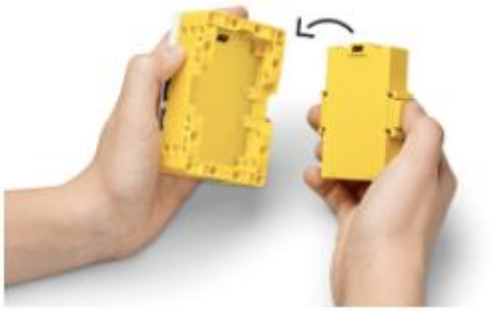
Este banco de conocimientos está organizado en las siguientes secciones:

1. Una guía "Para empezar" **paso a paso** que te orientará a medida que descubres las funciones más habituales utilizadas para controlar el Hub, los motores y los sensores SPIKE. Además, en esta sección, el **Traductor de bloques de palabras** te mostrará cómo convertir bloques de palabra en código Python para algunas de las funciones.
2. La lección **Pasa el ladrillo** con la que puedes practicar el trabajo en equipo.
3. La **Referencia de Python para SPIKE**, donde encontrarás descripciones de todas las funciones (por ejemplo, programas de muestra) disponibles para que pruebes con ellos.

Para descubrir y experimentar con estos programas de muestra, cópialos a tu proyecto y a continuación modifícalos y personalízalos.

Parte 1: Cómo programar salidas sencillas

Vas a necesitar…



Este es el Hub. Vas a crear unos cuentos programas breves para él.

- Asegúrate de haber instalado la batería.
- De haberla encendido pulsando el botón central
- Y de haberlo conectado a tu dispositivo mediante Bluetooth o USB.

¿Cómo controlar la matriz de luces?

Crea tu prime programa con Python.

1. Copia el siguiente código presionando el icono de copiar en el recuadro.
2. Pega el código en el panel de programación. Este es el espacio donde escribirás tu código Python. Las líneas vedes son comentarios y no influirá en las acciones. Las otras líneas son tu programa. ¿Puedes deducir en qué consiste el primer programa?
3. Entonces, ¡ejecútalo!

```
# Importa la clase PrimeHub
from spike import PrimeHub
from spike.control import wait_for_seconds
# Inicializa el Hub
your_hub = PrimeHub()
# Ilumina una cara sonriente
your_hub.light_matrix.show_image('HAPPY')
wait_for_seconds(5)
your_hub.light_matrix.off()
```

Deberías verlo en acción en el Hub.

4. Cambia la imagen que muestra en la matriz de luces.
    a. Puedes cambiar el parámetro del “HAPPY” a “HEART” en el código que ya tienes. Esto hará que se ilumine un corazón en lugar de una cara sonriente en el Hub.
    b. O, si lo prefieres, puedes copiar el código del siguiente recuadro y pegarlo tras la última línea de tu programa. Esto hará que se iluminen con una cara sonriente durante 5 segundos y que después lo haga con un corazón durante otros 5 segundos.

```
# Importa la clase PrimeHub
from spike import PrimeHub
from spike.control import wait_for_seconds
# Inicializa el Hub
your_hub = PrimeHub()
# Ilumina una cara sonriente
your_hub.light_matrix.show_image('HAPPY')
wait_for_seconds(5)
your_hub.light_matrix.off()
# Añade otra imagen
your_hub.light_matrix.show_image('HEART')
wait_for_seconds(5)
your_hub.light_matrix.off()
```

¡Enhorabuena! Acabas de escribir tu primer programa en Python. Continua para seguir aprendiendo.

Reproducción de pequeños pitidos y tiempo

Vamos a hacer que el Hub reproduzca algunos pitidos.

1. Si tienes un programa ahora mismo en tu panel de programación, te recomendamos que lo borres o que comiences un nuevo proyecto para continuar.
2. A continuación, copia el siguiente código en el panel de programación.
3. Asegúrate de tener solamente una línea de código que comience con from spike import.
4. ¡Ejecuta el programa!

```
# Importa la clase PrimeHub
from spike import PrimeHub
# Inicializa el Hub
hub = PrimeHub()
# ¡Bip bip bip!
hub.speaker.beep(60, 1)
```

5. Cambia el ritmo y el tono. Aquí tienes una manera de hacerlo.

```
# Aquí tienes una nueva
canción
hub.speaker.beep(60, 0.5)
hub.speaker.beep(67, 1.0)
wait_for_seconds(0.5)
hub.speaker.beep(60, 0.5)
```

¿Cómo reproducir sonidos?

También puedes añadir algunos sonidos para reproducir desde tu dispositivo.

1. Copia el siguiente código en el panel de programación. ¡Ejecuta el programa!

```
# Importa la clase PrimeHub
from spike import App
# Inicializa la app
app = App()
app.play_sound('Cat Meow 1')
```

2. Escoge otro sonido que reproducir o utiliza este programa.

```
app.play_sound('Triumph')
```

Desafío

Utiliza las funciones que has aprendido hasta ahora.

Crea un breve programa de cuanta atrás algo como: 3, 2, 1, ¡BUUUM!

Parte 2: ¿Cómo controlar motores?

Vas a necesitar…

Ahora, vamos a conectar un motor y encenderlo. Conecta un motor a uno de los puertos con letra (por ejemplo, puerto C).

Accionar un único motor durante un tiempo determinado

Vamos a hacer funcionar el motor durante 2 segundos.

1. Copia y pega este código en el panel de programación.
2. Ejecuta el programa y observa el motor.

```
from spike import Motor
# Inicializa el motor
motor = Motor('C')
# Gira en sentido horario
durante 2 segundos al 75 % de
velocidad
motor.run_for_seconds(2.0, 75)
```

3. Modifica tu código para cambiar la velocidad del motor y la duración de su movimiento Por ejemplo:

```
# Gira en sentido horario
durante 6,5 segundos al 30 % de
velocidad
motor.run_for_seconds(6.5, -30)
```

Accionar un único motor unos grados determinados

Vamos a hacer que el motor se desplace 360 grados.

1. Copia el siguiente código en el panel de programación.
2. Ejecuta el programa y observa el motor.

```
from spike import Motor
# Inicializa el motor
motor = Motor('C')
# Gira el motor 360 grados en
sentido horario:
motor.run_for_degrees(360)
```

3. Modifica tu código para cambiar la dirección del motor mediante grados de rotación. Por ejemplo:

```
# Acciona el motor 360 grados
en sentido horario al 30 % de
velocidad
motor.run_for_degrees(-360, 30)
```

Accionar un único motor hasta una posición

Vamos a llevar el motor a la posición de 0 grados desde cualquier posición de la que te encuentres actualmente. Su posición viene indicada por el marcador del motor.

1. Copia y pega este código en el panel de programación.
2. Ejecuta el programa y observa el motor.

```
from spike import Motor
# Inicializa el motor
motor = Motor('C')
# Establece el motor en la
posición "0" alineando las
señales
motor.run_to_position(0,
'shortest path', 75)
```

3. Cambia tu código para hacer que el motor se detenga en diferentes posiciones.

```
# Acciona el motor hasta
diferentes posiciones a
diferentes velocidades
wait_for_seconds(1)
motor.run_to_position(0,
'shortest path', 30)
wait_for_seconds(1)
motor.run_to_position(90,
'clockwise', 100)
```

Desafío

Crea un programa corto para accionar dos motores según un ritmo: algo como ambos motores en un solo sentido, ambos en el sentido opuesto, un motor en sentido opuesto al otro ¡y mucho más!

Utiliza las funciones que has aprendido hasta ahora.

Parte 3: Cómo usar un sensor de fuerza

Vas a necesitar…

Conecta un Sensor de fuerza en el puerto B y un motor al puerto C.

Empujar, arrancar y detener

Vamos a utilizar el Sensor de fuerza para controlar el motor.

1. Copia y pega este código en el panel de programación.
2. Ejecuta el programa y pulsa el botón del Sensor de fuerza.

```
from spike import ForceSensor,
Motor
# Inicializa el Sensor de fuerza
y un motor
force_sensor = ForceSensor('B')
motor = Motor('C')
# Pulsa el botón lentamente:
solo funcionará una sola vez #
Ejecuta de nuevo el programa
para intentarlo de nuevo
motor.set_default_speed(25)
force_sensor.wait_until_pressed()
motor.start()
force_sensor.wait_until_released()
motor.stop()
```

3. Cambia tu código para descubrir otra interacción con el sensor.

```
motor.set_default_speed(25)
force_sensor.wait_until_pressed()
force_sensor.wait_until_released()
motor.start()
force_sensor.wait_until_pressed()
force_sensor.wait_until_released()
motor.stop()
```

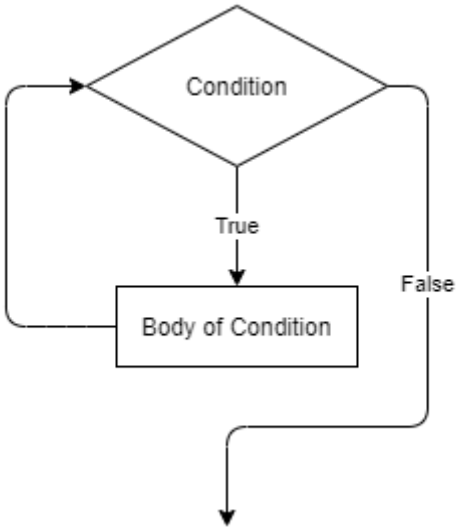
Parte 4: ¿Cómo cambiar el flujo mediante bucles y condiciones?

Vas a necesitar…

Con un Sensor de fuerza conectado al puerto B y un motor en el puerto C, vamos a descubrir maneras de cambiar el flujo de ejecución del programa.

Bucle Mientras

El bucle Mientras es una estructura utilizada para repetir algo. Se usa con una condición. El bucle se repetirá hasta que la condición sea falsa.



Para indicar qué se incluye en el cuerpo del bucle, debes aplicar sangría al texto. A continuación te mostramos un ejemplo.

1. Copia y pega este código en el panel de programación.
2. Ejecuta el programa y pulsa el botón del Sensor de fuerza.

```
from spike import ForceSensor, Motor
# Inicializa el Sensor de fuerza, un motor y una variable
force_sensor = ForceSensor('B')
motor = Motor('C')
count = 0
# Puedes pulsar cinco veces el Sensor de fuerza
motor.set_default_speed(25)
while count < 5:
    force_sensor.wait_until_pressed()
    motor.start()
    force_sensor.wait_until_released()
    motor.stop()
    count = count + 1
```

3. Cambia tu código para descubrir otra interacción en el sensor.

```
from spike import ForceSensor, Motor
# Inicializa el Sensor de fuerza, un motor y una variable
force_sensor = ForceSensor('B')
motor = Motor('C')
count = 0
# Puedes pulsar cinco veces el Sensor de fuerza
motor.set_default_speed(25)
# Esta condición siempre será verdadera, por lo que estará para siempre en bucle
while True:
    # Mide la fuerza en newtons o como porcentaje
    percentage = force_sensor.get_force_percentage()
    # Utiliza la fuerza medida para arrancar el motor
    motor.start(percentage)
```

Parte 5: Usar sensor de color

Vas a necesitar…

Conecta un Sensor de color al puerto F y dos motores a los puertos A y E.

¿Amarillo o violeta?

Vamos a utilizar el Sensor de color para controlar los motores.

1. Copia y pega este código en el panel de programación.
2. Ejecuta el programa. Preséntale al Sensor de color y observa los motores.

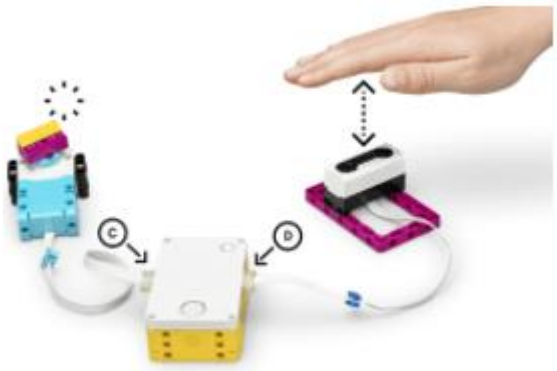
```
from spike import ColorSensor,
Motor
from spike.control import Timer
# Inicializa el Sensor de color,
dos motores y un cronómetro
color_sensor = ColorSensor('F')
motor_a = Motor('A')
motor_e = Motor('E')
timer = Timer()
# Presenta cada ladrillo de
color al Sensor de color y
observa lo que ocurre: detectará
colores durante 30 segundos
while timer.now() < 30:
  color =
color_sensor.wait_for_new_color()
  if color == 'violet':
    motor_a.run_for_rotations(1)
  elif color == 'yellow':
    motor_e.run_for_rotations(1)
```

3. Cambia tu código para descubrir otra interacción con el sensor.

```
# Esto empleará el valor
reflejado de los colores para
establecer la velocidad del
motor (amarillo es
aproximadamente un 80 % y
violeta un 60 %)
while True:
  color =
color_sensor.wait_for_new_color()
  percentage =
color_sensor.get_reflected_light()
  if color == 'magenta':
    motor_a.run_for_rotations(1,
percentage)
  elif color == 'yellow':
    motor_e.run_for_rotations(1,
percentage)
```

Parte 6: ¿Cómo utilizar el Sensor de Distancia?

Vas a necesitar…



Conecta un Sensor de distancia al puerto D y un motor al puerto C.

Más cerca o más lejos

Vamos a utilizar el Sensor de distancia para controlar el motor.

1. Copia y pega este código en el panel de programación.
2. Ejecuta el programa y mueve la mano sobre el Sensor de distancia.

```
from spike import DistanceSensor, Motor
# Inicializa el Sensor de distancia y el motor
distance_sensor = DistanceSensor('D')
motor = Motor('C')
# Mueve tu mano lentamente aproximándola a y alejándola del Sensor de distancia
while True:
    distance_sensor.wait_for_distance_farther_than(20, 'cm')
    motor.start()
    distance_sensor.wait_for_distance_closer_than(20, 'cm')
    motor.stop()
```

Más allá de

Más cerca que

3. Cambia tu código para descubrir otras interacciones con el sensor.

```
from spike import DistanceSensor, Motor
# Inicializa el Sensor de distancia y el motor
distance_sensor = DistanceSensor('D')
motor = Motor('C')
# Mueve tu mano lentamente aproximándola a y alejándola del Sensor de distancia;
# la velocidad del motor cambiará en función de la distancia detectada
while True:
    percentage = distance_sensor.get_distance_percentage()
    if percentage is not None:
        motor.start(100 - percentage)
```

Parte 7: ¿Cómo utilizar el sensor de movimiento?

