_示例代码_单电机

2019年4月12日 20:07

lm1 = LargeMotor(OUTPUT_A)

lm2 = LargeMotor(OUTPUT_B)

mm = MediumMotor(OUTPUT_D)

查看连接端口

```
debug_print('address: ', mm.address)
address:  ev3-ports:outD
```

查看可执行指令

```
debug_print('commands: ', mm.commands)


commands:  ['run-forever', 'run-to-abs-pos',
'run-to-rel-pos', 'run-timed', 'run-direct',
'stop', 'reset']
```

查看电机每1转的计数

```
debug_print('count per rotation for LargeMotor: ', lm1.count_per_rot)
debug_print('count per rotation for MediumMotor: ',
mm.count_per_rot)
```

count per rotation for LargeMotor: 360

count per rotation for MediumMotor: 360

查看该电机驱动器的名称

```
debug_print('driver name: ', mm.driver_name)
river name:  lego-ev3-m-motor
```

查看占空比

```
debug_print('current duty circle: ', mm.duty_cycle)
```

current duty circle: 0

写入设置占空比设定值，返回当前值

```
debug_print('set duty circle: ', mm.duty_cycle_sp)
```

```
set duty circle:  0
```

设置占空比设定值

```
mm.duty_cycle_sp = -80
debug_print('current duty circle setted: ', mm.duty_cycle_sp)
```

```
current duty circle setted:  -80
```

电机的极性

如果设置为'normal'，+值正转

如果设置为'inversed'，+值反转

```
debug_print('polarity for LargeMotor: ', lm1.polarity)
debug_print('polarity for MediumMotor: ', mm.polarity)
```

```
polarity for LargeMotor:  normal
polarity for MediumMotor:  normal
```

查看电机在旋转中的位置（可看作在一条数轴上左右移动）

顺时针旋转时数值增加，逆时针旋转时数值减小

```
debug_print('position: ', mm.position)
```

```
position:  0
```

查看位置PID的比例常数

```
debug_print('位置pid的比例常数：', mm.position_p)
```

```
\u4f4d\u7f6epid\u7684\u6bd4\u4f8b\u5e38\u6570\uff1a 160000
```

查看位置PID的积分常数

```
debug_print('位置pid的积分常数：', mm.position_i)
```

```
\u4f4d\u7f6epid\u7684\u79ef\u5206\u5e38\u6570\uff1a 0
```

查看位置PID的导数常数

```
debug_print('位置pid的导数常数: ', mm.position_d)
```

```
\u4f4d\u7f6epid\u7684\u5bfc\u6570\u5e38\u6570:  0
```

查看为以位置为基础的运行模式设置的目标位置，可转换为角度或

旋转数

```
debug_print('不知道是啥: ', mm.position_sp)
```

```
\u4e0d\u77e5\u9053\u662f\u5565:  0
```

查看理论上的最大速度

```
debug_print('max speed for LargeMotor', mm.max_speed)
debug_print('max speed for MediumMotor', lm1.max_speed)
```

```
max speed for LargeMotor 1560
max speed for MediumMotor 1050
```

读取当前每秒tacho计数，负值代表反方向旋转

```
debug_print('current motor speed in tacho count: ', mm.speed)
```

```
current motor speed in tacho count:  0
```

读取当前提升设定值

```
debug_print('current ramp up speed: ', mm.ramp_up_sp)
```

```
current ramp up speed:  0
```

读取当前减小设定值

```
debug_print('current ramp down speed: ', mm.ramp_down_sp)
```

```
current ramp down speed:  0
```

查看速度调节pid的比例常数

```
debug_print('速度调节pid的比例常数', mm.speed_p)
```

```
\u901f\u5ea6\u8c03\u8282pid\u7684\u6bd4\u4f8b\u5e38
\u6570 1000
```

查看速度调节pid的积分常数

```
debug_print('速度调节pid的积分常数', mm.speed_i)
```

```
\u901f\u5ea6\u8c03\u8282pid\u7684\u79ef\u5206\u5e38
```

