

ohmetro digitale con ESP32

Un ohmetro misura la **resistenza** di un componente, e per farlo con l'ESP32 (che non ha un ADC particolarmente preciso, ma sufficiente per questo uso) possiamo usare il **metodo del partitore di tensione**.

Materiali necessari:

- ESP32 Dev Board
- Resistenza di riferimento (ad esempio 10 kΩ, valore noto e preciso)
- Resistenza da misurare
- Breadboard e jumper
- Display (opzionale, tipo OLED I2C o LCD 16x2)
- (opzionale) Pulsante per attivare la misura

Principio di funzionamento:

Colleghi la **resistenza incognita** in serie con una **resistenza nota**, e misuri la tensione in mezzo con l'ADC dell'ESP32. Da lì, usando la legge di Ohm, ricavi il valore della resistenza incognita.

Formula:

$$R_x = R_{\text{ref}} \cdot \left(\frac{V_{\text{in}}}{V_{\text{out}}} - 1 \right)$$

Dove:

- R_x
= resistenza incognita
- R_{ref}
= resistenza di riferimento
- V_{in}
= tensione di alimentazione (es. 3.3V)

- V_{out}

= tensione misurata tra le due resistenze (letto via ADC)

🔧 Schema di collegamento:

```
3.3V ----[ R_ref ]-----+----[ R_x ]---- GND
                        |
                        +-----> GPIO ADC (es. GPIO34)
```

🧠 Codice base per ESP32 (Arduino IDE):

```
const int adcPin = 34; // pin ADC
const float R_ref = 10000.0; // 10k ohm
const float Vin = 3.3;

void setup() {
  Serial.begin(115200);
  analogReadResolution(12); // ESP32 ADC a 12 bit: valori da 0 a 4095
}

void loop() {
  int adcValue = analogRead(adcPin);
  float Vout = (adcValue / 4095.0) * Vin;
  float R_unknown = R_ref * ((Vin / Vout) - 1.0);

  Serial.print("Vout: ");
  Serial.print(Vout, 3);
  Serial.print(" V - Resistenza sconosciuta: ");
  Serial.print(R_unknown, 1);
  Serial.println(" ohm");

  delay(1000);
}
```

Note importanti:

- Usa una **resistenza di riferimento precisa** (tolleranza 1% o meglio).
- Evita di misurare **resistenze molto basse** ($< 100\ \Omega$) perché l'ESP32 non è preciso in quell'intervallo.
- Per migliori risultati, puoi fare **medie su più letture ADC**.

Vuoi anche visualizzarlo su display (OLED o LCD)?

Fammi sapere se vuoi aggiungere l'OLED I2C o un display LCD 1602 e ti preparo anche quella parte!

Ti serve aiuto per montare fisicamente il circuito?

You:

che range di misura posso avere

ChatGPT:

Ottima domanda! Il **range di misura** che puoi ottenere con un ohmetro basato su partitore di tensione dipende principalmente da:

2. La **resistenza di riferimento** scelta (R_{ref})
3. La **risoluzione dell'ADC** (sull'ESP32 è di 12 bit \rightarrow 4096 livelli)
4. La **precisione della lettura ADC** (e delle resistenze)
5. L'**intervallo di tensione misurabile** (nel nostro caso 0–3.3V)

Range pratico di misura

Supponiamo di usare una **R_{ref} da 10k Ω** (comune e precisa).

Con questa, in pratica puoi misurare **resistenze da circa 500 Ω fino a 200k Ω** , con una discreta precisione.

Perché non funziona bene fuori da questo range:

- **Resistenze troppo basse ($< 500\Omega$) :**

V_{out} si avvicina troppo a V_{cc} (3.3V), e quindi diventa difficile distinguere i valori ADC.

- **Resistenze troppo alte ($> 200k\Omega$) :**

V_{out} scende troppo vicino a 0V \rightarrow rumore ADC + corrente troppo bassa = letture imprecise.