Vas a necesitar…

Solo necesitarás el Hub, que vas a coger y a inclinar.

Detectar la posición

Vamos a controlar la matriz de luces utilizando el Sensor de movimiento del Hub.

1. Copia y pega este código en el panel de programación.
2. Ejecuta el programa e inclina el Hub a izquierda y derecha.

```
from spike import PrimeHub, App
# Inicializa el Hub y la app
hub = PrimeHub()
app = App()

while True:
    orientation = hub.motion_sensor.wait_for_new_orientation()
    if orientation == 'front':
        hub.light_matrix.show_image('ASLEEP')
        app.start_sound('Snoring')
    elif orientation == 'up':
        hub.light_matrix.show_image('HAPPY')
        app.start_sound('Triumph')
```

Parte delantera

Arriba

3. Cambia tu código para descubrir otra interacción con el sensor.

```
from spike import PrimeHub, App
# Inicializa el Hub y la app
hub = PrimeHub()
app = App()
while True:
    angle = abs(hub.motion_sensor.get_pitch_angle()) * 2
    hub.light_matrix.show_image('HAPPY', angle)
```

Desafío

Utiliza sensores para crear un instrumento musical que pueda modular diferentes sonidos.

Parte 8: Conduciendo

Vas a necesitar…

Necesitas una Base de conducción. Puedes construir el modelo como tu prefieras, pero la opción más sencilla es unir dos motores al Hub orientados en sentidos opuestos. ¡Conecta algunas ruedas y listo! Visita la página de instrucciones de construcción para inspirarte.

Moverse hacia adelante o hacia atrás

Vamos a programar tu Base de conducción para que se mueva en línea recta.

1. Copia y pega ese código en el panel de programación.
2. ¡Ejecuta el programa! Asegúrate de que tienes suficiente espacio para que tu Base de conducción se pueda mover.

```
from spike import MotorPair
# Inicializa la pareja de
motores
motor_pair = MotorPair('E', 'F')
# Inicializa la velocidad
predeterminada
motor_pair.set_default_speed(50)
# Muévete en un sentido durante
2 segundos
motor_pair.move(2, 'seconds')
```

3. Utiliza este código para cambiar el movimiento de tu base de conducción.

```
# Muévete en el sentido
opuesto durante 2 segundos
motor_pair.set_default_speed(-50)
motor_pair.move(2, 'seconds')
```

Hacer girar una Base de conducción (giro sobre sí mismo)

¿Vas en línea recta pero necesitas girar? Vamos a probar lo que llamamos un “giro sobre sí mismo”. Girará la Base de conducción sobre un mismo punto.

1. Copia y pega este código en el panel de programación.
2. ¡Ejecuta el programa! Asegúrate de que tienes suficiente espacio para que tu Base de conducción se pueda mover.

```
from spike import MotorPair
# Inicializa la pareja de
motores
motor_pair = MotorPair('E', 'F')
# Gira en un sentido durante 2
segundos
motor_pair.move_tank(10, 'cm',
left_speed=25, right_speed=75)
```

3. Utiliza este código para cambiar el movimiento de tu Base de conducción.

```
# Muévete en el sentido
opuesto durante 2 segundos
motor_pair.move_tank(1,
'rotations', left_speed=-50,
right_speed=50)
```

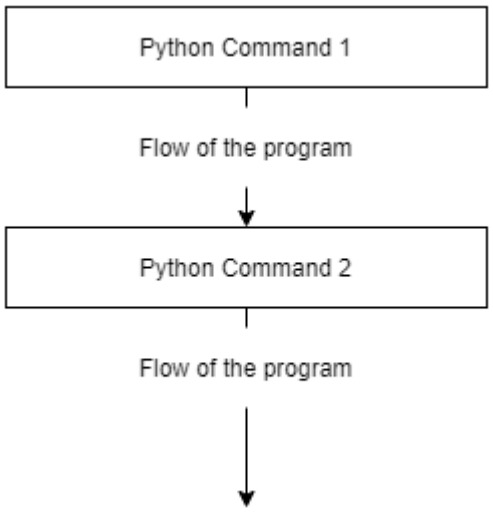
Desafío

¡Este es un clásico! Programa tu Base de conducción para que se desplace formando un cuadrado mediante las funciones que acabas de probar.

Más nociones básicas de Python

Al ser un lenguaje de programación basado en texto, Python se basa en ciertos principios que merecen tratarse en más detalle.

- Flujo de ejecución
- Tipos de datos
- Clases, métodos y objetos

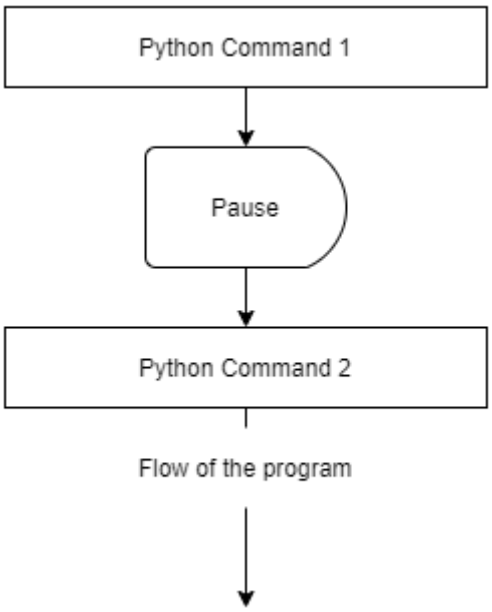


Flujo de ejecución

Recuerda que el código en Python se ejecuta línea a línea. Esto se denomina “Flujo de la ejecución”.

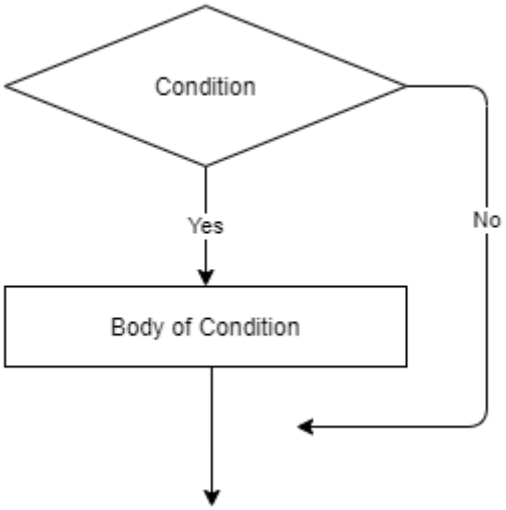
Hay muchas maneras de modificar este flujo. Por ejemplo, siempre puedes pausar la ejecución del programa mediante el comando:

wait_for_seconds(1) (el número especificado en el campo() se aporta en segundos). Tendrá el siguiente efecto en el flujo de ejecución.



If/Else

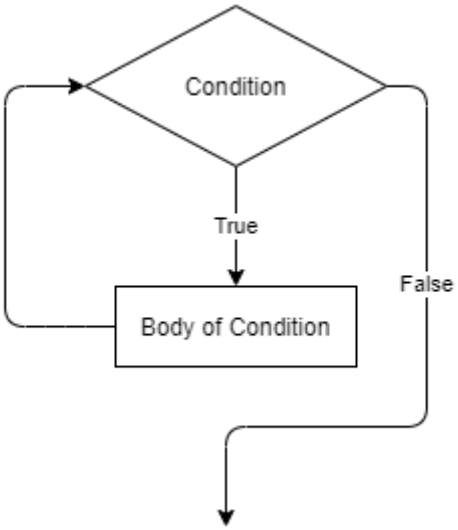
Las estructuras “if/else” te permiten modificar el flujo de ejecución del programa El uso de la estructura “if/else” depende de la habilidad de crear y utilizar instrucciones condicionales apropiadas. El programa comprobará si la condición en verdadera. En caso afirmativo, ejecutará los comandos contenidos en la estructura “if”. Si no, ejecutará los comandos contenidos en la estructura “else”.



Para indicar qué se incluye en el cuerpo de la condición, debes aplicar sangría al texto.

Bucle Mientras

El bucle Mientras es una estructura utilizada para repetir algo. Se usa con una condición. El bucle se repetirá hasta que la condición sea falsa.



En Python, a menudo utilizamos la estructura Mientras verdadero: lo que quiere decir que repetimos acciones indefinidamente porque la condición es siempre verdadera.

Tipo de datos

Al usar un lenguaje de programación basado en texto, tendrás que experimentar con diferentes tipos de valores. Al principio, usarás principalmente números, cadenas y listas.

- Números (enteros): números enteros positivos o negativos incluyendo el 0.

```
my_integer = 7
print(my_integer)
```

- Números (flotantes): Un número con decimales.

```
my_float = 7.0
print(my_float)
```

- Cadena (texto): Cualquier carácter.

```
my_string = 'Hello World'
print(my_string)
```

- Listas: múltiples valores combinados, siendo cada valor accesible mediante un índice. El índice comienza con una valor de “0”.

```
mylist = [1,2,3]
print(mylist[1])
```

Clase, objetos, métodos y parámetros

Python es un lenguaje de programación orientado al objeto.

Así es como funciona:

- En la biblioteca SPIKE, hemos definido los componentes electrónicos que puedes programar. Los hemos agrupado en clases Un ejemplo de la clase Hub, que define todo lo que puedes hacer con el Hub. (from spike import PrimeHub).
- Para utilizar una clase, tienes que crear una copia de ella. El acto de crear una copia se denomina "crear un objeto". Esto se hace iniciando un componente: My_own_hub_object-Hub() → Mi propio objeto central.
- Ahora que has creado una copia (es decir, un objeto) de una clase, puedes hacer multitud de "cosas" con ella. Estas "cosas" se denominan métodos o, a veces funciones. Los métodos a veces incluyen parámetros y a veces no. Por ejemplo:

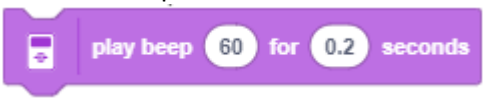
```
# El método show_image() cuenta con un parámetro denominado "happy"
my_own_hub_object.light_matrix.show_image('HAPPY')


# El método stop() carece de parámetro
motor.stop()
```

Traductor de bloque de palabras

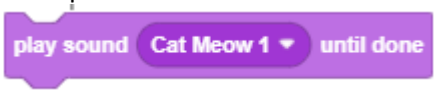
Estos son los bloques de palabras más habituales traducidos a Python para SPIKE.

1.- Sonido de pitido.



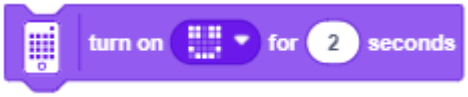

```
hub.speaker.beep(60, 0.2)
```

2.- Reproducir sonido.




```
app.play_sound('Cat Meow 1')
```

3.- Matriz de luces




```
hub.light_matrix.show_image('HAPPY')
wait_for_seconds(2)
```

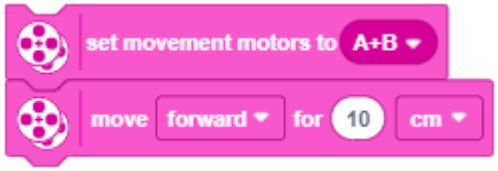
4.- Motor individual encendido durante ciertos segundos.


```
motor = Motor('A')
motor.run_for_seconds(1, 75)
```

5.-. Motor múltiple durante ciertos grados


```
motor_a = Motor('A')
motor_e = Motor('E')
motor_a.run_for_rotation(1, 75)
motor_e.run_for_rotation(1, 75)
```

6.- Desplazarse en línea recta (Base de conducción)




```
motor_pair = MotorPair('E',
'F')
motor_pair.move(10, 'cm')
```

7.- Esperar 2 segundos


```
wait_for_seconds(2)
```

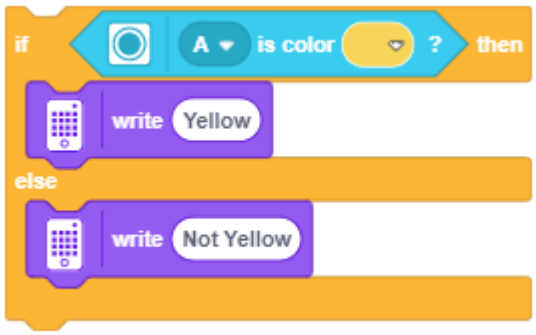
8.- Esperar el Sensor de fuerza


```
force_sensor.wait_until_pressed()
```

9.- Repetir 10 veces

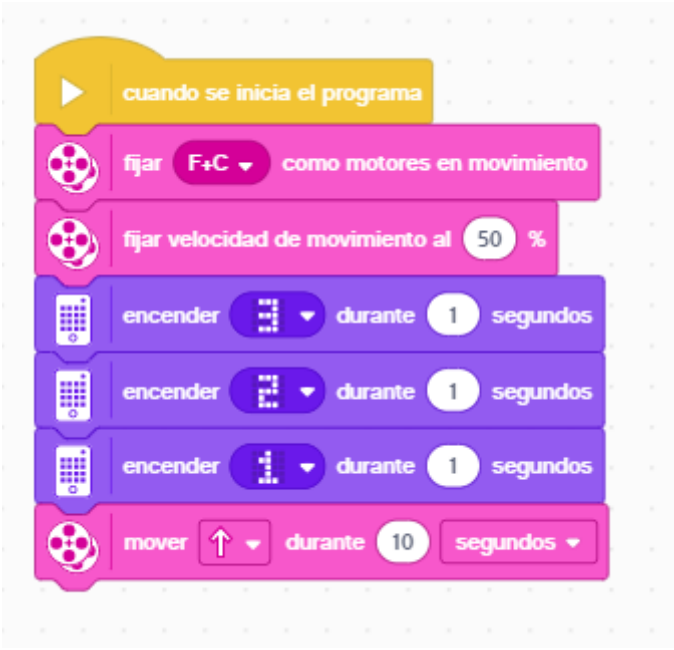

```
count = 0
while (count < 10):
  count = count + 1
  hub.light_matrix.write(count)
```

10.- Si… Si no…




```
color_sensor =
ColorSensor('A')
while True:
  color =
color_sensor.wait_for_new_color()
  if color == 'yellow':
    print('Yellow')
  else:
    print('Not Yellow')
```