\u6570 60

查看速度调节pid的导数常数

```
debug_print('速度调节pid的导数常数', mm.speed_d)
```

\u901f\u5ea6\u8c03\u8282pid\u7684\u5bfc\u6570\u5e38\u6570 0

查看可允许的状态，如'running'、'ramping'、'holding'

```
debug_print('a list of state flags for MediumMotor', mm.state)
debug_print('a list of state flags for LargeMotor', lm1.state)
```

a list of state flags for MediumMotor []
a list of state flags for LargeMotor []

查看停止状态下的行为表

```
debug_print('all stop actions for MediumMotor', mm.stop_actions)
debug_print('all stop actions for LargeMotor', lm1.stop_actions)
```

all stop actions for MediumMotor ['coast', 'brake', 'hold']

all stop actions for LargeMotor ['coast', 'brake', 'hold']

设置停止状态下的行为

```
mm.stop_action = 'hold'
debug_print('current stop action: ', mm.stop_action)
```

current stop action: hold

查看 ‘run-timed'状态下需要持续的时间

```
debug_print('the amount of time the motor will run in ''run-timed mode' , mm.time_sp)
```

the amount of time the motor will run in run-timed mode 0

根据'duty_cycle_sp'设置的占空比运转，与其他command不同，对'duty_cycle_sp'的修改即时执行

查看电机是否供电

```
if mm.is_running is None:
debug_print('no')
else:
debug_print(mm.is_running)
False
```

查看电机在达到稳定值前是否增加或减小

```
if mm.is_ramping is None:
debug_print('no')
else:
debug_print(mm.is_ramping)
False
```

判断电机是停止转变还是保持固定值

```
if mm.is_holding is None:
debug_print('no')
else:
debug_print(mm.is_holding)
False
```

查看电机是否过载

```
if mm.is_overloaded is None:
debug_print('no')
else:
debug_print(mm.is_overloaded)
False
```

查看电机是否故障

```
if mm.is_stalled is None:
debug_print('no')
else:
debug_print(mm.is_stalled)
False
```

阻塞直到电机状态变为
`stalled`或`holding`

(run_timed等函数根据speed等sp全局变量运行，因此需要在使用前修改对应值)

```
sv.speed_sp = int(SpeedDPS(360).to_native_units(sv))
sv.time_sp = int(10 * 1000)
sv.run_timed()
sv.wait_until_not_moving()  # 这里的not_moveing是指run_timed执行完毕
```

阻塞直到状态变为 s

```
sv.speed_sp = int(SpeedDPS(360).to_native_units(sv))
sv.time_sp = int(10 * 1000)
sv.run_timed()
sv.wait_until('stalled')
```

阻塞直到 s 不再为当前状态(self.state)

```
debug_print('wait while')
sv.speed_sp = int(SpeedDPS(360).to_native_units(sv))
sv.time_sp = int(10 * 1000)
sv.run_timed()
sv.wait_while('running')
```

控制电机根据旋转数和速度运转

```
lm1.on_for_rotations(SpeedRPS(-1), 5)
```

控制电机根据角度和速度运转

```
lm2.on_for_degrees(SpeedPercent(80), -720)
```

控制电机根据位置和速度运转

```
lm1.on_to_position(SpeedDPS(180), -80)
```

控制电机根据速度、持续时间旋转

```
lm2.on_for_seconds(SpeedPercent(-80), 3)
lm2.on_for_seconds(SpeedPercent(80), 3) # 上一个执行完再执行下一个,不考虑time.sleep
```

控制电机按照速度持续运转

```
lm1.on(SpeedPercent(50))
```

控制电机停止运转

```
lm1.off()
```

返回当前旋转圈数

```
debug_print('current rotations: ', lm1.rotations)
current rotations:  2.6694444444444443
```

返回旋转角度

```
debug_print('current degrees: ', lm1.degrees)
current degrees:  964.9999999999999
```