Guia Rapido - Buzzer Passivo

Tipo: Buzzer Piezoeletrico Passivo | Controlado por PWM

1. O que e esse hardware?

O buzzer da BitDogLab e um buzzer PASSIVO (piezoeletrico). Diferente do buzzer ATIVO (que bipa sozinho ao receber 3.3V), o passivo precisa de um sinal PWM oscilando na frequencia desejada para gerar som. Isso permite tocar qualquer nota musical ou criar efeitos sonoros variando a frequencia do PWM.

```
Buzzer ATIVO:  recebe 3.3V -> bipa em frequencia fixa (sem controle de tom)
Buzzer PASSIVO: recebe PWM -> frequencia do PWM = frequencia do som gerado

Regra: freq(440) -> LA (nota A4)
       freq(262) -> DO (nota C4)
       duty_u16(0) -> silencio (sem cortar o PWM)
```

2. Conexao na BitDogLab

Buzzer	GPIO	Versao	Observacao
Buzzer A	GPIO21	v7	Via transistor NPN (maior potencia)
Buzzer B	GPIO10	v7	Conexao direta ao GPIO
Buzzer	GPIO10	v6	Unico buzzer (GPIO10 nao e botao C)

O Buzzer A usa um transistor como amplificador de corrente: o GPIO21 aciona a base do transistor, que por sua vez aciona o buzzer com mais corrente do que o GPIO sozinho forneceria. O Buzzer B e controlado diretamente pelo GPIO10.

3. Como o PWM gera som

O buzzer piezoeletrico vibra na mesma frequencia do sinal eletrico aplicado. Essa vibracao mecanica move o ar e gera o som. O duty cycle (largura do pulso) controla o VOLUME: 50% (32768) e o ponto de maior amplitude, valores acima ou abaixo reduzem o volume. duty_u16=0 = silencio total sem desligar o pino.

```
Frequencia (freq)  -> define o TOM (nota musical)
Duty cycle (duty)  -> define o VOLUME

50% duty  (32768) = volume maximo
25% duty  (16384) = volume medio
duty_u16(0)       = silencio (buzzer quieto, PWM ativo)
```

4. Codigo base em MicroPython

```
from machine import Pin, PWM
import time

buzzer = PWM(Pin(21))   # Buzzer A (v7)
# buzzer = PWM(Pin(10)) # Buzzer B (v7) ou unico (v6)
```

```
def tocar(freq_hz, duracao_ms, volume=32768):
    buzzer.freq(freq_hz)
    buzzer.duty_u16(volume)
    time.sleep_ms(duracao_ms)
    buzzer.duty_u16(0)        # silencio entre notas
    time.sleep_ms(20)

def silencio():
    buzzer.duty_u16(0)

tocar(440, 300)    # LA  por 300ms
tocar(523, 300)    # DO
tocar(659, 300)    # MI
silencio()
```

5. Notas musicais - frequencias

Nota	Freq (Hz)	Nota	Freq (Hz)	Nota	Freq (Hz)
DO (C4)	262	FA (F4)	349	SI (B4)	494
RE (D4)	294	SOL (G4)	392	DO (C5)	523
MI (E4)	330	LA (A4)	440	---	---

```
# Dicionario de notas para facil uso:
NOTAS = {
    'C4':262, 'D4':294, 'E4':330, 'F4':349,
    'G4':392, 'A4':440, 'B4':494, 'C5':523,
    'C3':131, 'G3':196, 'A3':220, 'B3':247,
}

# Exemplo: toca a melodia de Parabens
melodia = [('C4',200),('C4',200),('D4',400),('C4',400),
           ('F4',400),('E4',800)]
for nota, ms in melodia:
    tocar(NOTAS[nota], ms)
```

6. Efeitos sonoros uteis

```
# Alarme (dois tons alternados)
for _ in range(5):
    tocar(880, 200)   # tom agudo
    tocar(440, 200)   # tom grave

# Bip de confirmacao
tocar(1000, 100)

# Bip de erro (tom descendente)
for f in [800, 600, 400]:
    tocar(f, 150)
```

```
# Sirene (sweep de frequencia)
import time
for f in range(300, 1500, 20):
    buzzer.freq(f)
    buzzer.duty_u16(32768)
    time.sleep_ms(10)
buzzer.duty_u16(0)
```

7. O que ajustar na pratica

- Se o som estiver muito fraco: aumentar volume (duty_u16 ate 32768 = 50%).
- Frequencias audiveis: 100Hz (grave) a 4000Hz (agudo). Acima de 4kHz pode nao ser audivel no buzzer.
- Faixa mais clara na maioria dos buzzers piezo: 1kHz a 4kHz.
- Use duty_u16(0) para silencio (mantem PWM ativo) em vez de deinit() entre notas.
- Para musica: deixe um pequeno intervalo entre notas (20-50ms) para separa-las.

Ref: BitDogLab HDB (Unicamp) | v7: BuzzerA=GPIO21(transistor), BuzzerB=GPIO10 | v6: Buzzer=GPIO10