11.- Un programa.



```python
from spike import PrimeHub, MotorPair
from spike.control import wait_for_seconds

hub = PrimeHub()
motor_pair = MotorPair('F', 'C')
motor_pair.set_default_speed(50)

# 3
hub.light_matrix.off()
hub.light_matrix.set_pixel(1, 0)
hub.light_matrix.set_pixel(2, 0)
hub.light_matrix.set_pixel(3, 0)
hub.light_matrix.set_pixel(3, 1)
hub.light_matrix.set_pixel(1, 2)
hub.light_matrix.set_pixel(2, 2)
hub.light_matrix.set_pixel(3, 2)
hub.light_matrix.set_pixel(3, 3)
hub.light_matrix.set_pixel(1, 4)
hub.light_matrix.set_pixel(2, 4)
hub.light_matrix.set_pixel(3, 4)

wait_for_seconds(1)

# 2
hub.light_matrix.off()
hub.light_matrix.set_pixel(1, 0)
hub.light_matrix.set_pixel(2, 0)
hub.light_matrix.set_pixel(3, 0)
hub.light_matrix.set_pixel(3, 1)
hub.light_matrix.set_pixel(1, 2)
hub.light_matrix.set_pixel(2, 2)
hub.light_matrix.set_pixel(3, 2)
hub.light_matrix.set_pixel(1, 3)
hub.light_matrix.set_pixel(1, 4)
hub.light_matrix.set_pixel(2, 4)
hub.light_matrix.set_pixel(3, 4)

wait_for_seconds(1)

# 1
hub.light_matrix.off()
hub.light_matrix.set_pixel(2, 0)
hub.light_matrix.set_pixel(1, 1)
hub.light_matrix.set_pixel(2, 1)
hub.light_matrix.set_pixel(2, 2)
hub.light_matrix.set_pixel(2, 3)
hub.light_matrix.set_pixel(1, 4)
hub.light_matrix.set_pixel(2, 4)
hub.light_matrix.set_pixel(3, 4)

wait_for_seconds(1)
motor_pair.move(10, 'seconds')

```

Lección 1: Pasa el ladrillo

Esta es una lección que puedes probar en el aula. Después de completar algunos de los pasos de la sección “Para empezar”, puedes probar tus destrezas de programación en Python en un contexto didáctico. Esta lección, que emplea bloques de palabras, puedes encontrarla también en la sección “Otros recursos” de esta app.

¡Que lo disfrutes!

Objetivo clase

En esta lección, los alumnos demostrarán su habilidad a la hora de trabajar de manera eficaz y respetuosa con diversos tipos de persona mientras utilizan Python para construir una mano robótica.

¿Estás en algún equipo?

Deporte, música, baile, juegos… hay equipos por todas partes. ¿Tiene tu equipo un líder? ¿Qué hace tu líder?

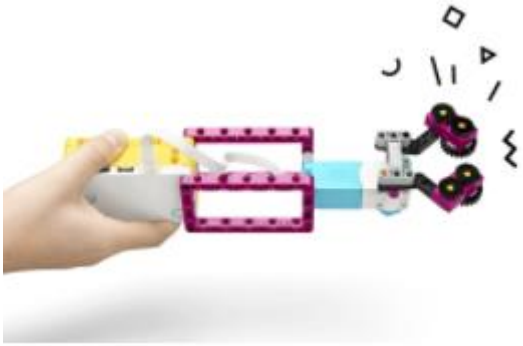
¡Tenéis 5 minutos para construir una mano robótica!

Desafío 1: Vais a elegir un líder de equipo y seguir sus indicaciones. Tenéis que trabajar lo más rápido que podáis para construir:

- Una mano robótica

También vais a necesitar siete ladrillos LEGO.

¡A probar la mano robótica!



Vais a ejecutar el programa presionando el botón izquierdo del Hub para comprobar que la mano robótica se cierra.

```python
from spike import Motor, PrimeHub
# Inicializa el Hub y el motor
hub = PrimeHub()
motor = Motor('F')

# Esto hará que la mano se abra una vez para comenzar
motor.run_for_seconds(1, 75)

while True:
    # Esto hará que la mano se cierre cuando pulses el botón izquierdo del Hub
    hub.left_button.wait_until_pressed()
    motor.set_stall_detection(False)
    motor.start(-75)


    # Esto hará que la mano se abra cuando pulses el botón izquierdo del Hub
    hub.left_button.wait_until_released()
    motor.set_stall_detection(True)


    #↑↑↑↑↑↑↑↑↑↑↑
    # Aquí falta una línea de código
```

¡Vosotros tenéis el control de la mano robótica!

Completa el programa para hacer que tu mano robótica se abra y se cierre. Le falta un comando que necesita para funcionar correctamente. (Pista: echa un vista a los comentarios del programa).

¡Toca mover 6 ladrillos en 2 minutos!

Desafío n.º 2: Vais a elegir otro líder y a seguir sus indicaciones.

Por turnos, usad la mano robótica para que lleves los ladrillos uno a uno de punto A al B.

Tres minutos para apilar todos los ladrillos posibles

Desafío n.º 3: Le toca a otro líder.

Vais a seguir sus indicaciones y, por turnos, usar la mano robótica para apilar los ladrillos.

¡Cuidado! Si se cae la torre, tienes que reconstruirla.

Mover los ladrillos de un lugar a otro

Desafío n.º 4: Ahora, tu equipo va a justarse con otro y elegir un líder.

Equipo A: Por turnos, usad la ano robótica para coger un ladrillo. Hay que encontrarse a medio camino con el otro equipo y pasarle el ladrillo.

Equipo B: Por turnos, vais a usar solo la mano robótica para agarrar el ladrillo que os pase el equipo contrario y dejarlo en el lugar que se haya señalado.

¿Qué tal te ha ido?

Piensa lo que has hecho bien y lo que podrías haber hecho mejor.

¿Has descubierto otras formas de mejorar tu manera de trabajar en equipo?

App

Para poder utilizar la aplicación, debes inicializarla.

Ejemplo:

```
from spike import App


# Inicializa la app.
app = App()
```

A continuación te mostramos todas las funciones vinculadas a los elementos programables de la app SPIKE.

play_sound()

play_sound(name, volume=100)

Reproduce un sonido desde el dispositivo (tableta u ordenador).

El programa no continuará hasta que no se haya terminado de reproducir el sonido.

Si no se encuentra ningún sonido con el nombre proporcionado, no sucederá nada.

Parámetros

El nombre del sonido que se quiere reproducir.

Tipo:	Cadena de texto
Valores:	Alert, Applause 1, Applause 2, Applause 3, Baa, Bang 1, Bang 2, Basketball Bounce, Big Boing, Bird, Bite, Boat Horn 1, Boat Horn 2, Bonk, Boom Cloud, Boop Bing Bop, Bowling Strike, Burp 1, Burp 2, Burp 3, Car Accelerate 1, Car Accelerating 2, Car Horn, Car Idle, Car Reverse, Car Skid 1, Car Skid 2, Car Vroom, Cat Angry, Cat Happy, Cat Hiss, Cat Meow 1, Cat Meow 2, Cat Meow 3, Cat Purring, Cat Whining, Chatter, Chirp, Clang, Clock Ticking, Clown Honk 1, Clown Honk 2, Clown Honk 3, Coin, Collect, Connect, Crank, Crazy Laugh, Croak, Crowd Gasp, Crunch, Cuckoo, Cymbal Crash, Disconnect, Dog Bark 1, Dog Bark 2, Dog Bark 3, Dog Whining 1, Dog Whining 2, Doorbell 1, Doorbell 2, Doorbell 3, Door Closing, Door Creek 1, Door Creek 2, Door Handle, Door Knock, Door Slam 1, Door Slam 2, Drum Roll, Dun Dun Dunnn, Emotional Piano, Fart 1, Fart 2, Fart 3, Footsteps, Gallop, Glass Breaking, Glug, Goal Cheer, Gong, Growl, Grunt, Hammer Hit, Head Shake, High Whoosh, Jump, Jungle Frogs, Laser 1, Laser 2, Laser 3, Laughing Baby 1, Laughing Baby 2, Laughing Boy, Laughing Crowd 1, Laughing Crowd 2, Laughing Girl, Laughing Male, Lose, Low Boing, Low Squeak, Low Whoosh, Magic Spell, Male Jump 1, Male Jump 2, Moo, Ocean Wave, Oops, Orchestra Tuning, Party Blower, Pew, Ping Pong Hit, Plane Flying By, Plane Motor Running, Plane Starting, Pluck, Police Siren 1, Police Siren 2, Police Siren 3, Punch, Rain, Ricochet, Rimshot, Ring Tone, Rip, Robot 1, Robot 2, Robot 3, Rocket Explosion Rumble, Rooster, Scrambling Feet, Screech, Seagulls, Service Announcement, Sewing Machine, Ship Bell, Siren Whistle, Skid, Slide Whistle 1, Slide Whistle 2, Sneaker Squeak, Snoring, Snort, Space Ambience, Space Flyby, Space Noise, Splash, Sport Whistle 1, Sport Whistle 2, Squeaky Toy, Squish Pop,

Suction Cup, Tada, Telephone Ring 2, Telephone Ring, Teleport 2, Teleport 3, Teleport, Tennis Hit, Thunder Storm, Toliet Flush, Toy Honk, Toy Zing, Traffic, Train Breaks, Train Horn 1, Train Horn 2, Train Horn 3, Train On Tracks, Train Signal 1, Train Signal 2, Train Start, Train Whistle, Triumph, Tropical Birds, Wand, Water Drop, Whistle Thump, Whiz 1, Whiz 2, Window Breaks, Win, Wobble, Wood Tap, Zip

Valor predeterminado:	Sin valor
Volumen:	El volume al que se reproducirá el sonido.
Tipo:	Entero (número entero positivo o negativo, incluido el 0.
Valores:	0 a 100%

Errores

TypeError	El nombre no es una cadena o el volumen no es un número entero.
RuntimeError	La aplicación SPIKE se ha desconectado del Hub

Ejemplo:

```
from spike import App

app = App()

app.play_sound('Cat Meow 1')
```

start_sound()

start_sound(name, volume=100)

Comienza a reproducir un sonido desde tu dispositivo (tableta u ordenador).

El programa no esperará hasta que el sonido termine de reproducirse antes de seguir el siguiente comando.

Si no se encuentra ningún sonido con el nombre proporcionado, no sucederá nada.

Parámetros

El nombre del sonido que se quiere reproducir.

Tipo:	Cadena de texto
Valores:	Alert, Applause 1, Applause 2, Applause 3, Baa, Bang 1, Bang 2, Basketball Bounce, Big Boing, Bird, Bite, Boat Horn 1, Boat Horn 2, Bonk, Boom Cloud, Boop Bing Bop, Bowling Strike, Burp 1, Burp 2, Burp 3, Car Accelerate 1, Car Accelerating 2, Car Horn, Car Idle, Car Reverse, Car Skid 1, Car Skid 2, Car Vroom, Cat Angry, Cat Happy, Cat Hiss, Cat Meow 1, Cat Meow 2, Cat Meow 3, Cat Purring, Cat Whining, Chatter, Chirp, Clang, Clock Ticking, Clown Honk 1, Clown Honk 2, Clown Honk 3, Coin, Collect, Connect, Crank, Crazy Laugh, Croak, Crowd Gasp, Crunch, Cuckoo, Cymbal Crash, Disconnect, Dog Bark 1, Dog Bark 2, Dog Bark 3, Dog Whining 1, Dog Whining 2, Doorbell 1, Doorbell 2, Doorbell 3, Door

Closing, Door Creek 1, Door Creek 2, Door Handle, Door Knock, Door Slam 1, Door Slam 2, Drum Roll, Dun Dun Dunnn, Emotional Piano, Fart 1, Fart 2, Fart 3, Footsteps, Gallop, Glass Breaking, Glug, Goal Cheer, Gong, Growl, Grunt, Hammer Hit, Head Shake, High Whoosh, Jump, Jungle Frogs, Laser 1, Laser 2, Laser 3, Laughing Baby 1, Laughing Baby 2, Laughing Boy, Laughing Crowd 1, Laughing Crowd 2, Laughing Girl, Laughing Male, Lose, Low Boing, Low Squeak, Low Whoosh, Magic Spell, Male Jump 1, Male Jump 2, Moo, Ocean Wave, Oops, Orchestra Tuning, Party Blower, Pew, Ping Pong Hit, Plane Flying By, Plane Motor Running, Plane Starting, Pluck, Police Siren 1, Police Siren 2, Police Siren 3, Punch, Rain, Ricochet, Rimshot, Ring Tone, Rip, Robot 1, Robot 2, Robot 3, Rocket Explosion Rumble, Rooster, Scrambling Feet, Screech, Seagulls, Service Announcement, Sewing Machine, Ship Bell, Siren Whistle, Skid, Slide Whistle 1, Slide Whistle 2, Sneaker Squeak, Snoring, Snort, Space Ambience, Space Flyby, Space Noise, Splash, Sport Whistle 1, Sport Whistle 2, Squeaky Toy, Squish Pop, Suction Cup, Tada, Telephone Ring 2, Telephone Ring, Teleport 2, Teleport 3, Teleport, Tennis Hit, Thunder Storm, Toliet Flush, Toy Honk, Toy Zing, Traffic, Train Breaks, Train Horn 1, Train Horn 2, Train Horn 3, Train On Tracks, Train Signal 1, Train Signal 2, Train Start, Train Whistle, Triumph, Tropical Birds, Wand, Water Drop, Whistle Thump, Whiz 1, Whiz 2, Window Breaks, Win, Wobble, Wood Tap, Zip

Valor predeterminado:	Sin valor
Volumen:	El volume al que se reproducirá el sonido.
Tipo:	Entero (número entero positivo o negativo, incluido el 0.
Valores:	0 a 100%

Errores

TypeError	El nombre no es una cadena o el volumen no es un número entero.
RuntimeError	La aplicación SPIKE se ha desconectado del Hub

Ejemplo:

```
from spike import App

app = App()

app.start_sound('Cat Meow 1')
```

Botones

A continuación te mostramos todas las funciones vinculadas a los botones programables (por ejemplo, los botones izquierdo y derecho) del Hub de SPIKE Prime.

Eventos

wait_until_pressed()

Espera hasta que se presiona el botón.

