Tecnologie Digitali – AA 2024/25

Scheda di laboratorio n.01

Strumenti software e hardware

1 Introduzione: strumenti di lavoro

Nelle prime settimane di laboratorio famigliarizzerete con gli strumenti sperimentali e di programmazione chiave del laboratorio didattico. Misureremo circuiti semplici e già noti, per concentrarci sugli strumenti useremo durante l'anno. Questi sono distinti in:

- Gli strumenti *hardware* presenti nel laboratorio e/o quelli forniti in comodato per l'approfondimento a casa[1];
- Pacchetti *software* per: *(i)* simulazione circuitake (`TINA`); *(ii)* interfaccia degli strumenti (alternativamente, potremo usare il software nativo `WaveForms` oppure lo scripting `python` in congiunzione con il modulo `tdwf` sviluppato appositamente per il corso); *(iii)* di analisi dati (`MATLAB`).

L'uso di questi strumenti (in particolare quelli software) *non è opzionale* e fa parte integrante del programma didattico del corso. Questi verranno illustrati uno per volta a seguire, insieme a semplici attività di addestramento in laboratorio che vi permetteranno di iniziare a conoscerli.

Prima di cominciare, definiamo alcuni aspetti organizzativi che ci aspettiamo seguiate (altrimenti per noi diventa inutilmente complicato monitorare le attività).

Task 1 Sigla del gruppo e cartella di lavoro. Potete usare il computer disponibile in H1 o un laptop personale. Per il computer in H1, usare il folder dei documenti per

- (oggi) creare una cartella di tavolo adottando una sigla derivata dalle prime tre lettere del cognome dei membri `A` e `B` del tavolo: avete due opzioni fra cui scegliere, ossia `AAABBB` oppure `BBBAAA`;
- (settimana per settimana salvare all'interno della cartella tutti i file di dati, i datasheets, ecc relativi alla settimana in una cartella cartella di lavoro denominata `SS##`, per esempio `SS01` per la scheda in corso;

Alla fine di ogni esperienza sarete tenuti a trasferire questo folder (eccetto al più files particolarmente ingombranti, chiedete se avete dubbi) nello share Teams del vostro tavolo.

CHIARIFICAZIONE. Parlando di upload e condivisione, anticipiamo l'ovvia domanda: che cosa "controlleremo" di quel che consegnate? La risposta è che verificheremo: (1) come avete fatto i codici di acquisizione richiesti; (2) come sono – in forma e contenuto – i pochi grafici sperimentali richiesti verso la fine della scheda; (3) più genericamente il contenuto del vostro *logbook*. In seguito, quando disponibili, sposteremo la nostra attenzione sulle relazioni e il *logbook* sarà soprattutto un riferimento per voi.

[1] Opzionalmente! Il corso è progettato per essere completato in laboratorio, ma gli strumenti forniti vi permetteranno di approfondire le esperienze, se lo ritenete opportuno. Ricordiamo però che il terzo anno include molti altri corsi, che non vanno trascurati.

2 Strumenti hardware

Illustriamo innanzitutto i primi due elementi hardware a disposizione in laboratorio: la breadboard e la scheda di acquisizione.

2.1 Breadboard

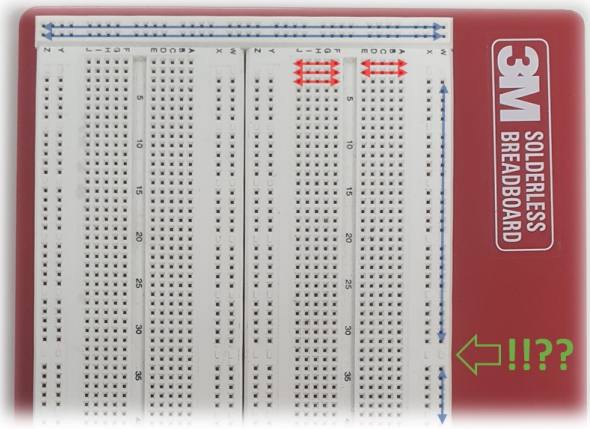

Figura 2.1: *`breadboard` in dotazione.*

I circuiti elettronici che studieremo verranno realizzati usando una `breadboard`[2] (Fig. 2.1), che permette di collegare fra loro diversi componenti elettronici. I contatti di ognuna delle colonne `ABCDE` sono elettricamente connessi fra di loro all'interno della `breadboard`, così come i contatti di ognuna delle colonne `FGHIJ` (frecce rosse). Le frecce blu indicano ulteriori lunghe file di piedini connessi detti *rail* e in genere usati per distribuire tensioni di alimentazione in maniera ordinata.

