Guia Rapido - Joystick Analogico

Modulo: KY-023 | 2 eixos analogicos + botao digital

1. O que e esse hardware?

O joystick KY-023 e composto por dois potenciometros (resistores variaveis) para os eixos X e Y, mais um botao digital (SW) que e acionado ao pressionar o cabo. Os potenciometros geram tensoes analogicas entre 0V e 3.3V conforme a posicao. O RP2040 le esses valores via ADC (conversor analogico-digital).

```
Centro (repouso): VRx = VRy = ~1.65V -> ADC = ~32767
Maximo (eixo +):  VRx ou VRy = 3.3V  -> ADC = 65535
Minimo (eixo -):  VRx ou VRy = 0V    -> ADC = 0

SW (botao): PULL_UP interno -> 1 = solto, 0 = pressionado
```

2. Conexao na BitDogLab

Sinal	GPIO	Tipo	Descricao
VRy	GPIO26	ADC0 (analogico)	Eixo Y (vertical) 0-3.3V
VRx	GPIO27	ADC1 (analogico)	Eixo X (horizontal) 0-3.3V
SW	GPIO22	Digital (entrada)	Botao de pressao (pull-up)

GPIO26=ADC0 e GPIO27=ADC1 sao os unicos pinos ADC disponiveis para o joystick. GPIO28=ADC2 e usado pelo microfone.

3. Zona morta (deadzone)

O centro mecanico do joystick raramente le exatamente 32767. Ha uma variacao natural de +/-2000 a +/-5000 unidades mesmo em repouso. Sempre implemente uma zona morta para evitar movimento involuntario quando o joystick esta solto.

```
CENTER   = 32767
DEADZONE = 4000

def em_zona_morta(valor):
    return abs(valor - CENTER) < DEADZONE
```

4. Codigo base em MicroPython

```
from machine import Pin, ADC
import time

jx  = ADC(Pin(27))   # eixo X
jy  = ADC(Pin(26))   # eixo Y
jsw = Pin(22, Pin.IN, Pin.PULL_UP)  # botao

CENTER   = 32767
DEADZONE = 4000
```

```
while True:
    x = jx.read_u16()   # 0 a 65535
    y = jy.read_u16()
    sw = jsw.value()     # 0 = pressionado

    dx = x - CENTER      # deslocamento X (-32767 a +32767)
    dy = y - CENTER      # deslocamento Y

    if abs(dx) > DEADZONE:
        print('X:', 'direita' if dx > 0 else 'esquerda')
    if abs(dy) > DEADZONE:
        print('Y:', 'baixo' if dy > 0 else 'cima')
    if sw == 0:
        print('Botao pressionado!')
    time.sleep(0.05)
```

5. Mapear para faixa customizada

Para converter o valor ADC para uma faixa util (ex: posicao 0-4 numa matriz 5x5):

```
def mapear(valor, min_in, max_in, min_out, max_out):
    return int((valor - min_in) / (max_in - min_in) * (max_out - min_out) + min_out)

col = mapear(jx.read_u16(), 0, 65535, 0, 4)   # 0 a 4
lin = mapear(jy.read_u16(), 0, 65535, 0, 4)   # 0 a 4
```

6. Especificacoes do modulo KY-023

Parametro	Valor
Eixos analogicos	2 (X e Y), potenciometros
Faixa de saida	0V a VCC (0 a 65535 no ADC)
Resolucao ADC (RP2040)	12-bit (0-4095) interno, 16-bit via read_u16()
Botao SW	SPST-NO, pull-up, ativo em LOW
Tensao de operacao	3.3V (compativel com RP2040)

7. Aplicacoes praticas

- Mover cursor ou personagem numa matriz/display.
- Controlar velocidade: velocidade proporcional ao deslocamento do eixo.
- Selecionar opcao de menu: direita/esquerda para navegar.
- Controle de angulo: mapear ADC para graus (0-180) para servomotor.

Ref: BitDogLab HDB | v7 e v6: VRy=GPIO26 (ADC0), VRx=GPIO27 (ADC1), SW=GPIO22