Ejemplo:

```
from spike import PrimeHub

hub = PrimeHub()


# un pitido cada vez que se presiona el botón izquierdo

while True:
    hub.left_button.wait_until_pressed()
    hub.speaker.start_beep()
    hub.left_button.wait_until_released()
    hub.speaker.stop()
```

wait_until_released()

Espera hasta que se suelta el botón.

Ejemplo:

```
from spike import PrimeHub

hub = PrimeHub()


# un pitido cada vez que se presiona el botón

while True:
    hub.left_button.wait_until_pressed()
    hub.speaker.start_beep()
    hub.left_button.wait_until_released()
    hub.speaker.stop()
```

was_pressed()

Comprueba si se ha presionado el botón desde la última vez que se empleó este método.

Una vez que este método indique un valor de “verdadero”, el botón debe liberarse y presionarse de nuevo antes de que este método indique nuevamente un valor de verdadero.

Indica

Si se ha presionado el botón, de los contrario.

Tipo: Booleano

Valores: True or False

Ejemplo:

```
from spike import PrimeHub
from spike.control import
wait_for_seconds

hub = PrimeHub()

while True:
    wait_for_seconds(5)
    if
hub.left_button.was_pressed():

        # haz algo
```

Medidas

is_pressed()

Comprueba si se presiona el botón.

Indica

“Verdadero” si se presiona el botón, de lo contrario “falso”.

Tipo: Booleano

Valores: True or False

Ejemplo:

```
from spike import PrimeHub

hub = PrimeHub()

if hub.left_button.is_pressed():

    # haz algo
```

Sensor de color

Para poder utilizar el Sensor de color, debes inicializarlo.

Ejemplo:

```
from spike import ColorSensor


# Inicializa el Sensor de color
color = ColorSensor('E')
```

A continuación te mostramos todas las funciones vinculadas al Sensor de color.

Medidas

get_color()

Obtiene el color detectado de una superficie.

Indica

Nombre del color.

Tipo: Cadena (Texto)

Valores: 'black','violet','blue','cyan','green','yellow','red','white','None'

Errores

RuntimeError

El sensor se ha desconectado del puerto.

Ejemplo:

```
from spike import ColorSensor


# Inicializa el Sensor de color

paper_scanner = ColorSensor('E')


# Mide el color

color = paper_scanner.get_color()


# Traslada el nombre del color a la consola

print('Detected:', color)


# Comprueba si se trata de un color específico

if color == 'red':
    print('It is red!')
```

get_ambient_light()

Obtiene la intensidad de la luz ambiental.

Esto hace que el Sensor de color cambie de modo, lo que puede afectar a tu programa de formas inesperadas. Por ejemplo, cuando el Sensor de color está en modo de luz ambiente no puede detectar colores.

Indica

La intensidad de luz ambiente.

Tipo: entero (número entero positivo o negativo, incluyendo 0)

Valores: 0 a 100%

Errores

RuntimeError

El sensor se ha desconectado del puerto.

get_refleted_light()

Obtiene la intensidad de luz reflejada.

Indica

La intensidad de luz reflejada.

Tipo: Entero (Número entero positivo o negativo, incluyendo 0)

Valores: 0 a 100%

Errores

RuntimeError

El sensor se ha desconectado del puerto.

get_rgb_intensity()

Obtiene la intensidad de color rojo, verde, azul y la intensidad global.

Indica

Tipo: tupla de int (En Python, una tupla es un conjunto ordenado e inmutable de elementos del mismo o diferente tipo.)

Valores: Red, green, blue and overall Intensity (0-1024) Intensidad general.
Errores

RuntimeError

El sensor se ha desconectado del puerto.

get_red()
Obtiene la intensidad del color rojo.

Indica

Tipo: Entero(número entero positivo o negativo, incluyendo 0)

Valores: 0 a 1024

Errores

RuntimeError

El sensor se ha desconectado del puerto.

get_green()
Obtiene la intensidad del color verde.

Indica

Tipo: Entero(número entero positivo o negativo, incluyendo 0)

Valores: 0 a 1024´

Errores

RuntimeError

El sensor se ha desconectado del puerto.

get_blue()
obtiene la intensidad del color azul

Indica

Tipo: Entero (número entero positivo o negativo incluyendo 0)´

Valores: 0 a 1024

Errores

RuntimeError

El sensor se ha desconectado del puerto.

Eventos

wait_until_color()
wait_until_color(color)

Espera hasta que el Sensor de color detecta el color especificado.

Parámetros

Color

El nombre del color

Tipo:

Cadena (texto)

Valores: ‘negro’, ‘violeta’,’azul’,’cian’,’verde’,’amarillo’,’rojo’,’blanco’,Ninguno

Valor predeterminado: sin valor predeterminado.

Errores

TypeError

El color no es una cadena o Ninguno.

ValueError

El color no es uno de los valores permitidos.

Runtime

El sensor se ha desconectado del puerto.

Ejemplo:

```
from spike import ColorSensor

color_sensor = ColorSensor('A')

color_sensor.wait_until_color('blue')

# Añadir acciones a continuación
```

wait_for_new_color()

Acciones

light_up_all()

light_up_all(brightness=100)

Ilumina todas las luces del Sensor de color con un brillo especificado.

Esto hace que el Sensor de color cambie de modo, lo que puede afectar a tu programa de formas inesperadas. Por ejemplo, cuando el Sensor de color está en modo de iluminación no puede detectar colores.

Parámetros

Brighness

El brillo deseado de las luces en el sensor de color.

Tipo: 0 a 100% (0 es apagado y 100 es brillo completo)

Valor predeterminado: 100%

Errores

TypeError

El brillo no es un número entero.

RuntimeError

El sensor se ha desconectado del puerto

Ejemplo:

```
from spike import ColorSensor

color_sensor = ColorSensor('A')


# Enciende las luces del Sensor de color

color_sensor.light_up_all()


# Apaga las luces del Sensor de color

color_sensor.light_up_all(0)
```

light_up()

light_up(light_1,light_2,light_3)

Establece el brillo de las luces individuales en el Sensor de color.

Esto hace que el Sensor de color cambie de modo, lo que puede afectar a tu programa de formas inesperadas. Por ejemplo, cuando el Sensor de color está en modo de iluminación no puede detectar colores.

Parámetros

light_1

El brillo deseado de la luz 1.

Tipo: entero (número entero positivo o negativo, incluyendo 0)

Valores: 0 a 100% (0 es apagado y 100 es brillo completo)

Valor predeterminado: 100%

light_2

El brillo deseado de la luz 2.

Tipo: entero (número entero positivo o negativo, incluyendo 0)

Valores: 0 a 100% (0 es apagado y 100 es brillo completo)

Valor predeterminado: 100%

light_3

El brillo deseado de la luz 3.

Tipo: entero (número entero positivo o negativo, incluyendo 0)

Valores: 0 a 100% (0 es apagado y 100 es brillo completo)

Valor predeterminado: 100%

Errores

TypeError

Light_1, light_2 o light_3 no son enteros.

RuntimeError

El sensor se ha desconectado del puerto

Ejemplo:

```
from spike import ColorSensor

color_sensor = ColorSensor('A')


# Enciende una luz (light_2) en el Sensor de color con el
máximo brillo

color_sensor.light_up(0, 100, 0)
```

Sensor de distancia

Para poder utilizar el Sensor de distancia debes inicializarlo.

Ejemplo:

```
from spike import DistanceSensor


# Inicializa el Sensor de
distancia
distance = DistanceSensor('A')
```

A continuación te mostramos todas las funciones vinculadas al Sensor de distancia.

Acciones

light_up_all()

light_up_all(brightness=100)

Ilumina todas las luces del Sensor de distancia con el brillo especificado.

Parámetros

Brightness

El brillo especificado de todas las luces.

Tipo: entero (número entero positivo o negativo, incluyendo 0)

Valores: 0 a 100% (0 es apagado y 100 es brillo completo.

Valor predeterminado: 100

Errores

TypeError

El brillo no es un número.

RuntimeError

El sensor se ha desconectado del puerto.

Ejemplo:

```
from spike import
DistanceSensor

distance_sensor =
DistanceSensor('A')
```

```
# Enciende las luces

distance_sensor.light_up_all()


# Apaga las luces

distance_sensor.light_up_all(0)
```

light_up()

light_up(right_top, left_top, right_bottom, left_bottom)

Establece el brillo de las luces individuales en el Sensor de distancia.

Tipo: entero (número positivo o negativo, incluyendo 0)

Valores: 0 a 100% (0 es apagado y 100 es brillo completo)

Valor predeterminado: 100

Parámetros

right_top

El brillo de la luz sobre la parte derecha del Sensor de distancia.

Tipo: entero (número positivo o negativo, incluyendo 0)

Valores: 0 a 100% (0 es apagado y 100 es brillo completo)

Valor predeterminado: 100

left_top

El brillo de la duz debajo de parte derecha del Sensor de distancia.

Tipo: entero (número positivo o negativo, incluyendo 0)

Valores: 0 a 100% (0 es apagado y 100 es brillo completo)

Valor predeterminado: 100

right_bottom

El brillo de la luz por debajo de la parte izquierda del Sensor a distancia.

Tipo: entero (número positivo o negativo, incluyendo 0)

Valores: 0 a 100% (0 es apagado y 100 es brillo completo)

Valor predeterminado: 100

left_bottom

El brillo de la luz por debajo de la parte izquierda del Sensor a distancia.

Tipo: entero (número positivo o negativo, incluyendo 0)

Valores: 0 a 100% (0 es apagado y 100 es brillo completo)

Valor predeterminado: 100

Errores

TypeError

Right_top, left_top, right_bottom, left_bottom no es un número.

RuntimeError

El sensor se ha desconectado del puerto.

Ejemplo:

```
from spike import
DistanceSensor

distance_sensor =
DistanceSensor('A')


# Enciende las luces superiores
del Sensor de distancia

distance_sensor.light_up(100,
100, 0, 0)
```

Medidas

get_distance_cm()

get_distance_cm(short_range=false)

Obtiene la distancia medida en centímetros.

Parámetros

Short_range

Si se debe utilizar o no el modo de corto alcance. El modo de corto alcance aumenta la precisión pero solo puede detectar los objetos cercanos.

Tipo: Boolean

Valores: Verdadero o falso

Valor predeterminado: Falso

Indica

La distancia medida o “ninguna” si esta no se puede medir.

Tipo: flotante (número decimal)

Valores: 0 hasta 200 cm

Errores

TypeError

Short_range no es Boolean

RuntimeError

El sensor se ha desconectado del puerto

Ejemplo:

```
from spike import DistanceSensor


# Inicializa el Sensor de distancia

wall_detector = DistanceSensor('E')


# Mide las distancias entre el Sensor de distancia y un objeto en centímetros o pulgadas.

dist_cm = wall_detector.get_distance_cm()
dist_inches = wall_detector.get_distance_inches()


# Traslada ambos resultados a la consola

print('cm:', dist_cm, 'Inches:', dist_inches)

```

get_distance_inches()

get_distance_inches(short_range=false)

Obtiene la distancia media en pulgadas.

Parámetros

short_range

Si se debe utilizar o no el modo corto alcance. El modo de corto alcance aumenta la precisión pero solo puede detectar los objetos cercanos.

Tipo: Boolean

Valores: Verdadero o falso

Valor predeterminado: Falso

Indica

La distancia medida o “ninguna” si esta no se puede medir.

Tipo: flotante (número decimal)

Valores: cualquier valor entre 0 y 79

Errores

TypeError

Short_range no es Boolean

RuntimeError

El sensor se ha desconectado del puerto

get_distance_percentage()

get_distnce_percentage(short_range=False)

Obtiene la distancia medida en porcentaje.

Parámetros

short_range

Si se debe utilizar o no el modo de corta alcance. El modo de corto alcance aumenta la precisión pero solo puede detectar los objetos cercanos.

Tipo: Boolean

Valores: Verdadero o falso

Valor predeterminado: Falso

Indica

La distancia medida a “ninguna” si esta no se puede medir.

Tipo: entero (número entero positivo o negativo, incluyendo 0)

Valores: Verdadero o Falso

Valor predeterminado: Falso

Indica

La distancia medida o “ninguna” si esta no se puede medir.

Tipo: entero (número entero positivo o negativo, incluyendo 0)

Valores: Cualquier valor entre 0 y 100.

Errores

TypeError

Short_range no es Boolean.

RuntimeError

El sensor se ha desconectado del puerto

Eventos

wait_for_distance_farther_than()

wait_for_distance_farther_than(distance, unit='cm', short_rangue=False)

Espera hasta que la distancia medida sea mayor que la distancia.

Parámetros

distance

La distancia objetivo que se debe detectar del sensor a un objeto.

Tipo: float (número decimal)

Valores: cualquier valor.

Valor predeterminado: sin valor predeterminado.

unit

Unidad de medida de la distancia.

Tipo: cadena (texto)

Valores: 'cm','in','%'

Valor predeterminado: cm

short_range

Si se debe utilizar o no el modo de corto alcance. El mod de corto alcance aumenta la precisión pero solo puede detectar los objetos cercanos.

Tipo: boolean

Valores: Verdadero o falso

Valor predeterminado: Falso

Errores

TypeError

La distancia no es un número o la unidad no es una cadena o short_range no es un booleano.

ValueError

Unidad no es uno de los valores permitidos.

RuntimeError

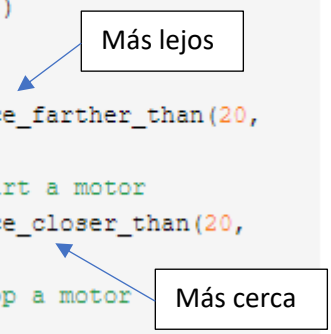
El sensor se ha desconectado del puerto.

Ejemplo:

```
from spike import DistanceSensor

distance_sensor = DistanceSensor('A')

while True:
    distance_sensor.wait_for_distance_farther_than(20,
'cm')
    # do something, for example, start a motor
    distance_sensor.wait_for_distance_closer_than(20,
'cm')
    # do something, for example, stop a motor
```



wait_for_distance_closer_than()

wait_for_distance_closer_than(distance, unit='cm', short_range=False)

Espera hasta que la distancia medida sea menor que la distancia.

Parámetros

distance

La distancia objetivo que se debe detectar del sensor a un objeto.

Tipo: float (número decimal)

Valores: cualquier valor.

Valor predeterminado: sin valor predeterminado.

unit

Unidad de medida de la distancia.

