I found a ADS1115 I2C breakout . Can you explain me how can I use it with ESP32 and micropython

**ChatGPT said:**

Memory updated

Great! The **ADS1115 I2C breakout** is an excellent choice for precise voltage measurements with the ESP32. I'll guide you through **wiring, installing the library, and writing MicroPython code**.

**🔌 Wiring ESP32 with ADS1115**

Since ADS1115 communicates over **I²C**, you'll connect it like this:

| **ADS1115 Pin** | **ESP32 Pin** |
| --- | --- |
| VDD | 3.3V |
| GND | GND |
| SDA | GPIO 21 (or any I²C SDA pin) |
| SCL | GPIO 22 (or any I²C SCL pin) |
| ADDR | GND (for I2C address 0x48) |
| A0, A1, A2, A3 | Your sensor/signal input |

📌 **Note:**

* The **ADDR pin** selects the I²C address:
  + GND → **0x48** (default)
  + VDD → **0x49**
  + SDA → **0x4A**
  + SCL → **0x4B**

**📥 Installing the ADS1115 Library**

**Option 1: Using Thonny (Easiest)**

1. Open **Thonny IDE**.
2. Connect ESP32 and go to **Tools → Manage Packages**.
3. Search for **micropython-ads1x15**.
4. Click **Install**.

**Option 2: Manually Upload Library**

1. Download the **ads1x15.py** library from GitHub:  
   👉 <https://github.com/micropython/micropython-lib>
2. Upload **ads1x15.py** to your ESP32 using Thonny (drag & drop into the ESP32 file system).

**📝 MicroPython Code for ADS1115 on ESP32**

python

Copy

from machine import I2C, Pin

import time

import ads1x15

# Initialize I2C on ESP32

i2c = I2C(0, scl=Pin(22), sda=Pin(21)) # Adjust pins if needed

# Create ADC object

adc = ads1x15.ADS1115(i2c, address=0x48) # Default I2C address

# Set gain (smaller gain for higher voltage range)

adc.setGain(ads1x15.GAIN\_ONE) # ±4.096V range

while True:

raw\_value = adc.read(0) # Read from AIN0 (Channel 0)

voltage = raw\_value \* (4.096 / 32767) # Convert to voltage

print(f"Raw: {raw\_value}, Voltage: {voltage:.3f} V")

time.sleep(1)

**📌 Understanding the Code**

* **ads1x15.ADS1115(i2c, address=0x48)** → Initializes ADS1115.
* **adc.setGain(ads1x15.GAIN\_ONE)** → Sets input range to **±4.096V**.
* **adc.read(0)** → Reads from **AIN0**.
* **voltage = raw\_value \* (4.096 / 32767)** → Converts raw ADC value to volts.

**⚡ Choosing the Right Gain**

| **Gain Setting** | **Voltage Range (±V)** | **Resolution per Step (mV)** |
| --- | --- | --- |
| GAIN\_TWOTHIRDS | ±6.144V | 0.1875 |
| GAIN\_ONE | ±4.096V | 0.1250 |
| GAIN\_TWO | ±2.048V | 0.0625 |
| GAIN\_FOUR | ±1.024V | 0.03125 |
| GAIN\_EIGHT | ±0.512V | 0.015625 |
| GAIN\_SIXTEEN | ±0.256V | 0.0078125 |

🔹 **Use lower gain for higher voltages** (e.g., GAIN\_TWOTHIRDS for up to 6.144V).  
🔹 **Use higher gain for small signals** (e.g., GAIN\_SIXTEEN for millivolts).

**🔧 Need More Features?**

* **Read multiple channels** → adc.read(1), adc.read(2), etc.
* **Measure differential voltage**:

python

Copy

diff\_value = adc.read(ads1x15.DIFF\_0\_1) # AIN0 - AIN1

* **Trigger interrupts on a threshold** (useful for low-power applications).

**🚀 Next Steps**

Would you like help **integrating ADS1115 with sensors** (e.g., Ohm meter, temperature, etc.)? 😊

4o

**You said:**

Yes with ohm meter

**ChatGPT said:**

Memory updated

**📏 Creating an Ohm Meter with ADS1115 & ESP32**

We'll use **Ohm’s Law** to measure resistance with a **voltage divider circuit**:

Runk=Rfixed×(VmeasuredVsupply−Vmeasured)R\_{unk} = R\_{fixed} \times \left( \frac{V\_{measured}}{V\_{supply} - V\_{measured}} \right)Runk​=Rfixed​×(Vsupply​−Vmeasured​Vmeasured​​)

Where:

* **Runk** = Unknown resistance
* **Rfixed** = Known reference resistor (e.g., 10kΩ)
* **Vmeasured** = Voltage read by ADS1115
* **Vsupply** = 3.3V (ESP32 power)

**🔌 Wiring ESP32 + ADS1115 for Ohm Meter**

pgsql

Copy

ESP32 ADS1115 Rfixed Runk GND

3.3V ---> VDD

GND ---> GND

SDA ---> SDA

SCL ---> SCL

AIN0 ----| 10kΩ |------| Runk |------ GND

|

Vout (Voltage measured by ADS1115)