Arduino Alvik API

Para acceder a las funciones de Arduino Alvik API tenemos que ejecutar las instrucciones:

```
alvik = ArduinoAlvik()
alvik.begin()
```

Entonces ya podemos usar las siguientes: (extraido de https://docs.arduino.cc/tutorials/alvik/api-overview/)

> Luego veremos en el apartado de programación del Arduino Alvik con código Arduino IDE que utilizaremos una biblioteca #include "Arduino_Alvik.h" que importa prácticamente las mismas funciones, ver https://libros.catedu.es/books/arduino-alvik/page/programas-arduino-ide-sin-iot

FUNCION con sus Inputs	**Outputs**
stop()	para todas las funciones Alvik
is_on()	true si esta encendido false si esta apagado
is_target_reached()	true si ha enviado M o R en el mensaje
get_ack()	last_ack: el valor del último mensaje
stop()	para todas las funciones Alvik
get_orientation()	r: valor de balanceo p: valor de cabeceo y: valor de guiñada
get_accelerations() ver uso en https://libros.catedu.es/books/arduino-alvik/page/programas-de-ejemplo	ax ay az
get_gyros() ver uso en https://libros.catedu.es/books/arduino-alvik/page/programas-de-ejemplo	gx by gz

get_imu()	las 6 anteriores
get_line_sensors()	left center right
brake()	Frena el robot
get_battery_charge()	battery_soc: el % de la batería
get_touch_any()	touch_any es true si se ha apretado cualquier botón
get_touch_ok() get_touch_cancel() get_touch_center() get_touch_up() get_touch_left() get_touch_down() get_touch_right()	touch_ok es true si se ha apretado ok etc... ver ejemplos en https://libros.catedu.es/books/arduino-alvik/page/robotica-para-infantil y en https://libros.catedu.es/books/arduino-alvik/page/mensajes-a-telegram
get_color_raw() get_color_label()	color
get_version() print_status()	versión del firmware para actualizarlo ver https://docs.arduino.cc/tutorials/alvik/user-manual/#how-to-upload-firmware
set_behaviour(behaviour: int)	
rotate(angle: float, unit: str = 'deg', blocking: bool = True)	
move(distance: float, unit: str = 'cm', blocking: bool = True)	
get_wheels_speed(unit: str = 'rpm')	left_wheel_speed: the speed value right_wheel_speed: the speed value
set_wheels_speed(left_speed: float, right_speed: float, unit: str = 'rpm')	
set_wheels_position(left_angle: float, right_angle: float, unit: str = 'deg')	
get_wheels_position(unit: str = 'deg')	angular_velocity
drive(linear_velocity: float, angular_velocity: float, linear_unit: str = 'cm/s',angular_unit: str = 'deg/s')	
get_drive_speed(linear_unit: str = 'cm/s', angular_unit: str = 'deg/s')	linear_velocity: speed of the robot. angular_velocity: speed of the wheels.
reset_pose(x: float, y: float, theta: float, distance_unit: str = 'cm', angle_unit: str = 'deg')	

get_pose(distance_unit: str = 'cm', angle_unit: str = 'deg')	x y theta
set_servo_positions(a_position: int, b_position: int)	
set_builtin_led(value: bool)	
set_illuminator(value: bool)	
color_calibration(background: str = 'white')	
rgb2hsv(r: float, g: float, b: float)	h: hue value s: saturation value v: brightness value
get_color(color_format: str = 'rgb')	r or h g or s b or v
hsv2label(h, s, v)	color label: like "BLACK" or "GREEN", if possible, otherwise return "UNDEFINED"
get_distance(unit: str = 'cm')	lee la distancia del sensor TOF: ver ejemplo en https://libros.catedu.es/books/arduino-alvik/page/evita-obstaculos left_tof: 45° to the left object distance center_left_tof: 22° to the left object distance center_tof: center object distance center_right_tof: 22° to the right object distance right_tof: 45° to the right object distance
get_distance_top(unit: str = 'cm')	top_tof: 45° to the top object distance
get_distance_bottom(unit: str = 'cm')	bottom_tof: 45° to the bottom object distance
on_touch_ok_pressed(callback: callable, args: tuple = ()) on_touch_cancel_pressed(callback: callable, args: tuple = ()) on_touch_center_pressed(callback: callable, args: tuple = ()) on_touch_up_pressed(callback: callable, args: tuple = ()) on_touch_left_pressed(callback: callable, args: tuple = ()) on_touch_down_pressed(callback: callable, args: tuple = ()) on_touch_right_pressed(callback: callable, args: tuple = ())	He intentado hacer programas con estas instrucciones, pero una vez pulsado la tecla, sigue llamando a callback continuamente No veo su utilidad teniendo get_touch

Unidades

- m: centimeters
  mm: millimeters
  m: meters
  inch: inch, 2.54 cm

in: inch, 2.54 cm

- deg: degrees, example: 1.0 as reference for the other unit. 1 degree is 1/360 of a circle.
  rad: radiant, example: 1 radiant is 180/pi deg.
  rev: revolution, example: 1 rev is 360 deg.
  revolution: same as rev
  perc: percentage, example 1 perc is 3.6 deg.
  %: same as perc

- 'cm/s': centimeters per second
  'mm/s': millimeters per second
  'm/s': meters per second
  'inch/s': inch per second
  'in/s': inch per second

- 'rpm': revolutions per minute, example: 1.0 as reference for the other unit.
  'deg/s': degrees per second, example: 1.0 deg/s is 60.0 deg/min that is 1/6 rpm.
  'rad/s': radiant per second, example: 1.0 rad/s is 60.0 rad/min that is 9.55 rpm.
  'rev/s': revolution per second, example: 1.0 rev/s is 60.0 rev/min that is 60.0 rpm.

> ¿Qué es eso de **bloking**?
> Por ejemplo en rotate(angle: float, unit: str = 'deg', blocking: bool = True)
>
> Si es true, todos los eventos no influyen, es decir el microprocesador esta centrado en esa instrucción
> Si es falso, el microprocesador es libre de hacer otra cosa a la vez
>
> Utiliza true si quieres precisión o no quieres que nada interaccione con la acción que estas ejecutando

---

Revision #11
Created 15 June 2024 08:10:06 by Javier Quintana
Updated 3 January 2025 08:56:38 by Javier Quintana