Tipo: cadena (texto)

Valores: 'cm','in','%'

Valor predeterminado: cm

short_range

Si se debe utilizar o no el modo de corto alcance. El mod de corto alcance aumenta la precisión pero solo puede detectar los objetos cercanos.

Tipo: boolean

Valores: Verdadero o falso

Valor predeterminado: Falso

Errores

TypeError

La distancia no es un número o la unidad no es una cadena o short_range no es un booleano.

ValueError

Unidad no es uno de los valores permitidos, short_range no es un booleano.

RuntimeError

El sensor se ha desconectado del puerto.

Ejemplo:

```
from spike import DistanceSensor

distance_sensor = DistanceSensor('A')

while True:
    distance_sensor.wait_for_distance_farther_than(20,
'cm')
    # do something, for example, start a motor
    distance_sensor.wait_for_distance_closer_than(20,
'cm')
    # do something, for example, stop a motor
```

Sensor de fuerza

Para poder utilizar el Sensor de fuerza, debes inicializarlo.

Ejemplo:

```
from spike import ForceSensor


# Inicializa el sensor de fuerza.
force = ForceSensor('E')
```

A continuación te mostramos todas las funciones vinculadas al Sensor de fuerza.

Medidas

is_pressed()

Comprueba si se presiona el botón del sensor.

Indica

“verdadero” si se presiona el botón.

Tipo: booleano

Valores: True or false

Errores

RuntimeError

El sensor de fuerza se ha desconectado del pueto.

Ejemplo:

```
from spike import ForceSensor


# Inicializa el sensor de
fuerza.

door_bell = ForceSensor('E')


# Comprueba si se presiona el
Sensor de fuerza

if door_bell.is_pressed():
    print('Hello!')
```

get_force_newton()

Obtiene la fuerza en newtons

Indica

La fuerza medida en newtons.

Tipo: flotante (número decimal)

Valores: Entre 0 y 10

Errores

RuntimeError

El sensor de fuerza se ha desconectado del puerto.

Ejemplo:

```
from spike import ForceSensor


# Inicializa el sensor de fuerza.

door_bell = ForceSensor('E')


# Mide la fuerza en newtons o como porcentaje.

newtons = door_bell.get_force_newton()
percentage = door_bell.get_force_percentage()
```

```
# Traslada ambos resultados

print('N:', newtons, '=', percentage, '%')


# Comprueba si se presiona el Sensor de fuerza

if door_bell.is_pressed():
    print('Hello!')
```

get_force_percentage()

Obtiene la fuerza medida como un porcentaje de la fuerza máxima.

Indica

La fuerza medida en porcentaje.

Tipo: entero (número positivo o negativo, incluyendo 0)

Valores: entre 0 – 100%

Errores

RuntimeError

El sensor de fuerza se ha desconectado del puerto

Eventos

wait_until_pressed()

Espera hasta que se presiona el Sensor de fuerza.

Errores

RuntimeError

El sensor se ha desconectado del puerto

Ejemplo:

```
from spike import ForceSensor

force_sensor = ForceSensor('A')

while True:
    force_sensor.wait_until_pressed()
    # do something, for example, start a motor
    force_sensor.wait_until_released()
    # do something, for example, stop a motor
```

wait_until_released()

Espera hasta que se libera el Sensor de fuerza

Errores

RuntimeError

El sensor se ha desconectado del puerto

Ejemplo:

```
from spike import ForceSensor

force_sensor = ForceSensor('A')

while True:
    force_sensor.wait_until_pressed()
    # do something, for example, start a motor
    force_sensor.wait_until_released()
    # do something, for example, stop a motor
```

Matriz de luces

A continuación te mostramos todas las funciones vinculadas a la matriz de luces del Hub de SPIKE Prime.

Acciones

show_image()

show_image(image, brightness=100)

Muetra una imagen en la matriz de luces.

Parámetros

Image

Nombre de la imagen.

Tipo: Cadena (texto)

Valores: ANGRY, ARROW_E, ARROW_N, ARROW_NE, ARROW_NW, ARROW_S, ARROW_SE, ARROW_SW, ARROW_W, ASLEEP, BUTTERFLY, CHESSBOARD, CLOCK1, CLOCK10, CLOCK11, CLOCK12, CLOCK2, CLOCK3, CLOCK4, CLOCK5, CLOCK6, CLOCK7, CLOCK8, CLOCK9, CONFUSED, COW, DIAMOND, DIAMOND_SMALL, DUCK, FABULOUS, GHOST, GIRAFFE, GO_RIGHT, GO_LEFT, GO_UP, GO_DOWN, HAPPY, HEART, HEART_SMALL, HOUSE, MEH, MUSIC_CROTCHET, MUSIC_QUAVER, MUSIC_QUAVERS, NO, PACMAN, PITCHFORK, RABBIT, ROLLERSKATE, SAD, SILLY, SKULL, SMILE, SNAKE, SQUARE, SQUARE_SMALL, STICKFIGURE, SURPRISED, SWORD, TARGET, TORTOISE, TRIANGLE, TRIANGLE_LEFT, TSHIRT, UMBRELLA, XMAS, YES

Valor predeterminado: Sin valor predeterminado

Brightness

Brillo de la imagen

Tipo: entero (número entero positivo o negativo, incluyendo 0)

Valores: 0 a 100%

Valor predeterminado: 100

Errores

TypeError

La imagen no es una cadena o el brillo no es un número entero.

ValueError

Imagen no es uno de los valores permitidos

Ejemplo:

```
from spike import PrimeHub
from spike.control import wait_for_seconds

hub = PrimeHub()

hub.light_matrix.show_image('HAPPY')
wait_for_seconds(5)
hub.light_matrix.show_image('ASLEEP')
wait_for_seconds(5)
```

set_pixel()

set_pixel(x,y,brightness=100)

Establece el brillo de un pixel (uno de los 25 LED) de la matriz de luces.

Parámetros

x

Posición del píxel contando desde la izquierda.

Tipo: entero (número entero positivo o negativo, incluyendo 0)

Valores: 1 a 5

Valor predeterminado: sin valor predeterminado.

y

Posición del pixel contando desde la izquierda.

Tipo: entero (número entero positivo o negativo, incluyendo 0)

Valores: 1 a 5

Valor predeterminado: sin valor predeterminado

Brigtness

Brillo del píxel

Tipo: entero (Número entero positivo o negativo, incluyendo 0)

Valores: 1 a 100%

Valor predeterminado: 100

Errores

TypeError

(x, y) el brillo no es un número entero.

ValueError

(x,y) no está dentro del rango permitido de 0-4.

Ejemplo:

```
from spike import PrimeHub

hub = PrimeHub()

hub.light_matrix.set_pixel(1, 4)
```

write()

write(text)

Escribe texto en la Matriz de luces, letra a letra desplazándose de derecha a izquierda.

Tu programa no continuará hasta que no se hayan mostrado todas las letras.

Parámetros

text

Texto que se quiere escribir.

Tipo: cadena (texto)

Valores: Cualquier texto.

Valor predeterminado: Sin valor predeterminado

Ejemplo:

```
from spike import PrimeHub

hub = PrimeHub()

hub.light_matrix.write('Hello!')


# Muestra el número 1 en la matriz de luces

hub.light_matrix.write('1')
```

off()

Apaga todos los píxeles de la matriz de luces.

Ejemplo:

```
from spike import PrimeHub

hub = PrimeHub()

hub.light_matrix.off()
```

Funciones matemáticas

El módulo matemático ofrece algunas funciones matemáticas básicas para trabajar con números en punto flotante.

acos()	**acos(x)** Indica el coseno inverso de x.
acosh()	**acosh(x)** Indica el coseno hiperbólico inverso de x.
asin()	**asin(x)** Indica el seno de x.
asinh()	**asinh(x)** Indica el seno hiperbólico de x.
atan()	**atan(x)** Indica la tangente inversa de x.
atan2()	**atan2(y,x)** Indica el valor principal de la tangente inversa de y/x
atanh()	**atanh(x)** Indica la tangente hiperbólica de x.
ceil()	**ceil(x)** Indica un entero, redondeándose x al infinito positivo.
copysign()	**copysign(x,y)** Indica x con el signo de y.
cos()	**cos(x)** Indica el coseno de x.
cosh()	**cosh(x)** Indica el coseno hiperbólico de x.
degrees()	**degrees(x)** Indica x radianes convertidos a grados.
erf()	**erf(x)** Indica la función de error de x.
erfc()	**erfc(x)** Indica la función de error complementaria de x.
exp()	**exp(x)** Indica la exponencial de x.
expm1()	**expm1(x)** Indica exp(x) – 1.
fabs()	**fabs(x)** Indica el valor absoluto de x.
floor()	**floor(x)** Indica un entero, redondeándose x al infinito negativo.
fmod()	**fmod(x,y)** Indica el resto de x/y.
frexp()	**frexp(x)** Descompone un número en punto flotante en su mantisa y su exponente. El valor indicado es la tupla (m,e), de tal manera que x == m*2**e exactamente. Si x == 0 entonces la función indica (0,0 0), de lo contrario se mantiene la relación 0,5 <= abs(m) <1.
gamma()	**gamma(x)** Indica la función gamma de x.
isfinite()	**isfinite(x)** Indica “verdadero” si x es finito.
isinf()	**isinf(x)** Indica “verdadero” si x es infinito.
isnan()	**isnan(x)** Indica “verdadero” si x es un número.
ldexp()	**ldexp(x, exp)** Indica x * (2**exp)
lgamma()	**lgamma(x)** Indica el logaritmo natural de la función gamma de x.
log()	**log(x)** Indica el logaritmo na de x.
log10()	**log10(x)** Indica el logaritmo de base 10 de x.
log2()	**log2(x)** Indica el logaritmo de base 2 de x.
modf()	**modf(x)** Indica una tupla de dos números en punto flotante, siendo las partes fraccionaria e integral de x. Ambos valores indicados tienen el mismo signo que x.
pow()	**pow(x,y)** Indica x a la potencia de y.
radians()	**radians(x)** Indica x grados convertidos a radianes.
sin()	**sin(x)** Indica el seno de x.
sinh()	**sinh(x)** Indica el seno hiperbólico de x.
sqrt()	**sqrt(x)** Indica la raíz cuadrada de x.
tan()	**tan(x)** Indica la tangente de x.
tanh()	**tanh(x)** Indica la tangente hiperbólica de x.
trunc()	**trunc(x)** Indica un entero, redondeándolo x a 0.

Constantes

e	**e**	La constante matemática e = 2,718282…, con la precisión disponible.
pi	**pi**	La constante matemática π = 3.141592…, con la precisión disponible.

Sensor de movimiento

A continuación te mostramos todas las funciones vinculadas al Sensor de movimiento, que combina un acelerómetro de tres ejes y un sensor giroscópico de tres ejes en el Hub.

Eventos

was_gesture()

was_gesture(gesture)

Comprueba si se ha producido un gesto desde la última vez que se utilizó was_gesture() o desde el principio del programa (en el primer uso).

Parámetros

gesture

El nombre del gesto.

Tipo: cadena (texto)

Valores: se agita, se toca, se toca dos veces, cae, ninguno.

Valor predeterminado: Sin valor predeterminado.

Errores

TypeError

El gesto no es un String.

ValueError

El gesto no es uno de los valores permitidos.

Indica

“Verdadero” si el gesto se ha producido desde la última vez que se empleó was_gesture(), de los contrario falso.

Tipo: booleano

Valores: True or false

Ejemplo:

```
from spike import PrimeHub
from spike.control import wait_for_seconds

hub = PrimeHub()
```

```
wait_for_seconds(5)
if hub.motion_sensor.was_gesture('shaken'):
    # the Hub was shaken some time within the
last 5 seconds
```

El Hub se agitó algún tiempo en los últimos 5 segundos.

wait_for_new_gesture()

Espera hasta que se produce un nuevo gesto.

Indica

El nuevo gesto.

Tipo: cadena (texto)

Valores: 'shaken', 'tapped', 'doubletapped','falling'

'agitada','tocado','doble toque','caido',

Ejemplo:

```
from spike import PrimeHub

hub = PrimeHub()

gesture = hub.motion_sensor.wait_for_new_gesture()
if gesture == 'shaken':
    # do one thing
elif gesture == 'tapped':
    # do another thing
```

wait_for_new_orintation()

Espera hasta que cambia la orientación del Hub.

La primera vez que se emplee este método indicará inmediatamente el valor actual. Luego, emplear este método bloqueará el programa hasta que la orientación del Hub haya cambiado desde la última vez que se empleó este método.

Indica

La nueva orientación del Hub.

Tipo: cadena (texto)

Valores: 'front','back','up','down','leftside','righside'

'parte delantera','parte trasera','arriba','abajo','lado izquierdo','lado derecho'

Ejemplo:

```
from spike import PrimeHub

hub = PrimeHub()

orientation = hub.motion_sensor.wait_for_new_orientation()
if orientation == 'leftside':
    # do one thing
elif orientation == 'rightside':
    # do another thing
```

Medidas

get_orientation()

Obtiene la orientación del Hub.

Indica

La orientación actual del Hub.

Tipo: cadena (texto)

Valores: 'front','back','up','down','leftside','rightside'

'parte delantera','parte trasera','arriba','abajo','lado izquierdo','lado derecho'

Ejemplo

```
from spike import PrimeHub

hub = PrimeHub()

if hub.motion_sensor.get_orientation() == 'front':
    # do something
```

get_gesture()

Obtiene el último gesto detectado.

Indica

El gesto.

Tipo: cadena (texto)

Valores: 'shaken','tapped','doubletapped','falling'

'agitado','tocado','doble toque','caido'

Ejemplo

```
from spike import PrimeHub
```

```
hub = PrimeHub()

while True:
  if hub.motion_sensor.get_gesture() == 'falling':
    print("Aaah!")
```

get_roll_angle()

Obtiene el ángulo de alabeo del Hub.

“Alabeo” es la rotación en torno al eje anterior y posterior (longitudinal).
“Guiñada” es la rotación en torno al eje superior e inferior (vertical).
“Cabeceo” en la rotación en torno al eje izquierdo-derecho (transversal)

Indica

El alebeo de giro especificado en grados.

Tipo: Entero (número entero positivo o negativo, incluyendo 0)

Valores: -180 hasta 180.

Ejemplo

```
from spike import PrimeHub

hub = PrimeHub()

if hub.motion_sensor.get_roll_angle() > 90:
    # do something
```

get_pitch_angle()

Obtiene el ángulo de cabeceo del Hub.

