To read an **audio signal** from your mobile phone’s **headphone output** using an **ESP32**, you need to properly condition the signal before feeding it into the ESP32’s **ADC (Analog-to-Digital Converter)**. The ESP32 can only read **positive voltages (0V to 3.3V)**, but an audio signal is an **AC waveform** that swings between positive and negative voltages (around -1V to +1V from a headphone jack).

**Steps to Read Audio Signal from Headphone Jack**

**1. Signal Conditioning Circuit**

* **Bias the Signal**: Use a **voltage divider** (two 100kΩ resistors) to shift the audio signal so it stays within the ESP32’s ADC range.
* **Capacitor for DC Blocking**: A **1µF capacitor** removes any DC offset from the mobile phone output.
* **Resistor for Protection**: A **10kΩ resistor** ensures safe ADC readings.

**2. Connecting to ESP32**

* Use **GPIO 34** (or any ADC-capable pin) to read the audio signal.
* Supply **3.3V** from the ESP32 for the voltage divider.

**Circuit Diagram (Text-Based)**

Mobile Audio Out (Tip) ----||---- ADC (GPIO 34, ESP32)

C1 (1µF)

Voltage Divider:

3.3V ---- [100kΩ] ----+---- [100kΩ] ---- GND

|

ADC Pin

* **C1 (1µF capacitor)**: Blocks DC offset from the mobile.
* **100kΩ + 100kΩ resistors**: Shift the signal to center it around 1.65V.

**MicroPython Code to Read Audio Signal**

import machine

import time

ADC\_PIN = 34 # ESP32 ADC pin for audio signal

# Initialize ADC

adc = machine.ADC(machine.Pin(ADC\_PIN))

adc.atten(machine.ADC.ATTN\_11DB) # Allows reading from 0V to 3.3V

while True:

audio\_value = adc.read() # Read ADC value (0-4095)

print("Audio Level:", audio\_value)

time.sleep(0.01) # Small delay for real-time reading

**How It Works**

1. The **ADC reads values from 0 to 4095** (since ESP32 has a 12-bit ADC).
2. The **bias circuit ensures the signal stays within the ADC range**.
3. The script prints real-time **audio level values** to check if the signal is being read properly.

Would you like a **detailed electronic schematic** for this? 🚀