小龟机器人开发手册

四轮运动接口

概要

小龟小车可以通过运动指令来控制，进行各种运动，如前进、后退、左转、右转等等。

一、前进(go)

语法：

car.go(power:int, keep:float)

car.go(power:int)

car.go()

说明：让小车前进。

参数：

power　小车动力百分比，0-100，100 代表 100%动力，缺省 80（代表 80%动力）。

keep　　运动持续时间，单位秒。

范例：

```
#小车以 100%动力前进 5 秒
car.go(100,5)
#小车以 30%动力一直前进
car.go(30)
```

二、后退(back)

语法：

car.back(power:int, keep:float)

car.back(power:int)

car.back()

说明：让小车后退。

参数：

power　小车动力百分比，0-100，100 代表 100%动力，缺省 80（代表 80%动力）。

keep　　运动持续时间，单位秒。

范例：

```
#小车以 100%动力后退 1 秒
car.back(100,1)
```

三、左转(left)

语法：

car.left(power:int, keep:float)

car.left(power:int)

car.left()

说明：让小车左转。

参数：

power　　小车动力百分比，0-100，100 代表 100%动力，缺省 80（代表 80%动力）。

keep　　运动持续时间，单位秒。

范例：

```
#小车以 70%动力左转 3 秒
car.left(70,3)
```

四、右转(right)

语法：

car.right(power:int, keep:float)

car.right(power:int)

car.right()

说明：

让小车右转。

参数：

power　　小车前进的动力百分比，0-100，100 代表 100%动力，缺省 80（代表 80%动力）。

keep　　运动持续时间，单位秒。

范例：

```
#小车以 57%动力一直右转
car.right(57)
```

五、刹车(stop)

语法：

car.stop()

说明：让小车刹车停止运动。

参数：无

范例：

```
#刹车
car.stop()
```

六、指定每一个车轮电机运动(motor)

语法：

car.motor(L1:int, R1:int, L2:int, R2:int)

car.motor(L1:int, R1:int, L2:int)

car.motor(L1:int, R1:int)

car.motor(L1:int)

说明：指定小车的每一个车轮电机的具体运动。

参数：

L1　　L1 轮(左前轮)的运动动力，-100 ~ 100，代表反转 100%到正转 100%。

R1　　R1 轮(右前轮)的运动动力，-100 ~ 100，代表反转 100%到正转 100%。

L2　　L2 轮(左后轮)的运动动力，-100 ~ 100，代表反转 100%到正转 100%。

R2　　R2 轮(右后轮)的运动动力，-100 ~ 100，代表反转 100%到正转 100%。

范例：

```
#仅左前轮以 100%动力向前正转
car.motor(100)
#以左前轮反向 50%，左后轮反向 50%，右前轮正向 70%，右后轮正向 70% 执行转弯
```

```
car.motor(-50, 70, -50, 70)
```

七、问候动作(hello)

语法：

car.driver.hello()

car.driver.hello(action)

说明：执行小车问候动作。

参数：

action 小车问候动作代号，1 代表抖三抖的 Hello 动作，2 代表抖一下的 Hi 动作，缺省默认为 1。

范例：

```
#执行抖三抖的 Hello 动作
car.driver.hello()
#执行抖一下的 Hi 动作
car.driver.hello(2)
```

舵机接口

概要

小龟主控板设计了一套基于舵机并行的运动模型，也就是同时提交两行舵机指令，会同步执行，而不是象传统的串行代码那样，先等待第一行舵机指令结束，再执行第二行舵机指令。基于这个接口，可以快速实现多个舵机的并行联动。

一、管脚集

在机器人结构中，通常每个结构单元都会由多个舵机组成，我们将这些多个舵机对应的管脚集，称为管脚集。

如：机械臂是由 S1、S2、S3 三个舵机组成，那么机械臂的管脚集是 [S1, S2, S3]；

如：机器狗尾巴是由 A1、S7 两个舵机组成，那么尾巴的管脚集是 [A1, S7]；

二、动作节点集

我们知道一个动作（比如：抬腿）可以分解成多个节点，比如先抬膝盖到指定高度，然后再抬小腿到指定高度两个阶段组成。

我们将这两个阶段称为两个节点，一组节点组成了一个完整的动作

单独每个节点的格式如：

[节点耗时，节点停留，舵机 1 的目标角度，舵机 2 的目标角度, ……]

分别意义如下：

节点耗时：代表期望用多少秒，将所有舵机旋转到目标角度。如果本节点内有多个舵机，则多个舵机都会消耗相同的时间到达指定角度；

节点停留：代表所有舵机到达本节点目标角度后，保持现有角度停留多少秒；

舵机 N 的目标角度：和管脚集对应的每个舵机期望旋转到的目标角度；

节点集则是由多个单一节点组合而成的数组集合：

[[节点 1 耗时，节点 1 停留，舵机 1 的目标角度 1，舵机 2 的目标角度 1, ……],

[节点 2 耗时，节点 2 停留，舵机 1 的目标角度 2，舵机 2 的目标角度 2, ……], ….

三、单舵机单角度运动

语法：

car.servo(管脚，目标角度，旋转耗时)

说明：

旋转指定管脚上的舵机到目标角度。

参数：

管脚：管脚的标准名称

目标角度：单位度，期望舵机最终到达的角度

旋转耗时：单位秒，期望舵机用多少时间完成从当前角度到达期望的目标角度的旋转工作。

范例：

```
#用 5 秒钟时间，将舵机 A1 从当前角度位置旋转到 90 度位置。
car.servo(A1, 90, 5)
#用 1.5 秒钟时间，将舵机 S1 从当前角度位置旋转到 0 度位置。
car.servo(S1, 0, 1.5)
```

单舵机连片运动

语法：

car.servo(管脚，动作节点集，重复动作时长)

car.servo(管脚，动作节点集)

说明：旋转指定管脚上的舵机到目标角度。

参数：

管脚：管脚的标准名称

动作节点集：参考本文动作节点介绍篇章

重复动作时长：单位秒，重复整个动作多长时间后才结束。（本参数缺省的场合，为动作仅执行一次）

范例：

```
#按这个次序旋转 A1 管脚上的舵机：
#1、先用 3 秒钟时间旋转到 0 度，然后停留 1 秒钟；
#2、接着用 1 秒钟时间旋转到 180 度，不做停留；
#3、最后用 3 秒钟时间旋转舵机到 95 度。
car.servo(A1, [
[3,1,0],
[1,0,180],
[3,0,95],
])
```