“Cabeceo” en la rotación en torno al eje izquierdo-derecho (transversal)
“Alabeo” es la rotación en torno al eje anterior y posterior (longitudinal).
“Guiñada” es la rotación en torno al eje superior e inferior (vertical).

Indica

El ángulo de cabeceo especificado en grados.

Tipo: Entero (número entero positivo o negativo, incluyendo 0)

Valores: -180 hasta 180.

Ejemplo

```
from spike import PrimeHub

hub = PrimeHub()
```

```
if hub.motion_sensor.get_pitch_angle() > 90:
    # do something
```

get_yaw_angle()

Obtiene el ángulo de guiñada del Hub.

“Guiñada” es la rotación en torno al eje superior e inferior (vertical).
“Cabeceo” en la rotación en torno al eje izquierdo-derecho (transversal)
“Alabeo” es la rotación en torno al eje anterior y posterior (longitudinal).

Indica

El ángulo de guiñada expresado en grados.

Tipo: Entero (número entero positivo o negativo, incluyendo 0)

Valores: -180 hasta 180.

Ejemplo

```
from spike import PrimeHub

hub = PrimeHub()

if hub.motion_sensor.get_yaw_angle() > 90:
    # do something
```

Ajustes

reset_yaw_angle()

Establece el ángulo de quiñada a 0.

Ejemplo:

```
from spike import PrimeHub

hub = PrimeHub()

hub.motion_sensor.reset_yaw_angle()
angle = hub.motion_sensor.get_yaw_angle()
print('Angle:', angle)

# El ángulo es ahora 0
```

Parejas de motor

Los objetos de pareja de motor se utilizan para controlar dos motores simultáneamente en direcciones opuestas.

Para poder utilizar las funciones de pareja de motor, debes inicializar los dos motores.

Ejemplo:

```
from spike import MotorPair


# Si el motor izquierdo está
conectado al puerto B

# Y el motor derecho está
conectado al puerto A.
motor_pair = MotorPair('B', 'A')
```

Acciones

move()

move(amount, unit='cm', steering=0, speed=None)

Enciende simultáneamente los dos motores para desplazar una base de conducción.

Un valor de dirección = 0 hace que la Base de conducción se desplace en línea recta. Los números negativos hacen que la Base de conducción gire hacia la izquierda. Los números positivos hacen que la Base de conducción gire hacia la derecha.

El programa no continuará hasta que no se haya alcanzado la cantidad.

Si el valor de la dirección es igual a -100 o 100, la base de conducción realizará una rotación sobre si misma (movimiento de una rueda hacia adelante, una rueda hacia atrás) con la velocidad predeterminada en cada motor.

Si el valor de dirección está fuera del intervalo permitido, el valor se establecerá en -100 o 100 dependiendo de si el valor es positivo o negativo.

Si la velocidad es negativa, entonces la Base de conducción se moverá hacia atrás en lugar de hacia adelante. De la misma manera, si la cantidad es negativa, la Base de conducción se moverá hacia atrás en lugar de hacia adelante. Si tanto la velocidad como la cantidad son negativas, entonces la Base de conducción se moverá hacia adelante.

Cuando la unidad es "cm" 0 "in" pulgadas, la cantidad de parámetro de unidad es la distancia horizontal que la Base de conducción recorrerá antes de detenerse. La relación entre las rotaciones de motor y la distancia recorrida pueden ajustarse empleando set_motor_rotation().

Cuando la "unidad" sea "rotaciones" o "grados", el valor de parámetro de cantidad especificada cuánto girará el eje del motor antes de detenerse.

Cuando la unidad sea “segundos”, el valor de parámetro de cantidad especificada la cantidad de tiempo que se accionarán los motores antes de detenerse.

Parámetros

Amount

La cantidad que debe desplazarse en relación con la unidad de media especificada.

Tipo: float (número decimal)

Valores: cualquier valor.

Valor: sin valor

Valor predeterminado: cm

steering

El sentido y la cantidad para la dirección de la Base de conducción.

Tipo: Entero (número entero positivo o negativo, incluyendo 0)

Valores: -100 a 100

Valor: La velocidad establecida por el parámetro de velocidad, set_default_spped()

Errores

TypeError

La cantidad no es un número, o la dirección o la velocidad no es un número entero, o la unidad no es una cadena.

ValueError

Unidad no es uno de los valores permitidos.

RuntimeError

Uno o ambos motores se han desconectado o los motores no se pudieron emparejar.

Ejemplo:

```
import math
from spike import MotorPair

motor_pair = MotorPair('B', 'A')


# Haz que una Base de conducción gire 180 grados sobre sí
misma (si las ruedas están separadas por 8,1 cm)

motor_pair.move(8.1 * math.pi / 2, 'cm', steering=100)
```

start()

start(steering=0, speed=None)

Enciende simultáneamente los dos motores para desplazar una Base de conducción.

Un valor de dirección = 0 hace que la Base de conducción se desplace en línea recta. Los números negativos hacen que la Base de conducción gire hacia la izquierda. Los números positivos hacen que la Base de conducción gire hacia la derecha.

El flujo del programa no se interrumpe. Esto es así a menos que se emplee una entrada de sensor y una condición.

Si el valor de la dirección es igual a -100 o 100, la base de conducción realizará una rotación sobre sí misma (movimiento de un rueda hacia delante, una rueda hacia atrás) con la velocidad predeterminada en cada motor.

Si el valor de dirección está fuera de intervalo permitido, el valor se establecerá en -100 o 100 dependiendo de si el valor es positivo o negativo.

Si la velocidad está fuera del intervalo permitido, el valor se establecerá en -100 o 100 dependiendo de si el valor es positivo o negativo.

Si la velocidad es negativa, entonces la Base de conducción se moverá hacia atrás en lugar de hacia adelante. De la misma manera, si la cantidad es negativa, la Base de conducción se moverá hacia atrás en lugar de hacia adelante. Si tanto la velocidad como la cantidad son negativas, entonces la Base de conducción se moverá hacia delante.

Parámetros

steering

El sentido y la cantidad para la dirección de la Base de conducción.

Tipo: entero (número entero positivo o negativo, incluyendo 0)

Valores: -100 a 100

Valor predeterminado: 0

speed

La velocidad con la que se moverá una Base de conducción mientras ejecuta una curva.

Tipo: entero (número entero positivo o negativo, incluyendo 0)

Valores: -100 a 100%

Valor predeterminado : si no se especifica ningún valor, utilizará la velocidad predeterminada establecida por **set_default_speed().**

Errores

TypeError

La dirección o la velocidad no es un número entero.

RuntimeError

Uno o ambos motores se has desconectado o los motores no se pudieron emparejar.

Ejemplo:

```
from spike import MotorPair

motor_pair = MotorPair('B', 'A')

motor_pair.start()

# Espera a algo...

motor_pair.stop()
```

stop()

Detiene los dos motores simultáneamente, lo que detendrá una Base de conducción.

Los motores mantendrán activamente su posición actual o se moverán en punto muerto dependiendo de la opción seleccionada por **set_stop_action().**

Ejemplo:

```
from spike import MotorPair

motor_pair = MotorPair('B', 'A')

motor_pair.start()

# Espera a algo...

motor_pair.stop()
```

move_tank()

move_tank(amount, unit='cm', left_speed=None, right_speed=None)

Mueve la Base de conducción mediante la dirección diferencial (una rueda hacia delante, una rueda hacia atrás).

La velocidad de cada motor puede controlarse de forma independiente para las Bases de conducción de accionamiento diferencial (una rueda hacia delante, un rueda hacia atrás).

Cuando la unidad es "cm" o "in", la cantidad del parámetro de unidad es la distancia horizontal que la Base de conducción recorrerá antes de detenerse. La relación entre las rotaciones de motor y la distancia recorrida pueden ajustarse empleando **set_motor_rotation().**

Cuando la "unidad" sea "rotaciones" o "grados", el valor de parámetro de cantidad especificada cuánto girará el eje del motor antes de detenerse.

Cuando la unidad sea “segundos”, el valor de parámetro de cantidad especifica la cantidad de tiempo que se accionarán los motores antes de detenerse.

Si left_speed o right_speed están fuera del intervalo permitido, el valor se establecerá en -100 o 100 dependiendo de si el valor es positivo o negativo.

Si una de las velocidades es negativa (left_speed o right_speed), entonces el motor con la velocidad negativa se accionará hacia atrás en lugar de hacia delante. Si el valor del parámetro de cantidad es negativo, ambos motores girarán hacia atrás en lugar de hacia adelante. Si ambos valores de velocidad (left_speed o right_speed) son negativos y el valor del parámetro de cantidad es negativo, entonces ambos motores girarán hacia adelante.

El programa no continuará hasta que no se hay alcanzado la cantidad.

Parámetros

amount

La cantidad que debe desplazarse en relación con la unidad de media especificada.

Tipo: float (número decimal)

Valores: Cualquier valor

Valor predeterminado: sin valor predeterminado.

Unit

Las unidades de medida del parámetro de cantidad.

Tipo: cadena (texto)

Valores: ‘cm’,’in’,’rotaciones’,’grados’,’segundos’

Valor predeterminado: cm

Left_speed

La velocidad del motor izquierdo.

Tipo: entero (número entero positivo o negativo, incluyendo 0)

Valores: -100 a 100

Valor Predeterminado: La velocidad establecida por el parámetro de velocidad set_default_speed().

Right_speed

La velocidad del motor derecho.

Tipo: entero (número entero positivo o negativo, incluyendo 0)

Valores: -100 a 100

Valor Predeterminado: La velocidad establecida por el parámetro de velocidad

Valor Predeterminado: La velocidad establecida por el parámetro de velocidad set_default_speed().

Errores

TypeError

Amount, left_speed o right_speed no es un número o la unidad no es una cadena.

ValueError

Unidad no es uno de los valores permitidos.

RuntimeError

Uno o ambos puertos no tienen un motor conectado o los motores no se pudieron emparejar.

Ejemplo

```
from spike import MotorPair

motor_pair = MotorPair('B', 'A')
motor_pair.move_tank(10, 'cm', left_speed=25,
right_speed=75)
```

start_tank()

start_tank(left_speed, right_speed)

Comienza a mover la Base de conducción mediante la dirección diferencial (una rueda hacia adelante, una rueda hacia atrás).

La velocidad de cada motor puede controlar ser de forma independiente para las Bases de conducción de accionamiento diferencial (una rueda hacia adelante, una rueda hacia atrás).

Si left_speed o right_speed está fuera del intervalo permitido, el valor se establecerá en -100 o 100 dependiendo si el valor es positivo o negativo.

Si una velocidad en negativa, entonces los motores se moverá hacia atrás en lugar de hacia adelante.

El flujo del programa no se interrumpe. Esto es así a menos que se emplee una entrada de sensor y una condición.

Parámetros

Left_speed

La velocidad del motor izquierdo.

Tipo: entero (número entero positivo o negativo, incluyendo el 0)

Valores: -100 a 100

Valor predeterminado: sin valor predeterminado.

Right_speed

La velocidad del motor derecho.

Tipo: entero (número entero positivo o negativo, incluyendo el 0)

Valores: -100 a 100

Valor predeterminado: sin valor predeterminado.

Errores

typeError

left_speed o right_speed no es un número entero.

RuntimeError

Uno o ambos puertos no tienen un motor conectado o los motores no se pudieron emparejar.

Ejemplo

```
from spike import MotorPair

motor_pair = MotorPair('B', 'A')


# Haz que la Base de conducción
gire sobre sí misma hacia la
derecha

motor_pair.start_tank(100, -100)
```

start_at_power()

start_ar_power(power, steering=0)

Comienza a mover la Base de conducción sin control de velocidad.

Los motores también se pueden accionar sin control de velocidad. Esto es útil si utilizas tu propio algoritmo de control (por ejemplo un seguidor de líneas proporcional).

Si la dirección está fuera del intervalo permitido, el valor se establecerá en -100 o 100 dependiendo de si el valor es positivo o negativo.

Si la potencia está fuera del intervalo permitido, el valor se establecerá en -100 o 100 dependiendo de si el valor es positivo o negativo.

Si la potencia es negativa la Base de conducción se moverá hacia atrás en lugar de hacia adelante.

El flujo del programa no se interrumpe. Esto es así a menos que se emplee una entrada de sensor y una condición.

Parámetros

power

La cantidad de energía que se quiere enviar a los motores.

Tipo: entero (número entero positivo o negativo, incluyendo 0)

Valores: -100 a 100%

Valor predeterminado: 100

steering

El sentido de dirección (-100 a 100). O hace que la Base de conducción se desplace en línea recta. Los números negativos hacen que la Base de conducción gire hacia la izquierda. Los números positivos hacen que la Base de conducción gire a la derecha.

Tipo: Integer

Valores: -100 a 100

Valor predeterminado: 0

Errores

TypeError

La dirección o la potencia no es un número entero.

RuntimeError

Uno o ambos puertos no tienen un motor conectado o los motores no se pudieron emparejar.

Ejemplo

```
from spike import MotorPair, ColorSensor

motor_pair = MotorPair('B', 'A')
color_sensor = ColorSensor('F')

while True:
    steering = color_sensor.get_reflected_light() - 50
    motor_pair.start_at_power(50, steering)
```

start_tank_at_power()

start_tank_at_power(left_power, right_power)

Comienza a mover la Base de conducción mediante la dirección diferencial (una rueda hacia delante, una rueda hacia atrás) sin control de velocidad.

Los motores también se pueden asociar sin control de velocidad. Esto es útil si utilizas tu propio algoritmo de control (por ejemplo un seguidor de líneas proporcional).

Si la potencia_izquierda o la potencia_derecha está fuera del intervalo permitido, el valor se rodeará -100 o 100 dependiendo de si el valor es positivo o negativo.

Si la potencia en negativa, entonces el motor correspondiente se moverá hacia atrás en lugar de hacia delante.

El flujo del programa no se interrumpe. Esto es así a menos que se emplee una entrada del sensor y una condición.

Parámetros

left_power

La potencia del motor izquierdo

Tipo: Integer

Valores: -100 a 100

Valor predeterminado: Sin valor predeterminado

right_power

La potencia del motor derecho

Tipo: Integer

Valores: -100 a 100

Valor predeterminado: Sin valor predeterminado

Errores

TypeError

left_power o right_power no es un número entero.

RuntimeError

Uno o ambos puertos no tienen un motor conectado o los motores no se pudieron emparejar.