Esempio (con $R_{ref} = 10k\Omega$ e $V_{in} = 3.3V$):

R_x (ohm)	V_{out} (approx)	ADC Value
100 Ω	~3.27V	~4060
1k Ω	~3.0V	~3720
10k Ω	~1.65V	~2048
100k Ω	~0.3V	~372
500k Ω	~0.06V	~75

Come vedi, sotto 100 Ω o sopra 200–300k Ω il valore letto cambia pochissimo \rightarrow poco affidabile.

ohmetro digitale con ESP32 da simulare in Wokwi , che:

- Usa **4 partitori di tensione** con resistenze di riferimento diverse,
- Seleziona automaticamente il pin ADC migliore in base al valore letto,
- Mostra la resistenza misurata nel **Serial Monitor** .

Specifiche del progetto:

- Microcontrollore: **ESP32 DevKit**
- 4 GPIO impostati come **input ADC** (uno per ciascun partitore)
- Ogni partitore usa una **resistenza di riferimento diversa** : 100 Ω , 1k Ω , 10k Ω , 100k Ω
- Il secondo ramo del partitore ha la **resistenza da misurare** (simulata con un potenziometro su Wokwi)
- Il codice seleziona il **partitore più adatto** in base al valore ADC
- Risultato mostrato via `Serial.println()`

Schema di principio (semplificato):

Hai 4 partitori tipo:

```
3.3V
|
[R_ref] <- valore fisso (es. 100Ω, 1kΩ, etc.)
|
+-----> ADCx (GPIO32, 33, 34, 35)
|
[R_x] <- resistenza da misurare (simulata con potenziometro)
|
GND
```

Nella simulazione colleghi lo **stesso potenziometro** al secondo ramo dei 4 partitori.

Codice completo per ESP32 (Wokwi ready):

```

const float Vin = 3.3;
const int adcPins[] = {32, 33, 34, 35}; // ADC GPIO pins
const float rRefs[] = {100.0, 1000.0, 10000.0, 100000.0}; // in ohm
const int numPartitori = 4;

void setup() {
  Serial.begin(115200);
  analogReadResolution(12); // 0-4095 su ESP32
}

void loop() {
  int bestIndex = -1;
  float bestReading = 0;
  int bestADC = 0;

  for (int i = 0; i < numPartitori; i++) {
    int adcValue = analogRead(adcPins[i]);
    if (adcValue > 300 && adcValue < 3700) { // intervallo valido
      bestIndex = i;
      bestReading = (adcValue / 4095.0) * Vin;
      bestADC = adcValue;
      break; // appena trovi un partitore valido, usalo
    }
  }

  if (bestIndex != -1) {
    float Rref = rRefs[bestIndex];
    float Vout = bestReading;
    float Rx = Rref * ((Vin / Vout) - 1.0);

    Serial.print("Partitore ADC: GPIO");
    Serial.print(adcPins[bestIndex]);
    Serial.print(" | Vout: ");
    Serial.print(Vout, 3);
    Serial.print(" V | ADC: ");
    Serial.print(bestADC);
    Serial.print(" | R_x = ");
    Serial.print(Rx, 1);
    Serial.println(" ohm");
  } else {
    Serial.println("Resistenza fuori range!");
  }
}

```

```
    delay(1000);  
}
```

Schema Wokwi (collegamenti da fare):

Collega 4 partitori in parallelo, tutti alimentati da 3.3V, ognuno con una resistenza diversa in serie, e **lo stesso potenziometro da 10kΩ** come resistenza incognita collegata a GND.