四、多舵机连片运动

语法：

car.servo(管脚集，动作节点集，重复动作时长)

car.servo(管脚集，动作节点集)

说明：旋转指定管脚上的舵机到目标角度。

参数：

管脚集：管脚集合，参考本文管脚集定义介绍篇章。

动作节点集：参考本文动作节点介绍篇章。

重复动作时长：单位秒，重复整个动作多长时间后才结束。（本参数缺省的场合，为动作仅执行一次）

范例：

```
#按下面次序依次同步旋转 S1、S2、S3 三个舵机：
#步骤 1：用 3.5 秒钟时间，同时将 S1 旋转到 0 度、S2 旋转到 0 度、S3 旋转到 0 度，旋转完成后，并停留 1 秒钟；
#步骤 2：用 2 秒钟时间，同时将 S1 旋转到 180 度、S2 旋转到 120 度、S3 旋转到 90 度，旋转完成后，并停留 1.5 秒钟；
#步骤 3：用 5 秒钟时间，同时将 S1 旋转到 30 度、S2 旋转到 90 度、S3 旋转到 180 度，旋转完成后，并停留 1 秒钟；
car.servo([S1, S2, S3], [
[3.5,  1,   0,   0,   0],
[2,  1.5, 180, 120,  90],
[5,    1,  30,  90, 180]
])
```

板载蜂鸣器

概要

小龟小车主控板内置了一枚蜂鸣器，可以编程发出各种频率的声音。

一、播放音乐(music)

语法：

car.buzzer.music(music:str)

说明：

播放音乐简谱。

参数：

返回值　　无。

范例：

```
#播放音乐  1 2 3
car.buzzer.music("1 2 3")
```

二、播放火警(fire)

范例：

```
#播放火警警报
car.buzzer.fire()
```

三、播放频率(freq)

语法：

car.buzzer.freq(freq:int, keep:int)

说明：

播放指定频率的声音，指定时间。

参数：

freq　　声音频率

keep　　播放时长，0 为一直播放

范例：

```
#播放 2000Hz 声音 5 秒种
car.buzzer.freq(2000, 5)
```

四、播放开机音(hello)

范例：

```
#播放小车开机音乐
car.buzzer.hello()
```

五、停止播放(close)

范例：

```
#停止播放音乐
car.buzzer.close()
```

板载 LED 灯

概要

小龟小车主控板前端有两枚 RGB 三色 LED 灯，可以通过编程控制这两盏灯的亮灭和色彩，两盏灯不可以独立控制其中一盏。

一、打开 LED 灯(on)

语法：

```
car.led.on()
car.led.on(RGB)
```

说明：

打开 LED 灯。

参数：

RGB　　显示的 RGB 颜色值，缺省为最亮颜色。

范例：

```
#打开 LED 灯，并显示红色
```

```
car.led.on(0xFF0000)

#打开 LED 灯，并显示绿色

car.led.on(0x00FF00)
```

二、关闭 LED 灯(off)

语法：

```
car.led.off()
```

说明：

关闭 LED 灯。

参数：

无

范例：

```
#关闭 LED 灯

car.led.off()
```

板载触摸按钮

小龟小车主控板带有两个触摸按钮 A 和 B，位于主控板背面。

一、按钮状态(is_pressed)

语法：

```
car.touch.is_pressed(pin)->bool
```

说明：

查询指定的触摸按钮是否被按住。

参数：

pin　　按钮名称。

返回值　True：被按住状态

False：没有按住状态

范例：

```
#查询触摸按钮 A 的状态
car.touch.is_pressed(A)
#也可以使用管脚连写模式
A.is_pressed()
```

二、按钮触摸值(val)

语法：

car.touch.val(pin)->int

说明：

查询指定的触摸按钮当前的触摸值。

参数：

pin　　按钮名称。

返回值　　触摸值，没有被触摸情况下的值比触摸情况下大。

范例：

```
#查询触摸按键 A 的触摸值
car.touch.val(A)
```

HCSR-04 超声波距离传感器

概要

小龟小车支持最常见的 HCSR-04 超声波距离探测传感器。

一、探测距离(hcsr04)

语法：

car.hcsr04(pin_trig, pin_echo)

说明：

对接在 pin_trig 和 pin_echo 管脚上的超声波传感器执行测量距离任务。如果测试结果是 0 的话，请检查管脚连接是否正确。

参数：

pin_trig　　对应超声波传感器的 Trig 管脚。

pin_echo　　对应超声波传感器的 Echo 管脚。

返回值　　测量的距离值，单位厘米(cm)

范例：

```
#测量 Trig 接在 A1，Echo 接在 A2 管脚的超声波前方障碍物的距离
distance = car.hcsr04(A1, A2)
print(distance)
```

WS2812 全彩灯

概要

小龟小车可以编程控制 WS2812 全彩灯、灯带与矩阵。

一、点亮一颗 WS2812 全彩灯

语法：

car.ws2812.write(pin, colors)

说明：

点亮接在管脚 pin 上的 WS2812 全彩灯。

参数：

pin　　连接 WS2812 的 DI 管脚。

colors　RGB 色彩代码

返回值　　无

范例：

```
#点亮 A1 管脚的 WS2812 灯并显示颜色为绿色
car.ws2812.write(A1, [0x00FF00])
```

二、点亮多颗 WS2812 全彩灯

语法：

car.ws2812.write(pin, [colors，……])

说明：

点亮接在管脚 pin 上的多颗 WS2812 全彩灯。

参数：

pin　　对应第一颗 WS2812 的 DI 管脚。

返回值　　无

范例：

```
#点亮 A1 管脚的 3 颗 WS2812 灯，并依次显示颜色红、绿、蓝
car.ws2812.write(A1, [0xFF0000，0x00FF00，0x0000FF])
```

三、关闭 WS2812 全彩灯

语法：

car.ws2812.clear(pin)

说明：

关闭接在管脚 pin 上的 WS2812 全彩灯。

参数：

pin　　连接第一颗 WS2812 的 DI 管脚。

返回值　　无

范例：

```
#关闭 A1 管脚的 WS2812 灯
car.ws2812.clear(A1)
```

四、查询灯珠数目(count)