Ejemplo

```
from spike import MotorPair

motor_pair = MotorPair('B', 'A')

# Haz que la Base de conducción gire sobre
sí misma hacia la derecha

motor_pair.start_tank_at_power(100, -100)
```

Medidas

get_default_speed()

Obtiene la velocidad predeterminada del motor.

Indica

La velocidad predeterminada del motor.

Tipo: entero (número positivo o negativo, incluyendo 0)

Valores: -100 hasta 100%

Ajustes

set_motor_rotation()

set_motor_rotation(amount, unit='cm')

Establece la relación de una rotación de motor con respecto a la distancia recorrida.

Si no se utilizan engranajes entre los motores y las ruedas de la Base de conducción, entonces la cantidad es la circunferencia de la rueda.

Emplear este método no afecta a la Base de conducción si ya está funcionando. Solo tendrá efecto la siguiente vez que se utilice unos de los métodos de movimiento o arranque.

Parámetros

amount

La distancia que recorre la Base de conducción cuando ambos motores realizan cada uno una rotación.

Tipo: float (número decimal)

Valores: cualquier valor.

Valor predeterminado: 17,6

Unit

Las unidades de medida del parámetro de cantidad.

Tipo: cadena (texto)

Valores: 'cm', 'in'

Valor predeterminado: cm

Errores

TypeError

La cantidad no es un número o la unidad no es una cadena.

ValueError

Unidad no es uno de los valores permitidos.

Ejemplo

```
import math
from spike import MotorPair

motor_pair = MotorPair('B', 'A')

# Las ruedas de SPIKE Prime cuentan con un diámetro
de 17,6 cm. Multiplicar por π arroja la distancia
recorrida (circunferencia)
```

```
motor_pair.set_motor_rotation(17.6 * math.pi, 'cm')
```

set_default_speed()

set_default_speed(speed)

Establece la velocidad predeterminada del motor.

Si la velocidad está fuera del intervalo permitido, el valor se establecerá en -100 o 100 dependiendo de si el valor es positivo o negativo.

Establecer la velocidad no tendrá ningún efecto hasta que no se emplee uno de los métodos de movimiento o arranque, incluso si la Base de conducción ya está en movimiento.

Parámetros

speed

La velocidad del motor por defecto.

Tipo: entero (número entero positivo o negativo, incluyendo 0)

Valores: -100 a 100

Valor predeterminado: 100

Errores

TypeError

La velocidad no es un número.

set_stop_action()

set_stop_action(action)

Establece la acción del motor que se utilizará cuando se detenga la Base de conducción.

Si la acción es “frenar”, los motores se detendrán rápidamente pero podrán seguir girando libremente.

Si la acción es “mantener”, los motores mantendrán de manera activa su posición actual y no se podrán hacer girar manualmente.

Si la acción se establece como “desplazamiento en punto muerto”, los motores se detendrán lentamente y podrán girarse libremente.

Establecer la acción de parada no ejerce un efecto inmediato en los motores. Se guardará el ajuste y este se utilizará siempre que se emplee el método stop() o cuando uno de los métodos de movimiento se haya completado sin ser interrumpido.

Parámetros

action

La acción deseada de los motores cuando la Base de conducción se detiene.

Tipo: cadena (texto)

Valores: freno, mantener posición, desplazamiento en punto muerto

Valor predeterminado: desplazamiento en punto muerto

Errores

TypeError

La acción no es una cadena.

ValueError

Action no es uno de los valores permitidos

Ejemplo

```
from spike import MotorPair

motor_pair = MotorPair('B', 'A')


# Permite que los motores giren libremente tras
detenerse

motor_pair.set_stop_action('coast')
```

Operadores

greater_than()

greater_than(a,b)

Comprueba si el valor a es mayor que el valor b.

Esto equivale a a>b

Parámetro

a

Cualquier objeto que se pueda comparar con b.

Tipo: Cualquier tipo

Valores: Cualquier valor

Valor predeterminado sin valor predeterminado

b

Cualquier objeto que se pueda comparar con a.

Tipo: Cualquier tipo

Valores: Cualquier valor

Valor predeterminado sin valor predeterminado

Indica

Tipo: booleano

Valores: True if a>b, de lo contrario False

Ejemplo

```
from spike import ColorSensor
from spike.control import wait_until
from spike.operator import greater_than

color_sensor = ColorSensor('A')

wait_until(color_sensor.get_reflected_light,
greater_than, 50)
```

greater_than_or_equal_to()

greater_than_or_equal_to(a,b)

Comprueba si a es mayor o igual que b.

Esto equivale a a>=b

Parámetro

a

Cualquier objeto que se pueda comparar con b.

Tipo: Cualquier tipo

Valores: Cualquier valor

Valor predeterminado sin valor predeterminado

b

Cualquier objeto que se pueda comparar con a.

Tipo: Cualquier tipo

Valores: Cualquier valor

Valor predeterminado sin valor predeterminado

Indica

Tipo: booleano

Valores: True if a>=b, de lo contrario False

Ejemplo

```
from spike import ColorSensor
from spike.control import wait_until
from spike.operator import greater_than_or_equal_to

color_sensor = ColorSensor('A')

wait_until(color_sensor.get_reflected_light,
greater_than_or_equal_to, 50)
```

less_than()

less_than(a,b)

Comprueba si a es menor que b.

Esto equivale a a<b

Parámetro

a

Cualquier objeto que se pueda comparar con b.

Tipo: Cualquier tipo

Valores: Cualquier valor

Valor predeterminado sin valor predeterminado

b

Cualquier objeto que se pueda comparar con a.

Tipo: Cualquier tipo

Valores: Cualquier valor

Valor predeterminado sin valor predeterminado

Indica

Tipo: booleano

Valores: True if a<b, de lo contrario False

Ejemplo

```
from spike import ColorSensor
from spike.control import wait_until
from spike.operator import less_than

color_sensor = ColorSensor('A')

wait_until(color_sensor.get_reflected_light, less_than,
50)
```

less_than_or_equal_to()

less_than_or_equal_to(a,b)

Comprueba si a es menor o igual que b.

Esto equivale a a<=b

Parámetro

a

Cualquier objeto que se pueda comparar con b.

Tipo: Cualquier tipo

Valores: Cualquier valor

Valor predeterminado sin valor predeterminado

b

Cualquier objeto que se pueda comparar con a.

Tipo: Cualquier tipo

Valores: Cualquier valor

Valor predeterminado sin valor predeterminado

Indica

Tipo: booleano

Valores: True if a<=b, de lo contrario False

Ejemplo

```
from spike import ColorSensor
from spike.control import wait_until
from spike.operator import less_than_or_equal_to

color_sensor = ColorSensor('A')

wait_until(color_sensor.get_reflected_light,
less_than_or_equal_to, 50)
```

equal_to()

equal_to(a,b)

Comprueba si a es igual a b.

Esto equivale a a==b

Parámetro

a

Cualquier objeto que se pueda comparar con b.

Tipo: Cualquier tipo

Valores: Cualquier valor

Valor predeterminado sin valor predeterminado

b

Cualquier objeto que se pueda comparar con a.

Tipo: Cualquier tipo

Valores: Cualquier valor

Valor predeterminado sin valor predeterminado

Indica

Tipo: booleano

Valores: True if a==b, de lo contrario False

Ejemplo

```
from spike import ColorSensor
from spike.control import wait_until
from spike.operator import equal_to

color_sensor = ColorSensor('A')

wait_until(color_sensor.get_color, equal_to, 'red')
```

not_equal_to()

not_equal_to(a,b)

Comprueba si a no es igual a b.

Esto equivale a a!=b

Parámetro

a

Cualquier objeto que se pueda comparar con b.

Tipo: Cualquier tipo

Valores: Cualquier valor

Valor predeterminado sin valor predeterminado

b

Cualquier objeto que se pueda comparar con a.

Tipo: Cualquier tipo

Valores: Cualquier valor

Valor predeterminado sin valor predeterminado

Indica

Tipo: booleano

Valores: True if a!=b, de lo contrario False

Ejemplo

```
from spike import ColorSensor
from spike.control import wait_until
from spike.operator import not_equal_to

color_sensor = ColorSensor('A')

wait_until(color_sensor.get_color, not_equal_to, None)
```

PrimeHub

La clase PrimeHub se divide en seis componentes, cada uno con varias funciones asiciadas.

Para poder utilizar el Hub, debes inicializarlo.

Ejemplo

```
from spike import PrimeHub


# Inicializa el Hub
hub = PrimeHub()
```

Constantes

Constantes

PrimeHub.left_button

El botón izquierdo del Hub.

Ejemplo:

```
from spike import PrimeHub

hub = PrimeHub()

hub.left_button.wait_until_pressed()
```

PrimeHub.right_button

El botón derecho del Hub.

Ejemplo:

```
from spike import PrimeHub

hub = PrimeHub()

hub.right_button.wait_until_pressed()
```

PrimeHub.speaket

El altavoz del Hub

Ejemplo:

```
from spike import PrimeHub

hub = PrimeHub()

hub.speaker.beep()
```

PrimeHub.light_matrix

La matriz de luces del Hub

Ejemplo:

```
from spike import PrimeHub

hub = PrimeHub()

hub.light_matrix.off()
```

PrimeHub.status_light

La luz de estado del ladrillo del botón central del Hub.

Ejemplo:

```
from spike import PrimeHub

hub = PrimeHub()

hub.status_light.on('blue')
```

PrimeHub.motion_sensor

El sensor de movimiento del Hub

Ejemplo:

```
from spike import PrimeHub

hub = PrimeHub()

yaw = hub.motion_sensor.get_yaw_angle()
```

PrimeHub.PORT_A
El puerto “A” del Hub

PrimeHub.PORT_B
El puerto “B” del Hub

PrimeHub.PORT_C
El puerto “C” del Hub

PrimeHub.PORT_D
El puerto “D” del Hub

PrimeHub.PORT_E
El puerto “E” del Hub

PrimeHub.PORT_F
El puerto “F” del Hub

Motores individuales

Para poder utilizar los motores, debes inicializar los dos motores.

Ejemplo:

```
from spike import Motor


# Inicializa el motor
motor = Motor('A')
```

A continuación te mostramos las funciones vinculadas al Motor mediano y al motor grande de SPIKE.

Acciones

run_to_position()

run_to_position(degrees, direction='shortest path', speed=None)

Acciona el motor hasta una posición absoluta.

Se ignora el signo de velocidad (valor absoluto) y el motor se desplazará siempre en la dirección especificada por el parámetro de dirección. Si la velocidad es superior a 100, se limitará a 100.

Parámetros

degrees

La posición objetivo.

Tipo: entero (positive o negativo, incluyendo 0)

Valores: 0 a 359

Valor predeterminado: sin valor predeterminado.

direction

La dirección que se ha de seguir para alcanzar la posición objetivo.

Tipo: cadena (texto)

Valores: La "ruta más corta" puede discurrir en cualquiera de ambos sentidos en función de la distancia más corta hacia el objetivo. -El "sentido horario" hará que el motor se accione en sentido horario hasta que alcance la posición objetivo. -El "sentido antihorario" hará que el motor se accione en sentido antihorario hasta que alcance la posición objetivo.

Valor predeterminado: sin valor predeterminado

speed

La velocidad del motor.

Tipo: entero (número entero positivo o negativo, incluyendo 0)

Valores: 0 a 100%

Valor predeterminado: Si no se especifica ningún valor, utilizará la velocidad predeterminada establecida por set_default_speed()

Errores

TypeError

Grados o velocidad no es un número entero la dirección no es una cadena.

ValueError

La dirección no es uno de los valores permitidos o los grados no están dentro del rango de 0 a 359 (ambos incluidos).

RuntimeError

El motor se ha desconectado del puerto.

Ejemplo

```
from spike import Motor

motor = Motor('A')


# Establece el motor en la posición 0 alineando
las señales

motor.run_to_position(0)
```

run_to_degrees_counted()

run_to_degrees_counted(degrees, speed=None)

Acciona el motor hasta que los grados contados sean iguales al valor especificado por el parámetro de grados.

Se ignorará el signo de velocidad y el motor se desplazará siempre en la dirección requerida para alcanzar los grados. Si la velocidad es superior a 100, se limitará a 100.

Parámetros

degrees

Los grados objetivo contados.

Tipo: entero (número entero positivo o negativo, incluyendo 0)

Valores: cualquier número.

Valor predeterminado: Sin valor predeterminado.

Errores

TypeError

Grados o la velocidad no es un número entero.

RunTimeError

El motor se ha desconectado del puerto

Ejemplo:

```
from spike import Motor
from spike.control import wait_for_seconds

motor = Motor('A')

for deg in range(0, 721, 90):
    motor.run_to_degrees_counted(deg)
    wait_for_seconds(1)
```

run_for_degrees()

run_for_degrees(degrees, speed=None)

Acciona el motor durante un número dado de grados.

Parámetros

degrees

El número de grados que debe accionarse el motor.

Tipo: entero (número entero positivo o negativo, incluyendo 0)

Valores: cualquier número.

Valor predeterminado: Sin valor predeterminado.

speed

La velocidad del motor.

Tipo: entero (número entero positivo o negativo, incluyendo 0)

Valores: -100 a 100%

Valor predeterminado: Si no se especifica ningún valor, utilizará la velocidad predeterminada establecida por set_default_speed().

Errores

TypeError

Grados o la velocidad no es un número entero.

RuntimeError

El motor se ha desconectado del puerto.

Ejemplo

```
from spike import Motor

motor = Motor('A')


# Acciona el motor 90 grados en sentido horario

motor.run_for_degrees(90)


# Acciona el motor 90 grados en sentido antihorario

motor.run_for_degrees(-90)


# Ejecuta el motor 360 grados en sentido horario a máxima
velocidad (100 %)

motor.run_for_degrees(360, 100)
```

run_for_rotations()

run_for_rotations(rotations, speed=None)

Acciona el motor durante un número especifico de rotaciones.

Parámetros

rotations

El número de rotaciones que debe accionarse el motor.

Tipo: float (número decimal)

Valores: Sin valor

Valor predeterminado: Sin valor predeterminado.

speed

La velocidad del motor

Tipo: entero (número entero positivo o negativo, incluyendo 0)

Valores: -100 a 100%

Valor predeterminado: Si no se especifica ningún valor, utilizará la velocidad predeterminada establecida por set_default_speed()

Errores

TypeError

Rotaciones no es un número o la velocidad no es un número entero.