Attenzione. Alcuni dettagli da ricordare:

- A seconda dei modelli, i *rail* **potrebbero essere interrotti a metà** (grafica in verde in Fig. 2.1), per maggiore flessibilità nel montaggio. Ricordatelo e/o fate qualche verifica con il tester.
- Questo montaggio **non è adatto alle alte frequenze**, per via delle molte capacità parassite (fino a circa 20 pF) fra le molte piste di connessione parallele.
- Questo montaggio **non è adatto alla alte correnti** ed è sconsigliato superare le poche centiaia di milliAmpere, se non si vuole rischiare di sciogliere la plastica.

2.2 Analog Discovery 2

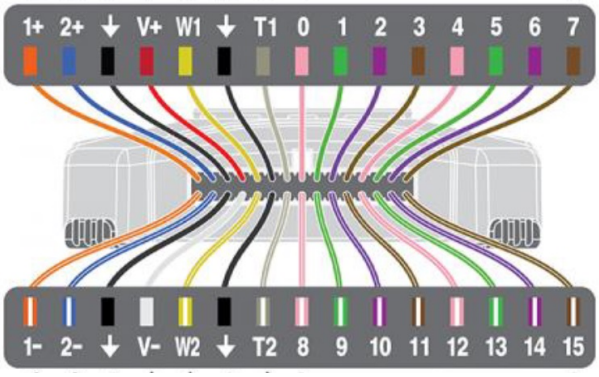

Figura 2.2: *Connettore di `AD2` con relativo codice cromatico per distinguere a vista i segnali.*

Lo strumento principale che useremo sarà la scheda di acquisizione su USB `Analog Discovery 2`. La `Digilent` fornisce una `GUI` (*Graphic User Interface*) abbastanza generale e completa chiamata `WaveForms` che potrà essere utilizzata durante il corso. Tipicamente sarete invitati a controllare in maniera più "diretta" e flessibile l'*hardware*. Vedremo in particolare come sia possibile controllare `Analog Discovery 2` usando `python`Ėsistono vari motivi per cui spingervi a fare questo piccolo "sforzo": in primis, è bene non perdere l'abitudine a programmare, possibilmente con vari linguaggi; in secondo luogo, questo approccio vi darà una flessibilità molto superiore sul livello di automazione che sarete in grado di realizzare nelle vostre misure.

L'interfaccia principale di `Analog Discovery 2` è costituito da un *bundle* di cavi riportato in Fig. 2.2 che collegherete al circuito. Fra le connessioni che ci interessano immediatamente, evidenziamo qui:

- $\times 2$ ***input*** **analogici**, connessi con un doppio `ADC` (*Analog to Digital Converter*) a 14 bit operante fino a 100 milioni di campioni al secondo. Gli ingressi sono differenziali e indicati qui come `1+/-` e `2+/-`, e in seguito chiamati anche `Ch1+/-` e `Ch2+/-`. In assenza della specifica `+/-`, capiterà anche di usare `Ch1` e `Ch2` come sinonimi di `Ch1+` e `Ch2+`, dando per scontato che il terminale negativo sia collegato a terra[3].
- $\times 2$ **due** ***output*** **analogici**, costituiti da un doppio `DAC` (*Digital to Analog Converter*) a 14 bit e operante da -5 a $+5$ V. Nel connettore, queste uscite sono indicate come `W1` e `W2`.

[2] Il termine deriva dal fatto che in passato i componenti elettronici venivano fissati a tavole di legno simili a quelle per tagliare il pane (da B.Dobkin e J.Williams, *Analog Circuit Design*, Elsevier 2013, `https://doi.org/10.1016/C2012-0-00027-0`).