语法：

car.ws2812.count(pin)

说明：

查询当前已经输出的灯珠数目。

参数：

pin　　炫彩灯连接的管脚。

范例：

```
#查询当前 A1 管脚已经输出的灯珠数目
car.ws2812.count(A1)
```

管脚接口

概要

小龟主控板拥有一套非常简洁的管脚端口操作语法。

一、查询触摸状态(is_pressed)

语法：

Pin.is_pressed()

car.touch.is_pressed(pin)

说明：

查询指定的管脚是否处于被触摸中。

参数：

pin　　管脚名称

返回值　　返回是否被触摸

范例：

```
#查询按键 A 是否被按下
A.is_pressed()
#另一种写法
```

```
car.touch.is_pressed(A)
#查看返回结果，按下返回 True，未按下返回 False
print(A.is_pressed())
```

二、GPIO 功能(gpio)

语法：

```
car.gpio(pin)          #读管脚
car.gpio(pin, val)     #写管脚
```

说明：

设置或者读取指定管脚的状态。

参数：

pin　　管脚名称

val　　设置管脚状态 0-设置管脚为低位，1-设置管脚为高位，缺省为读取管脚当前状态。

返回值　返回管脚状态

范例：

```
#读取 A1 管脚的状态
car.gpio(A1)
#设置 A1 管脚低电位
car.gpio(A1,0)
```

三、打开管脚(on)

语法：

```
car.on(pin)
```

Pin.on()

说明：

设置指定管脚输出高位信号。

参数：

pin　　管脚名称

范例：

```
#打开 D3 管脚
car.on(D3)
#另一种写法
D3.on()
```

四、关闭管脚(off)

语法：

car.off(pin)

Pin.off()

说明：

设置指定管脚输出低位信号。

参数：

pin　　管脚名称

范例：

```
#关闭管脚 SCL
car.off(SCL)
#另一种写法
```

```
SCL.off()
```

五、查询管脚状态(val)

语法：

car.val(pin)

Pin.val()

说明：

查询管脚当前状态。

参数：

pin　　管脚名称

返回值　　返回管脚状态

范例：

```
#查询管脚 SDA 的状态
car.val(SDA)
#另一种写法
SDA.val()
#查看返回值
print(car.val(SDA))
```

六、发送脉冲(pulse)

语法：

pin.pulse(level, keep)

car.pin.pulse(level, keep)

说明：

输出指定高低位指定时长的脉冲信号。

参数：

pin　　管脚名称

level　　脉冲信号状态，0-低位， 1-高位

keep　　脉冲持续时长

范例：

```
#在 A1 管脚上发送 10 毫秒的高位脉冲信息
car.A1.pulse(1, 0.01)
#另一种写法
A1.pulse(1, 0.01)
```

七、操控舵机(servo)

语法：

car.servo(pin, angle，耗时)

car.servo(pin,angle)

pin.servo(angle)

说明：

设置接在指定管脚上的舵机角度。

参数：

pin　　管脚名称

angle　　舵机角度 0 - 180，代表 0 到 180 度。

耗时　　期望用多少秒实现舵机旋转到指定角度。

返回值　　返回管脚状态

范例：

```
#控制接在 A1 管脚上的舵机旋转到 90 度
car.servo(A1, 90)
#另一种写法
A1.servo(90)
#控制接在 A1 管脚上的舵机用 2 秒的时间旋转到 90 度(标点符号使用英文输入法)
car.servo(A1,90,2)
```

八、设置 PWM 信号频率(pwm_freq)

语法：

car.pwm_freq(pin, freq)

说明：

设置整块主控板的 PWM 输出频率。

参数：

pin	管脚名称
freq	PWM 输出频率

范例：

```
#设置主控板 PWM 信号频率为 100Hz
car.pwm_freq(A1, 100)
```

九、设置 PWM 信号占空比(pwm_duty)

语法：

car.pwm_duty(pin, duty)

说明：

设置指定管脚输出的 PWM 信号的占空比。

参数：

pin　　管脚名称

duty　　PWM 信号的占空比， 0 - 100，代表 0%到 100%。

范例：

```
#设置 A1 管脚发送 50%占空比的 PWM 信号
car.pwm_duty(A1, 50)
```

ADC 接口

概要

小龟小车主控板以下管脚支持 ADC 功能。

正面：D1、D2

背面：

另外小龟小车也使用了 ADC 功能对锂电池正极进行电压测量。

一、测量电池电压(battery)

语法：

car.adc.battery()->int

说明：

测量接在小龟小车主控板锂电池接口上的电池电压。该测量电压值可以用来判断剩余电量。

参数：

返回值　电池电压，单位毫伏(mV)。

范例：

```
#测量电池电压
car.adc.battery()
#查看返回值，3920 代表电池电压 3920mV，也就是 3.92V
print(car.adc.battery())
```

二、测量管脚电压(val)

语法：

car.adc.val(pin)->float

说明：

测量接在对应管脚上的电压，该功能可以用在诸如电压型的操作杆。

参数：

pin　　管脚名称，请参考页头的管脚支持列表

返回值　电池电压，单位福特(V)。

范例：

```
#测量管脚电压
car.adc.val(D1)
#查看返回值， 3.92 代表接在管脚 D1 上的传感器管脚电压是 3.92V
print(car.adc.val(D1))
```

DAC 接口

概要

小龟小车主控板以下管脚支持 DAC 功能

正面：A1、A2

输出指定电压(dac)

语法：

car.dac(voltage:float)

说明：

在小龟小车主控板指定管脚上输出指定电压。

参数：

voltage　　希望输出电压大小， 单位福特(V)。

范例：

```
#在 A1 管脚输出 2.2V 电压

car.dac(A1, 2.2)
```

I2C 用户总线接口

概要

小龟小车支持一路用户 I2C 总线，默认管脚是主板正面 D3 管脚边上的 SCL、SDA。

一、打开总线(open)

语法：

car.i2c.open()

car.i2c.open(speed)

说明：

以指定的参数打开用户 I2C 总线。

参数：

speed　　指定 I2C 用户总线的通讯速率，默认 100kbps。

返回值　　打开是否成功。

范例：

```
#以 400kbps 速率打开默认用户总线
car.i2c.open(400000)
```

二、扫描总线(scan)