Puoi simulare in Wokwi aggiungendo 4 resistenze fisse (R_ref) e una **linea comune con potenziometro** :

```
3.3V  
|  
[R_ref1] --+-- GPIO32  
[R_ref2] --+-- GPIO33  
[R_ref3] --+-- GPIO34  
[R_ref4] --+-- GPIO35  
      |  
      [ Pote ] → GND
```

Componenti utilizzati:

- **ESP32 DevKit**
- **4 Resistenze di riferimento** : 100Ω, 1kΩ, 10kΩ, 100kΩ
- **Potenziometro da 10kΩ** : Simula la resistenza incognita da misurare

Schema dei collegamenti:

2. Resistenze di riferimento :

- Collega un'estremità di ciascuna resistenza di riferimento (100Ω, 1kΩ, 10kΩ, 100kΩ) al pin **3.3V** dell'ESP32.
- L'altra estremità di ciascuna resistenza va collegata a un pin ADC dedicato dell'ESP32:
 - 100Ω → **GPIO32**
 - 1kΩ → **GPIO33**
 - 10kΩ → **GPIO34**
 - 100kΩ → **GPIO35**

4. Potenziometro (Resistenza incognita) :

- Collega il terminale centrale del potenziometro al nodo comune tra le resistenze di riferimento e i rispettivi pin ADC.
- Un terminale esterno del potenziometro va collegato a **GND** .
- L'altro terminale esterno può essere lasciato scollegato o collegato anch'esso a GND.

Codice ESP32:

Carica il seguente codice nel tuo ESP32 utilizzando l'Arduino IDE o l'editor di codice di Wokwi:


```

const float Vin = 3.3;
const int adcPins[] = {32, 33, 34, 35}; // GPIO pins collegati ai partitori
const float rRefs[] = {100.0, 1000.0, 10000.0, 100000.0}; // Valori delle resistenze di r
const int numPartitori = 4;

void setup() {
  Serial.begin(115200);
  analogReadResolution(12); // Imposta la risoluzione dell'ADC a 12 bit (valori da 0 a 4095)
}

void loop() {
  int bestIndex = -1;
  float bestReading = 0;
  int bestADC = 0;

  // Scansiona tutti i partitori per trovare il più adatto
  for (int i = 0; i < numPartitori; i++) {
    int adcValue = analogRead(adcPins[i]);
    if (adcValue > 300 && adcValue < 3700) { // Se il valore ADC è nel range valido
      bestIndex = i;
      bestReading = (adcValue / 4095.0) * Vin;
      bestADC = adcValue;
      break; // Usa il primo partitore valido trovato
    }
  }

  if (bestIndex != -1) {
    float Rref = rRefs[bestIndex];
    float Vout = bestReading;
    float Rx = Rref * ((Vin / Vout) - 1.0);

    Serial.print("Partitore selezionato: GPIO");
    Serial.print(adcPins[bestIndex]);
    Serial.print(" | Vout: ");
    Serial.print(Vout, 3);
    Serial.print(" V | ADC: ");
    Serial.print(bestADC);
    Serial.print(" | Resistenza misurata: ");
    Serial.print(Rx, 1);
    Serial.println(" ohm");
  } else {
    Serial.println("Resistenza fuori dal range misurabile!");
  }
}

```

```
    delay(1000); // Attende 1 secondo prima della prossima misura  
}
```

Istruzioni per la simulazione su Wokwi:

2. Accedi a [Wokwi](#) e crea un nuovo progetto selezionando "ESP32".
3. Aggiungi al progetto le seguenti componenti:
 - **ESP32 DevKit**
 - **4 Resistenze** con valori di 100Ω, 1kΩ, 10kΩ e 100kΩ
 - **Potenziometro** da 10kΩ
6. Collega le componenti seguendo lo schema dei collegamenti descritto sopra.
7. Carica il codice nell'editor di Wokwi.
8. Avvia la simulazione.
9. Apri il **Serial Monitor** per visualizzare le letture della resistenza misurata.
10. Regola il potenziometro per simulare diverse resistenze e osserva come l'ESP32 seleziona automaticamente il partitore più adatto e calcola il valore della resistenza.