RuntimeError

El motor se ha desconectado del puerto

Ejemplo

```
from spike import Motor

motor = Motor('A')


# Acciona el motor 90 grados en sentido horario:

motor.run_for_rotations(0.25)


# Acciona el motor 90 grados en sentido antihorario:

motor.run_for_rotations(-0.25)
```

run_for_seconds()

run_for_seconds(seconds, sepeed=None)

Acciona el motor durante un número especifico de segundos.

Parámetros

seconds

El número de segundos durante los cuales se debe accionar el motor.

Tipo: Float (número decimal)

Valores: Cualquier número.

Valor predeterminado: Sin valor predeterminado.

speed

La velocidad del motor.

Tipo: entero (número entero positivo o negativo, incluyendo 0)

Valores: -100 a 100%

Valor predeterminado: Si no se especifica ningún valor, utilizará la velocidad predeterminada establecida por set_default_speed().

Errores

TypeError

Los segundos no son un número o la velocidad no es un número entero.

RuntimeError

El motor se ha desconectado del puerto.

Ejemplo

```
from spike import Motor

motor = Motor('A')


# Acciona en sentido horario durante medio segundo al 75 %
de velocidad

motor.run_for_seconds(0.5, 75)


# Acciona en sentido horario durante 6 segundos al 30 % de
velocidad

motor.run_for_seconds(6, -30)
```

start()

start(speed_None)

Comienza a accionar el motor a una velocidad especificada.

El motor seguirá moviéndose a esta velocidad hasta que le indiques otro comando de motor o cuando tu programa finalice.

Parámetros

speed

La velocidad del motor.

Tipo: entero (número entero positivo o negativo, incluyendo 0)

Valores: -100 a 100%

Valor predeterminado: Si no se especifica ningún valor, utilizará la velocidad predeterminada establecida por set_default_speed().

Errores

TypeError

La velocidad no es un número entero.

RuntimeError

El motor se ha desconectado del puerto.

Ejemplo

```
from spike import Motor

motor = Motor('A')

motor.start()

# espera a algo...

motor.stop()
```

stop()

Detiene el motor.

Lo que el motor hace después de detenerse depende de la acción establecida en set_stop_action(). El valor por defecto de set_stop_action() es “desplazamiento en punto muerto”.

Errores

RuntimeError

El motor se ha desconectado del puerto.

Ejemplo

```
from spike import Motor

motor = Motor('A')

motor.start()

# espera a algo...

motor.stop()
```

start_at_power()

start_at_power(power)

Comienza a hacer girar el motor a un nivel de potencia especificado.

El motor seguirá moviéndose a este nivel de potencia hasta que le indiques otro comando de motor o cuando tu programa finalice.

Parámetros

power

Potencia del motor.

Tipo: Entero (número entero positivo o negativo, incluyendo 0)

Valores: -100 a 100%

Valor predeterminado: Sin valor predeterminado.

Errores

TypeError

Power no es un número entero.

RuntimeError

El motor se ha desconectado del puerto.

Medidas

get_speed()

Obtiene la velocidad del motor.

Indica

La velocidad actual del motor.

Tipo: Entero (número entero positivo o negativo, incluyendo 0)

Valores: -100% hasta 100%

Errores

RuntimeError

El motor se ha desconectado del puerto.

Get_position()

Obtiene la posición del motor. Esta consiste en el ángulo horario entre la señal en movimiento y la señal de punto cero del motor.

Indica

La posición del motor.

Tipo: Entero (númeo entero positivo o negativo, incluyendo 0)

Valores: 0 hasta 359 grados.

Errores

RuntimeError

El motor se ha desconectado del puerto

get_degrees_counted()

Obtiene los grados contados por el motor.

Indica

El número de grados contados.

Tipo: entero (número entero positivo o negativo, incluyendo 0)

Valores: Cualquier número.

Errores

RuntimeError

El motor se ha desconectado del puerto

get_default_speed()

Obtiene la velocidad del motor predeterminada actual.

Indica

La velocidad de motor predeterminada.

Tipo: entero (número entero positivo o negativo, incluyendo 0)

Valores: -100% hasta 100%

Eventos

was_interrupted()

Comprueba si se ha interrumpido el motor.

Indica

“Verdadero” si el motor se ha interrumpido desde la última vez que was_interrupted() se empleó, de lo contrario “falso”.

Tipo: Booleano

Valores: True or False

Errores

RuntimeError

El motor se ha desconectado del puerto.

Ejemplo

```
from spike import Motor

motor = Motor('A')

motor.run_for_rotations(2)
if motor.was_interrupted():
```

```
    # the motor did not complete two rotations
```

El motor no complete dos rotaciones

was_stalled()

Comprueba si se ha bloqueado el motor.

Indica

“Verdadero” si el motor se ha bloqueado desde la última vez que was_stalled() se empleó, de lo contrario “falso”.

Tipo: booleano

Valores: True or False

Errores

RuntimeError

El motor se ha desconectado del puerto.

Ejemplo

```
from spike import Motor

motor = Motor('A')

motor.set_stall_detection(True)
motor.run_for_rotations(2)
if motor.was_stalled():
    # the motor did not complete two rotations
```

El motor no completó dos rotaciones

Ajustes

set_degrees_counted()

set_degrees_counted(degrees_counted)

Establece el número de grados contados en el valor deseado.

Parámetros

degrees_counted

Valore en el que se debe establecer el número de grados contados.

Tipo: entero (número entero positivo o negativo, incluyendo 0)

Valores: cualquier número.

Valor predeterminado: Sin valor predeterminado

Errores

TypeError

Degrees_counte no es un número entero.

RuntimeError

El motor se ha desconectado del puerto.

set_default_speed()

set_default_speed(default_speed)

Establece la velocidad predeterminada del motor. Se utilizará esta velocidad cuando omitas el argumento de velocidad en uno de los métodos tales como run_for_degrees.

Establecer la velocidad predeterminada no afecta a ninguno de los motores que están actualmente en funcionamiento.

Solo tendrá efecto cuando se emplee otro método de motor después de este.

Si la velocidad predeterminada está fuera del intervalo permitido, la velocidad predeterminada se establecerá en -100 o 100 dependiendo de si el valor era positivo o negativo.

Parámetros

default_speed

El valor de velocidad predeterminado.

Tipo: entero (número entero positivo o negativo, incluyendo 0)

Valores: -100 a 100%

Valor: Sin valor

Valor predeterminado: Sin valor predeterminado.

Errores

TypeError

default_speed no es un número entero

set_stop_action()

set_stop_action(action)

Establece el comportamiento predeterminado cuando un motor se detiene.

Parámetros

action

El comportamiento de motor deseado cuando el motor se detiene.

Tipo: cadena (texto)

Valores: desplazamiento en punto muerto, freno, mantener posición

Valor predeterminado: desplazamiento en punto muerto

Errores

TypeError

action no es una cadena.

ValueError

action no es uno de los valores permitidos.

RuntimeError

El motor se ha desconectado del puerto

set_stall_detection()

set_stall_detection(stop_when_salled)

Activa o desactiva la detección de bloqueo.

La detección de bloqueo detecta cuándo se ha bloqueado un motor y no se puede mover. Si se ha activado la detección y se bloquea el motor, este se apagará tras dos segundos y el comando de motor actual se interrumpirá. Si la detección de bloqueo se ha desactivado, el motor seguirá intentando funcionar y los programas “se atascarán” hasta que el motor ya no esté bloqueado.

La detección de bloqueo está habilitada por defecto.

Parámetros

stop_when_stalled

Escoge “verdadero” para permitir la detección de bloqueo o “falso” para desactivarla.

Tipo: boolean

Valores: Verdadero o falso

Valor predeterminado: Verdadero

Errores

TypeError

Stop_when_stalled no es un booleano.

RuntimeError

El motor se ha desconectado del puerto

Ejemplo

```
from spike import Motor

motor = Motor('A')

motor.set_stall_detection(False)
motor.run_for_rotations(2)

# El programa nunca alcanzará este punto si se bloquea el motor
```

Altavoz

A continuación le mostramos todas las funciones vinculadas a los sonidos que emite el Hub de SPIKE Prime.

Acciones

beep()

beep(note=60, seconds=0.2)

Reproduce un pitido en el Hub,

Tu programa no continuará hasta que no se hayan pasado unos segundos.

Parámetro

note

El número de nota MIDI.

Tipo: float (número decimal)

Valores: 44 a 123 (60 en la nota Do central)

Valor predeterminado: 60 (nota Do central)

seconds

La duración del pitido en segundos.

Tipo: float (número decimal)

Valores: Cualquier valor

Valor predeterminado: 0,2 segundos.

Errores

TypeError

La nota no es un número entero o los segundos no son un número.

ValueError

Nota no está dentro del rango permitido de 44-123

Ejemplo

```
from spike import PrimeHub

hub = PrimeHub()


# ¡Bip bip bip!

hub.speaker.beep(60, 0.5)
hub.speaker.beep(67, 0.5)
hub.speaker.beep(60, 0.5)
```

start_beep()

start_beep(note=60)

Comienza a reproducir un pitido.

El pitido se reproducirá indefinidamente hasta que se emplee stop() o hasta que se emplee otro método de pitido.

Parámetro

note

El número denota MIDI.

Tipo: float (número decimal)

Valores: 44-123 (60 es la nota Do central)

Errores

TypeError

La nota no es un número entero.

ValueError

nota no está dentro del rango permitido de 44-123

Ejemplo

```
from spike import PrimeHub

hub = PrimeHub()

hub.speaker.start_beep()

# haz algo

hub.speaker.stop()
```

stop()

Detiene cualquier sonido que se esté reproduciendo.

Ejemplo

```
from spike import PrimeHub

hub = PrimeHub()

hub.speaker.start_beep()

# haz algo

hub.speaker.stop()
```

Medidas

get_volumen()

Obtiene el valor del volumen del altavoz.

Esto solo obtiene el volumen del Hub, y no el de la app de programación.

Indica

El volumen actual

Tipo: entero (número entero positivo o negativo, incluyendo 0)

Valores: 0 hasta 100%

Ejemplo

```
from spike import PrimeHub

hub = PrimeHub()


# Aumenta el volumen del altavoz del Hub en un
10 %

hub.speaker.set_volume(hub.speaker.get_volume()
+ 10)
```

Ajustes

Set_volumen()

set_volume(volume)

Ajusta el volumen del altavoz.

Si el volumen asignado está fura del intervalo, se empleará en su lugar el volumen más cercano (0 o 100). Esto solo ajusta el volumen del Hub, y no el de la app.

Parámetros

volume

El nuevo porcentaje de volumen.

Tipo: entero (número entero positivo o negativo, incluyendo 0)

Valores: 0 a 100%

Valor predeterminado: 100%

Errores

TypeError

El volumen no es un número entero.

Ejemplo

```
from spike import PrimeHub

hub = PrimeHub()


# Establece el volumen del altavoz del Hub en
el 50 %

hub.speaker.set_volume(50)
```

Luz de estado

A continuación te mostramos todas las funciones vincuadas a la luz de estado del ladrillo programable dl Hub de SPIKE Prime.

Acciones

on()

on(color='white')

Establece el color de la luz.

Parámetros

color

Ilumina la luz de estado del ladrillo del Hub con el color especificado.

Tipo: Cadena (texto)

Valores: azul cielo, negro,'azul','cian','verde','naranja','rosa','viioleta','amarillo','blanco'

Valor predeterminado: blanco

Errores

TypeError

El color no es un string

ValueError

El color o es uno de los valores permitidos

Ejemplo:

```
from spike import PrimeHub

hub = PrimeHub()

hub.status_light.on('blue')
```

off()

Apaga la luz

Ejemplo:

```
from spike import PrimeHub

hub = PrimeHub()

hub.status_light.off()
```

Funciones de espera

wait_for_seconds()

wait_for_seconds(seconds)

Espera un número específico de segundos antes de proseguir con el programa.

Parámetro

seconds

El tiempo de espera en segundos.

Tipo: float (número decimal)

Valores: cualquier valor.

Valor predeterminado: Sin valor predeterminado

Errores

TypeError

Los segundos no son un número.

ValueError

Segundos no es al menos 0.

Ejemplo

```
from spike.control import
wait_for_seconds


# Espera 3 segundos (pausa el
flujo del programa)

wait_for_seconds(3)
```

wait_until()

wait_until(get_value_function, operator_function=<function equal_to>, target_value=True)

Espera hasta que la condición sea “verdadero” antes de continuar con el programa.

Parámetros

Get_value_function

Tipo: Función invocable

Valores: Función que indica el valor actual que se va a comparar con el valor objetivo.

Valor predeterminado: Sin valor predeterminado.

Operator_function

Tipo: Función invocable

Valores: Una función que compara dos argumentos. El primer argumento será el resultado de get_value_function() y el segundo argumento será target_value. La función comparará ambos valores e indicará el resultado.

Valor predeterminado: Sin valor predeterminado.

Errores

TypeError

Get_value_function o operator_function no es invocable o operator_function no compara dos argumentos.

Ejemplo

```
from spike import ColorSensor
from spike.control import wait_until
from spike.operator import equal_to

color_sensor = ColorSensor('A')


# Espera a que el Sensor de color detecte el rojo

wait_until(color_sensor.get_color, equal_to, 'red')
```

Ejemplo

```
from spike import ColorSensor, Motor
from spike.control import wait_until

color_sensor = ColorSensor('A')
motor = Motor('B')

def red_or_position():
    return color_sensor.get_color() == 'red' or
motor.get_position() > 90

wait_until(red_or_position)
```

Cronómetro

Para poder utilizar el cronómetro, debes inicializarlo.

Ejemplo

```
from spike.control import Timer


# Inicializa el cronómetro.
timer = Timer()
```

A continuación te mostramos todas las funciones vinculadas al cronómetro.

reset()

Pone el cronómetro a 0.

Ejemplo

```
from spike.control import Timer

timer = Timer()
# Después de cierto tiempo...
timer.reset()
```

now()

Obtiene el tiempo de cronómetro para “ahora mismo”

Indica

El tiempo actual en segundos.

Tipo: Entero (número entero positivo o negativo, incluyendo 0)

Valores: Un valor mayor que 0.

Ejemplo

```
from spike.control import Timer

timer = Timer()

while True:
    # Si han pasado más de 5 segundos desde que se inició
el cronómetro
    if timer.now() > 5:
        # then break out of the while loop
        break
```

Contenido