语法：

car.i2c.scan()

说明：

扫描当前用户总线，并返回发现的传感器地址。

参数：

返回值　　发现的传感器地址列表

范例：

```
#扫描用户 I2C 总线
car.i2c.scan()
#返回  [0, 11, 59]
```

三、读取数据(read)

语法：

car.i2c.read(addr:int, reg:int)

car.i2c.read(addr:int, reg:int, count:int)

说明：

从指定地址的传感器的指定内部数据寄存器位置读入指定数量的数据。

参数：

addr　　需要读取的传感器 I2C 地址。

reg　　需要读取的传感器内部数据寄存器地址。

count　　需要读取的数据个数，默认 1 字节。

返回值　　读到的数据

范例：

```
#读取地址为 109 的传感器的内部数据寄存器位置为 11 处 1 个字节数据
car.i2c.read(109, 11, 1)
```

四、不指定寄存器读取数据(read_raw)

语法：

car.i2c.read_raw(addr:int)

car.i2c.read_raw(addr:int, count:int)

说明：

从指定地址的传感器的读入指定数量的数据。

参数：

addr　　需要读取的传感器 I2C 地址。

count　　需要读取的数据个数，默认 1 字节。

返回值　　读到的数据。

范例：

```
#读取地址为 109 的传感器的 1 个字节数据
car.i2c.read(109, 1)
```

五、写入数据(write)

语法：

car.i2c.write(addr:int, reg:int)

car.i2c.write(add:int, reg:int, count:int)

说明：

向指定地址的传感器内部指定位置的数据寄存器写入指定长度的数据。

参数：

addr　　需要写入的传感器 I2C 地址。

reg　　需要写入的传感器内部数据寄存器地址。

count　　需要写入的数据个数，默认 1 字节。

返回值　　处理是否成功。

范例：

```
#向地址为 105 的传感器的第 57 个内部数据寄存器写入 1 个字节的数据 19
car.i2c.write(105, 57, [19])
```

六、不指定寄存器写入数据(write_raw)

语法：

car.i2c.write(addr:int, reg:int)

car.i2c.write(add:int, reg:int, count:int)

说明：

向指定地址的传感器写入指定长度的数据。

参数：

addr　　需要写入的传感器 I2C 地址。

reg　　需要写入的传感器内部数据寄存器地址。

count　　需要写入的数据个数，默认 1 字节。

返回值　处理是否成功。

范例：

```
#向地址为 105 的传感器写入 2 个字节的数据 19 和 36

car.i2c.write(105, [19, 36])
```

I2C 系统总线接口

概要

小龟小车主控板设计了一路用于主控板板载芯片通讯的 I2C 总线，这路 I2C 系统总线会在主控板开机后，自动加载。

一、读取数据(read)

语法：

```
car.i2csys.read(addr:int, reg:int)

car.i2csys.read(addr:int, reg:int, count:int)
```

说明：

从指定地址的传感器的指定内部数据寄存器位置读入指定数量的数据。

参数：

addr　　需要读取的传感器 I2C 地址。

reg　　需要读取的传感器内部数据寄存器地址。

count　　需要读取的数据个数，默认 1 字节。

返回值　读到的数据。

二、不指定寄存器读取数据(read_raw)

语法：

car.i2csys.read_raw(addr:int)

car.i2csys.read_raw(addr:int, count:int)

说明：

从指定地址的传感器的读入指定数量的数据。

参数：

addr　　需要读取的传感器 I2C 地址。

count　　需要读取的数据个数，默认 1 字节。

返回值　读到的数据。

三、扫描总线(scan)

语法：

car.i2csys.scan()

说明：

扫描当前用户总线，并返回发现的传感器地址。大部分传感器是一个传感器一个地址，但也有不少传感器会存在多个地址

参数：

返回值　　发现的传感器地址列表

四、写入数据(write)

语法：

car.i2csys.write(addr:int, reg:int)

car.i2csys.write(add:int, reg:int, count:int)

说明：

向指定地址的传感器内部指定位置的数据寄存器写入指定长度的数据。

参数：

addr　　需要写入的传感器 I2C 地址。

reg　　需要写入的传感器内部数据寄存器地址。

count　需要写入的数据个数，默认 1 字节。

返回值　处理是否成功

五、不指定寄存器写入数据(write_raw)

语法：

car.i2csys.write(addr:int, reg:int)

car.i2csys.write(add:int, reg:int, count:int)

说明：

向指定地址的传感器写入指定长度的数据。

参数：

addr　　需要写入的传感器 I2C 地址。

reg　　需要写入的传感器内部数据寄存器地址。

count　需要写入的数据个数，默认 1 字节。

返回值　处理是否成功

系统屏幕

概要

小龟小车主控板支持一枚 SSD1306 芯片做为系统屏幕(I2C 协议),使用的时候将屏幕模块的四根管脚插入系统 I2C 总线对应的管脚即可。在执行显示输出前，如不使用 car.screen.open 接口配置屏幕的特征的话，小车系统会议默认的参数初始化屏幕。

一、打开屏幕(open)

语法:

car.screen.open(width:int, height:int, i2c_addr:int)->bool

car.screen.open()

说明:

初始化 SSD1306 屏幕，如不进行初始化，在首次使用相关的屏幕输出接口的时候，会以默认配置初始化屏幕。默认配置：128 像素宽、64 像素高、I2C 地址 60。

参数:

width　　屏幕宽度，单位像素。如不填，则默认 128 像素。

height　　屏幕高度，单位像素。如不填，则默认 64 像素。

i2c_addr　屏幕模块的 I2C 地址，如不填，则默认 60。

返回值　　处理是否成功。

范例:

```
#打开屏幕，并向系统登记屏幕大小为 128*64，I2C 地址 60
car.screen.open(128, 64, 60)
```

二、调节屏幕亮度(brightness)

语法：

car.screen.brightness(level:int)->bool

说明：

调节屏幕亮度。

参数：

level　　亮度级别 0 到 255，0 最暗，255 最亮。

范例：

```
#调节屏幕到最亮

car.screen.brightness(255)
```

三、清空屏幕(clear)

语法：

car.screen.clear()