[3] Lasciare un contatto fottante in un ingresso differenziale non è una buona idea e sebbene probabilmente si misurerà "qualcosa", il risultato sarà piuttosto caotico. Cerchiamo quindi di ricordarci che **i due cavi `+/-` vanno sempre entrambi collegati**.

- un sistema di alimentazione duale, costituito dalle uscite `V+/V-` e dal riferimento di terrra `GND`[4]. Queste alimentazioni saranno utili per i circuiti attivi e possono arrivare fino a $\pm 5\,\mathrm{V}$.

Lo strumento ha varie altre connessioni: alcune non verranno mai usate, altre verranno illustrate via via quando necessario. Per (le molte) ulteriori specifiche, rimandiamo ai manuali forniti su piattaforma Teams.

2.3 Altri strumenti

In un qualsiasi esperimento vero uno sperimentatore deve quasi sempre confrontarsi con una varietà di strumenti, che possono avere natura simile o magari diversa, eventuali protocolli e metodi di comunicazione diversi, formati diversi e così via. All'occorrenza sulla vostra postazione avrete anche a disposizione: (i) un multimetro digitale; (ii) un alimentatore; (iii) un generatore di funzioni convenzionale; (iv) in aula H1, il PC sono anche collegati a una scheda multifunzione[5] detta `DAQ board` o più semplicemente `DAQ` (che sta per *Data AcQuisition*), con una contattiera visibile in Fig. 2.3. La funzione di ogni connettore può essere dedotta dai *datasheet* che trovate su Teams.

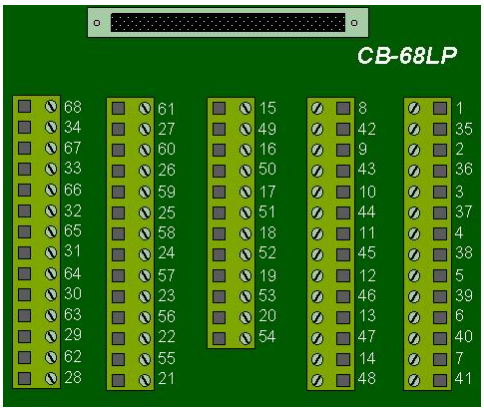

Figura 2.3: *DAQ connection board*

Le schede di acquisizione montate nei PC in aula H1 sono più lente di `Analog Discovery 2`, ma dispongono di ben **sedici canali di input analogici**, configurabili anche come otto coppie differenziali, con cui è possibile digitalizzare le tensioni a 16 bit. Internamente, questo avviene grazie a un `ADC` multiplexato[6]. Sono inoltre presenti **due canali di output analogici** con cui si può generare tensioni e che sono connessi internamente a dei `DAC`. I canali di più frequente utilizzo sono collegati a dei connettori coassiali a baionetta detti BNC:

- `CB29`, connesso al riferimento di terra
- `CB68`, connesso al primo ingresso `Ch1`, o analog input 0
- `CB33`, connesso a `Ch2`, o analog input 1
- `CB22`, connesso alla prima uscita `W1`, o analog output 0.

Per il significato degli altri canali, si prega di consultare il datasheet della scheda di acquisizione, che trovate nello share della settimana.

3 Strumenti software

Nel seguito illustriamo brevemente i passaggi chiave che sono necessari per controllare il sistema di acquisizione tramite dei semplici notebook `python`. Sebbene questo richieda uno sforzo un minimo superiore ad aprire `WaveForms`, questo approccio sarà quello che permetterà il migliore livello di automazione e controllo nei processi di misura, per esempio – caso davvero banale – renderà molto semplice ripetete la stessa misura più volte al variare di un qualche parametro.

3.1 Il modulo `tdwf`

Il pacchetto software fornito dalla `Digilent` include non solo `WaveForms` ma anche una così detta `SDK` (*Software Development Kit*), che non è altro che una libreria di funzioni a basso livello che permette di interagire con l'hardware. Lo stesso `WaveForms` sfrutta la `SDK` per comunicare con `Analog Discovery 2`, e in maniera simile ora vedremo come usarla per controllare l'*hardware* da `python`.