说明：

清空屏幕上所有内容。

参数：

无

范例：

```
#清空屏幕

car.screen.clear()
```

四、关闭屏幕(close)

语法：

car.screen.close()

说明：

关闭屏幕，屏幕上的显示内容也会被同步清空。

范例：

```
car.screen.close()
```

五、旋转屏幕(rotate)

语法：

car.screen.rotate(mode:str)->bool

说明：

旋转屏幕，因为屏幕可以自由的安装到主控板任意一面或者外壳的任何地方，

如果显示内容的方向不对，则可以通过本函数接口旋转屏幕。

参数：

mode　　旋转方式：B-180 旋转，空格是默认模式。

返回值　　处理是否成功。

范例：

```
#旋转屏幕 180 度
car.screen.rotate("B")
```

六、查询屏幕尺寸(size)

语法：

car.screen.size()->(width, height)

说明：

查询返回当前屏幕的宽和高。

参数：

返回值　tuple 数组，（width, height)。

范例：

```
#查询登记的屏幕大小

car.screen.size()
```

七、绘制点(point)

语法：

car.screen.point(x:int, y:int, v:int)->bool

说明：

对屏幕上指定 XY 坐标的某一点设置或者清空显示。。

参数：

x　　屏幕上指定点的横坐标（从 0 开始，到屏幕宽减一结束）。

y　　屏幕上指定点的纵坐标（从 0 开始，到屏幕高减一结束）。

v　　0 代表清空该点的显示，1 代表该点点亮显示出来。

返回　处理是否成功。

范例：

```
#给屏幕上 x=5, y=6 的点设置开启

car.screen.point(5, 6, 1)
```

八、打印文本(print)

语法：

car.screen.print(text:str, x:int, y:int)->bool

说明：

在屏幕上指定 XY 坐标的位置开始打印 text 文本。本接口在打印内容超过屏幕边界以后，会忽略后续打印，不会自动换行到下一行。

参数：

text　　需要被打印的文本内容，支持 UTF-8 编码格式的各种中文外文字符。

x　　屏幕上指定点的 X 坐标。

y　　屏幕上指定点的 Y 坐标。

返回　　处理是否成功。

范例：

```
#在屏幕上 x=1, y=2 处打印 ABC 三个字母
car.screen.print("ABC", 1, 2)
```

九、绘制像素图基于字节(draw)

语法：

car.screen.draw(image:str, width:int, height:int, x:int, y:int)->bool

说明：

在屏幕上指定 XY 坐标的位置开始绘制 image。

参数：

image　代表图像的字符串，一个字节代表一个像素（空格是该像素不显示，其它字符显示）

width　图像自身的宽度（缺省值：屏幕宽）

height　图像自身的高度（缺省值：屏幕高）

x　　图像绘制到屏幕上左上角对 应点的 X 坐标(缺省值：0)。

y　　图像绘制到屏幕上左上角对应点的 Y 坐标(缺省值：0)。

返回　　处理是否成功。

范例：

```
#在屏幕 x=0, y=0 处绘制一条 14 个像素的虚线
car.screen.draw("  **  **  **  ", 14, 1, 0, 0)
```

十、绘制像素图基于位(draw_bits)

语法：

car.screen.draw_bits(image:str, width:int, height:int, x:int, y:int)->bool

说明：

在屏幕上指定 XY 坐标的位置开始绘制 image。

参数：

image　代表图像的字符串，一个字节里的每一个位(bit)代表一个像素（0 不显示，1 则显示），每一行最后一个像素如果不能抽满一个字节，则补全一个字节。下一行的第一个像素从新的字节开始。

width　图像自身的宽度(缺省值：屏幕宽)

height　图像自身的高度(缺省值：屏幕高)

x　　图像绘制到屏幕上左上角对应点的 X 坐标(缺省值：0)。

y　　图像绘制到屏幕上左上角对应点的 Y 坐标(缺省值：0)。

返回　　处理是否成功。

范例：

```
#在屏幕 x=0,y=0 处绘制一条 16 个像素的实线
car.screen.draw_bits(b"\xFF\xFF", 16, 1, 0, 0)
```

十一、刷新屏幕(flush)

语法：

car.screen.flush()->bool

说明：

重绘屏幕。在进行类似绘制点的操作，不会立刻看到效果，而需要执行这个 flush 输出绘制结果。

参数：

返回值　　处理是否成功。

范例：

```
car.screen.flush()
```

十二、汉字显示工具

工具链接：

http://guidan.com/appssd1306/

说明：

在输入框输入要显示的汉字，复制下方生成的代码到小龟小车 Python 编辑器，执行程序。

文字滚动效果选择。

参数：

返回值　　屏幕文字显示。

Web 开发手册

概要

小龟小车(小龟机器人)本身就是一台能提供 Web 服务的服务器，因此您可以用传统 Web 开发的相关技术来操控小龟小车，实现自己的想法，比如一个完全符合自己要求的操控台、全自动的批处理等等。小龟也非常感谢您能分享您的作品。

说明

所有可以进行 HTTP 访问的编程语言，不限 Java、Python、C 等等，都可以使用 HTTP 访问的方式来完整控制小车。

开发流程

您可以按照下面的开发流程来进行 Web 开发：

第一步：通过访问小龟小车官网(guidan.com)获取当前路由器下的所有小车的信息。

第二步：向指定小车的 IP 地址，发送您的操控请求。

获取当前路由器下所有小车

首先您需要通过 HTTP 或 HTTPS 请求访问以下网址，即可获得当前路由器(局域网)下的所有小车信息：

http://api.guidan.com/my/car.list

注意：只有获得小车的本地 IP 后，才能通过对这个 IP 发送各种 HTTP 请求来控制小车。

请求返回格式如下：

[{"ip": "192.168.5.95", "name": "\u5c0f\u8f661ad9", "model": "A2", "ver": "V211015", "update": 1634533876}]

这是一个符合 JSON4 规范的字符串，所以您可以用诸如"JSON.parse"来解析。

整体是一个数组，数组的每个元素是一辆小车的信息(JSON 对象格式)，对象内各个字段意义如下：

字段	作用	说明
ip	小车本地 ip	指令都是直接发到这个 IP
name	小车名称	
model	小车型号	
update	最新报告时间	

操作指定小车

比如，我们想让目标小车(IP：192.168.1.19)播放 Hello 开机音乐，则如下发送 HTTP 请求即可：

http://192.168.1.19/py?code=car.hello()

开发范例

JavaScript 范例

```
var xhr = new XMLHttpRequest();
//部分请求会非常耗时，请根据具体情况调整
//xhr.timeout = 5000;
xhr.onreadystatechange = function(){
        if(xhr.readyState!=4){return;}
        if(xhr.status!=200){return;}    //返回的答复报文格式
{"status":"OK", "msg":""}    var json = JSON.parse(xhr.responseText);

        //在下面处理
};
xhr.open("GET", "http://192.168.5.95/py?code=car.hello()", true);
xhr.send(null);
```

编程简谱格式

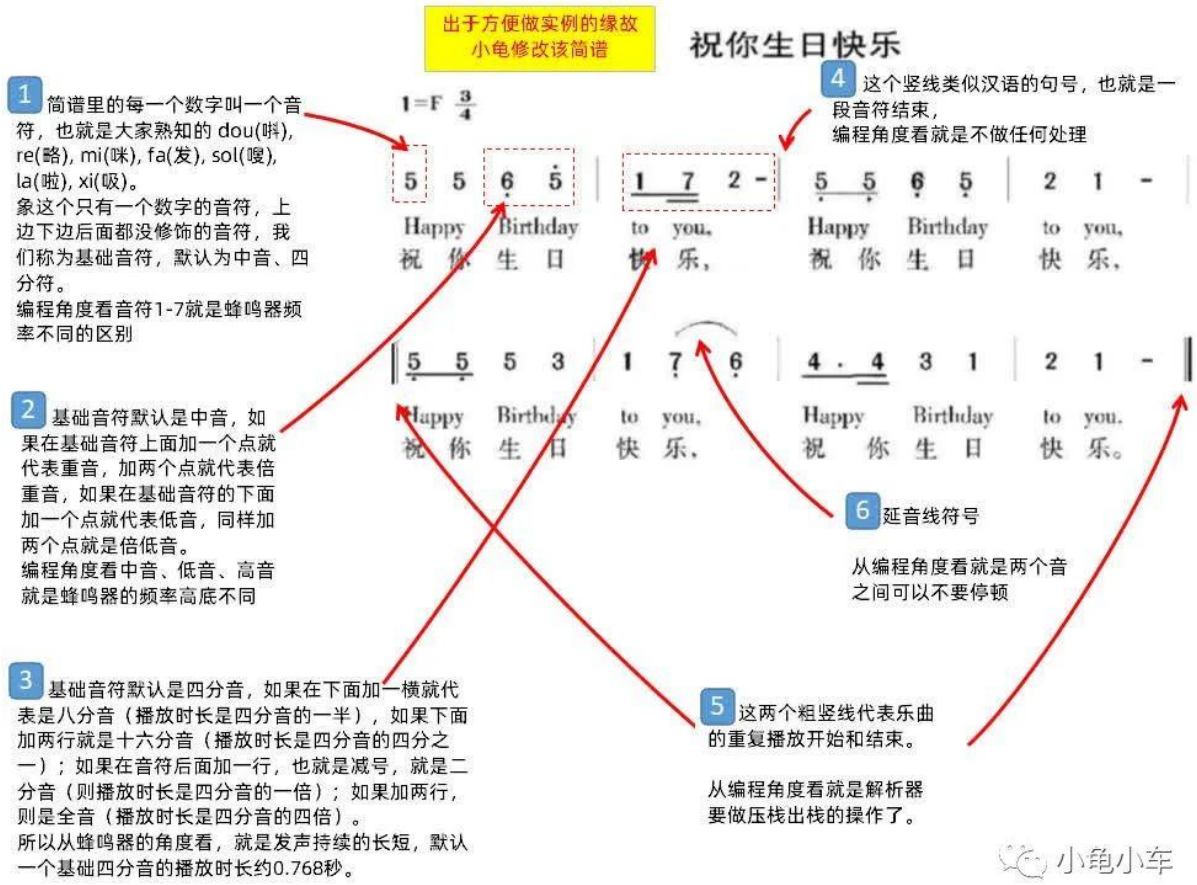
引言

下图是小伙伴们经常听到的《生日快乐歌》的音乐简谱，大家会发现真实的音乐简谱里面有非常多的不适合键盘输入的音符。

为了方便小伙伴们编程时简单又快速的输入音乐简谱，小龟特意设计了一套小车的简谱。它和真实音乐简谱的格式基本上是差不多的，只是对一些无法直接用键盘文本输入的音符进行了匹配改造。所以小伙伴们在实际使用过程中，如果真实音乐简谱上的符号在键盘上有的，就直接按键输入即可，没有的再参考下面的规则。

一、编程角度分析音乐简谱

首先小龟和大家简单科普一下音乐简谱的几个基本元素：

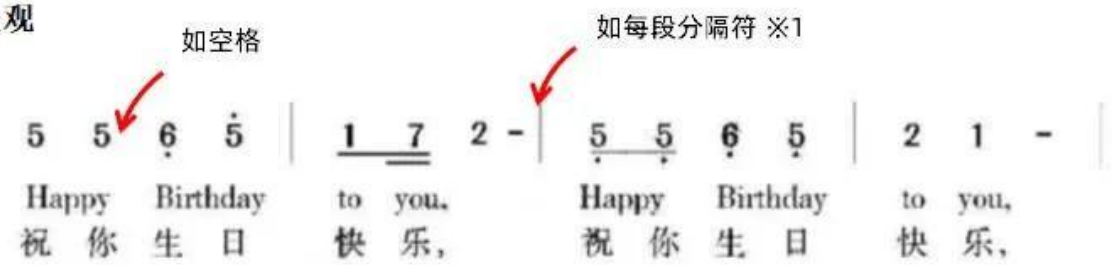


在分析了多个简谱以后，小龟发现其实大部分情况下的简谱都可以归纳成“后标识类”语法。也就是先一个代表基础音符的数字，然后跟随一堆对这个基础音符进行补充的定义，如下:

单个音符

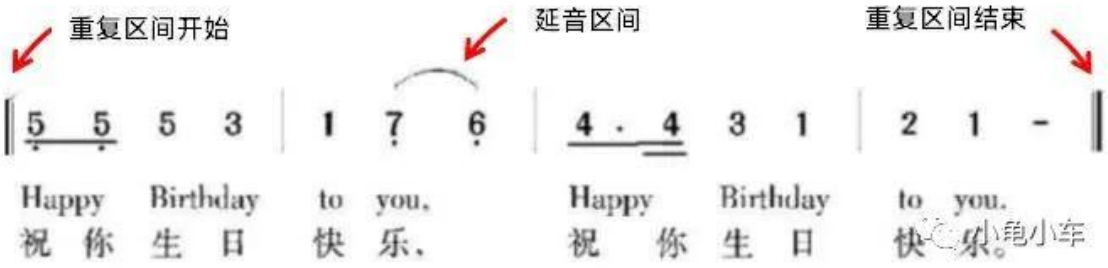
基础音阶　　对基础音阶的附加标识

仅美观



※1 每段分隔符实际上起到歌曲先高后低、抑扬顿挫的作用，
但对于这次的简单电路来说暂时忽略掉。

区间符号



于是我们设计了下面的简谱语法对应规则。

二、小龟简谱和真实简谱的对应规则

音阶的对应关系：

音阶	简谱表示符号	小龟表示符号	说明
dou	1	1	
re	2	2	

mi	3	3	
fa	4	4	
sol	5	5	
la	6	6	
si	7	7	
空	0	0	

说明：音阶部分和真实简谱的符号完全一致。

重音低音的表示：

音频	简谱表示符号	小龟表示符号	说明
倍重音		2''	两个单引号表示为倍高音
高音		2'	单引号表示为高音
中音（标准）	2	2	一致
低音		5,	逗号表示为低音
倍低音		5,,	两个逗号表示为倍低音

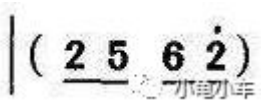
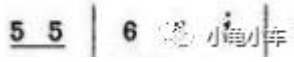
音节的表示：

音节	简谱表示符号	小龟表示符号	说明
全音	2----	2----	两者一致，用四个减号
二分音	2-	2-	两者一致，用一个减号
四分音	2	2	两者一致
八分音	2	2_	在音阶后面跟下划线
十六分音		2=	在音阶后面跟

			等号
半音	4 .	4.	在音阶后面跟句号

其它符号的表示：

符号	简谱表示符号	小龟表示符号	说明
空格	空格	空格	两者一致
延音		{ 6 - | 6 }	用大括号来表示,左大括号为连音始,右大括号为连音结束
重复		[3_2_ | 1]	用中括号表示,左中括号为重复开始,右中括号为重复结束。暂不支持复杂重复。 和真实简谱一样,左中括号省略代表从头

			开始重复
伴奏		|(2_5_ 6_ 2'_)	两者一致,用括号表示
段落		5_5_ | 6 5 1' |	两者一致,都用竖线表示

另外少量真实简谱上的符号，小龟暂时还不支持，可以忽略不转换。

转换范例

生日快乐歌：

转换后的小龟编程简谱：

```
car.buzzer.music(""" 5_5_ | 6 5 1' | 7 0 5_5_ | 6 5 2' | 1' - 5_5_ | 5' 3' 1' |
7 {6 - | 6} 0 4'_4'_ | 3' 1' 2' | 1' - ( 5_5_ ) ] """)
```

姿态传感器

概要

小龟主控板自带了一枚六轴姿态传感器（三轴加速度 + 三轴陀螺仪）所有姿态传感器的接口都位于 car.motion 下。

版本要求：不低于 V211216

姿态传感器的坐标系示意图：

一、读取三轴加速度实时值

语法：

car.motion.accel() -> [x, y, z]

说明：

读取三轴加速度计实时值。

参数：

返回值　　x, y, z 三轴加速度的实时值，单位是 g (1g = 1 个地球重力加速度)

范例：

```
#读取三轴加速度计的实时值 car.motion.accel()#(0.01, 0.01, 0.98111)
```

二、读取三轴陀螺仪实时值

语法：

car.motion.gyro() -> [x, y, z]

说明：

读取三轴陀螺仪的实时值，单位是 dps。

参数：

返回值　　x, y, z 三轴陀螺仪的实时值

范例：

```
#读取三轴陀螺仪的实时值 car.motion.gyro()
```

三、读取芯片温度

语法：

car.motion.temp() -> float

说明：

读取姿态传感器芯片温度。如果芯片没有高负荷工作情况下，该温度可以拿来做环境温度。

参数：

返回值　　芯片温度

范例：

```
#读取芯片温度 car.motion.temp()
```

集群操控

概要

小龟机器人支持集群功能，您可以通过集群功能实现小龟机器人 A 对同个路由器下其它小龟机器人 BCD...的访问。

一、获取当前路由器下所有机器人（list）

语法：

car.friend.list()

二、获取当前路由器下所有机器人信息。

参数：

返回值　　当前路由器下所有可以控制的机器人列表度

范例：

```
#获取当前路由器下所有机器人信息 car.friend.list()
```

三、给其他机器人发送指令（tell）

语法：

car.friend.tell(ip, python_code)

四、往默认串口输出数据

参数：

ip　　对方的 IP 地址

python_code　需要对方执行的 Python 代码

返回值　　执行结果

范例：

```
#给其它机器人发送指令 car.friend.tell("192.168.10.15", "car.hello")
```

摄像头接口

概要

小龟主控板支持摄像头模块，用户可以通过主控板的摄像头接口接入摄像头。

目前小龟主控板只支持 OV2640 芯片的摄像头模块。

一、摄像头模式说明

为了节省 CPU 的计算量，我们为摄像头设计了两种模式：视频模式和视觉模式。用户可以根据自己的需求进行模式的切换。视频模式下，可以进行全彩色的视频遥控图传，也支持多种摄像头分辨率。视觉模式下，仅仅用于车载视觉，不再进行全彩色的视频图传，同时摄像头的分辨率会降低。

二、重新启动摄像头

语法：

car.camera.reload()

说明：

重新加载摄像头

参数：

返回值　　加载是否成功

范例：

```
#重装摄像头 car.camera.reload()
```

三、摄像头参数读写

语法：

car.camera.sccb(reg)　读取摄像头参数

car.camera.sccb(reg, val)　修改摄像头参数

说明：

摄像头芯片读写接口，可以通过摄像头芯片支持的 sccb 协议说明书来读 取或者修改摄像头参数。

参数：

返回值

读取模式，返回读取的值

修改模式，返回修改是否成功。

范例：

```
#读取摄像头芯片参数 car.camera.sccb(25)
#修改摄像头芯片参数 car.camera.sccb(25, 15)
```

视觉接口

概要

视觉模式需要开关专门打开。为了节约 CPU 的计算量，小龟小车的视觉模式的图像被限制为 240 个像素宽，160 个像素高，每个像素值是 8 位(一个字节，256 色模式）。小龟小车固件集成了一些简单的视觉识别接口，比如水平线段查找等。

大部分视觉处理，都会按照下面的基本流程进行：

1. 用 car.vision.snap()采样一帧图像，
2. 用 car.vision.row()或者 car.vision.point()等进行具体的图像数据分析，
3. 执行相应的动作
4. 如果是动态跟踪一类处理，则循环进行前面的动作。

色彩空间与通道

小龟相信很多小伙伴都清楚 RGB 三色原理。这个 RGB 就是一个色彩空间，而其中单个的 R、单个的 G、单个的 B 是三个独立的通道。小龟小车的视觉模式目前仅仅支持一个色彩空间里的一个通道。因为视觉处理的最基础工作就是选择最合适的色彩空间和通道，所以小龟小车也支持下面这些色彩空间和通道。

RGB 色彩空间（R-红色通道、G-绿色通道、B-蓝色通道）

YCbCr 色彩空间（Y-灰度通道，Cb-蓝绿通道，Cr-红绿通道）

HSL 色彩空间（H-色相通道）

Python 接口中涉及到通道配置的参数，则使用上面黄颜色背景字母做为通道名称来配置调用。

一、car.vision.on(channel)->bool

小车开机后默认处于视频监控模式，需要通过这个接口来切换到视觉模式。

参数：

channel　　需要使用的色彩空间的通道名称。

返回值　　处理是否成功。

范例：

```
#打开视觉模式，并选择 YCbCr 色彩空间的 Y(灰度)通道
car.vision.on('Y')#打开视觉模式，并选择 RGB 色彩空间的 B(蓝色)通道
car.vision.on('B')
```

二、car.vision.off()->bool

小车的视觉模式，回到默认的视频监控模式。

```
#关闭视觉模式 car.vision.off()
```

三、car.vision.channel()->str

返回小车当前视觉模式选择的色彩空间通道名称。

```
car.vision.channel()#>>Y
```

四、car.vision.frame_id()->int

返回当前抓取的样本帧的编号。

五、car.vision.height()->int

视觉模式图像的宽度（单位：像素），固定为 160 像素。

六、car.vision.line_find([rows], min, max, min_width, noise)->(x, y, width)

寻找当前采样图像中的符合要求的水平线段，调用本函数前需要先执行采样一帧的命令。

参数：

rows　　指定寻找的水平行，比如 80 代表寻找第 80 行，[70, 80]代表依次寻找第 70 行、第 80 行。

min　　指定的寻找范围的最小值

max　　指定的寻找范围的最大值

min_width　　指定的寻找水平线段的最小长度

noise　噪点范围，0 代表无噪点，3 代表忽略间隔小于等于 3 的噪点。

返回值　x 和 y 代表符合的线段中心点的 x 和 y 坐标，width 代表符合的线程长度；如果本接口没有找线段到则返回 None。

范例：

```
#寻找第 80 行，范围在 80-125 之间长度大于 15 个像素的水平线段，允许间隔噪点 1 个像素 car.vision.snap()car.vision.line_find(80, 80, 125, 15, 1)#>> (48, 80, 20)  #找到了符合要求的水平线段，线段中心位置 x=48，y=80，宽度 20 个像素。
#依次在第 80、70、90 行寻找范围在 80-125 之间长度大于 15 个像素的水平线段，允许间隔噪点 0 个像素
car.vision.line_find([80, 70, 90], 80, 125, 15)
#>> None     #没有找到符合要求的水平线段
```

七、car.vision.point(x, y)->int

返回当前采样图片里指定坐标位置的值，调用本函数前需要先执行采样一帧的命令。

参数：

x　　水平坐标位置，范围等于大于 0，到小于 240。

y　　垂直坐标位置，范围等于大于 0，到小于 160。

返回值　指定位置的值，单字节数值等于大于 0 到小于 256。

范例：

```
#确定 (x=5, y=10)的像素点是否为白色(0xFF)，它下方的点是否为暗红色(0x00)car.vision.snap()if car.vision.point(5, 10)==0xFF && car.vision.point(5, 11)==0x00):    print("find")
```

八、car.vision.row(row_index)->list

返回当前采样图片里指定行的全部点的值，调用本函数前需要先执行采样一帧的命令。

参数：

row_index　　指定行的索引，范围等于大于 0 到小于 160。

返回值　指定行的所以点值的列表

范例：

```
#从第 80 行依次查找，直到遇到值为 0xFF 的点 car.vision.snap()data =
car.vision.row(80)for i in range(240):    if data[i]==0xFF:
print("find")
```

九、car.vision.snap()->int

采样一帧图像用于后续处理。各种视觉处理的第一步就是先执行这个采样接口，采样一帧。采样后，该帧数据会被冻结在内存中供后续各种图像处理，但是图传显示的图片还会继续更新的。如需查看该冻结的采样帧，可以使用页面访问接口。

十、car.vision.width()->int

视觉模式图像的宽度（单位：像素），固定为 240 像素。

DHT11 温湿度传感器接口

概要

机器人主控板支持对 DHT11 温湿度传感器模块的支持，您可以通过以下命令快速获得传感器对应的测量值。

一、传感器接口

获取当前温度湿度测量值(read)

语法：

car.dht11.read(管脚) -> (湿度，温度)

说明：

获取 DHT11 模块当前温度湿度的测量值。

参数：

返回值　　(湿度值，温度值)，测量异常的场合返回 None。

范例：

```
#获取连接在 A1 管脚上的 DHT11 传感器的值 car.dht11.read(A1)#结果 (7.0, 21.01)  代表湿度是 7.0，  温度是 21.01 摄氏度
```