Sebbene sia possibile usare direttamente le funzioni della `SDK`[7], questo richiederebbe troppi dettagli di programmazione che rischierebbero di distrarre troppo l'attenzione dal fuoco del corso, che è l'attività sperimentale di laboratorio. Per ovviare a questo problema useremo un semplice modulo chiamato `tdwf`, che è stato scritto per questo corso e sostanzialmente realizza un *wrapper* della `SDK`.

[4] Sempre corrispondente ai cavi di colore nero sul connettore: ce ne sono vari e tutti collegati fra di loro.

[5] I PC dell'aula H1 sono equipaggiati con schede della `National Instruments` tipo PCI-6024E, PCI-6221 o PCIe-6321

[6] ...ossia si tratta di un unico convertitore connesso – tramite una qualche circuiteria riconfigurabile a piacere – alle diverse uscite analogiche. Questo permette all'`ADC` di misurare in sequenza diversi segnali, ovviamente a scapito della velocità.

[7] Si faccia riferimento al sito della `Digilent` e al manuale della `SDK` che è possibile scaricare.

Task 2 **Verifica/installazione del modulo `tdwf`.** Il modulo è già installato sui computer del laboratorio. Nel caso fosse necessario, si considerino i seguenti comandi da eseguire sulla shell `python`

```
pip install tdwflib.zip  # installa/aggiorna il modulo tdwf
pip show tdwf            # verifica installazione
pip uninstall tdwf       # (casomai servisse) disintalla il modulo
```

3.2 Acquisizione dati con `python`

In questa parte dell'esperienza impareremo a programmare l'acquisizione di semplici segnali analogici da `Analog Discovery 2` tramite un codice `python` che costruiremo nel seguito. Lo script richiederà i seguenti passaggi, in cui illustreremo passo-passo quello che il modulo `tdwf` rende possibile:

1. La connessione con `Analog Discovery 2`, tramite l'inizializzazione di un oggetto di classe `tdfw.AD2`;
2. La creazione e configurazione di un oscilloscopio tramite la classe `tdwf.Scope`;
3. L'acquisizione di un segnale analogico;
4. La visualizzazione e salvataggio del misura.

I primi due punti sono illustrati a seguire.

Collegamento e configurazione generale dell'oscilloscopio. Qui illustriamo i passaggi chiave necessari, commentando in seguito le eventuali possibilità aggiuntive. La sintassi è piuttosto semplice:

```
import tdwf                    # importiamo il modulo...

ad2 = tdwf.AD2()               # connessione all'hardware
scope = tdwf.Scope(ad2.hdwf)   # inizializzazione oscilloscopio
scope.fs = 1e6                 # => frequenza di sampling impostata a 1MSa/s
scope.npt = 1000               # => acquisizione impostata a 1000 punti
```

Note. La corretta inizializzazione di `ad2` è segnalata da messaggi sulla shell `python`. Se lo strumento è invece occupato, per esempio perché è usato da `WaveForms`, ci sarà un messaggio di avviso e lo script terminerà prematuramente.

Qualche dettaglio tecnico in più:

- Nell'inizializzazione `tdwf.AD2()` è possibile definire anche la configurazione `iconfig`, che di default è uguale a 1. Questa specifica diversi modi di allocare le risorse limitate dell'hardware: per esempio, mentre nella configurazione di default il *buffer* di misura ha una lunghezza massima di 8192 valori, usando `tdwf.AD2(iconfig=3)` avremmo la configurazione #3, in cui il buffer può arrivare a 16384 valori, a scapito delle funzionalità digitali. Le configurazioni sono visibili nella schermata iniziale di `WaveForms`.
- La proprietà `ad2.hdwf` merita un commento, dato che va passata nell'inizializzazione dell'oscilloscopio (e poi anche in altri casi). Si tratta semplicemente di un numero identificativo dello strumento (detto anche *handle*, da cui la *h* iniziale) assegnato dalla libreria quando connette PC e `Analog Discovery 2`.
- La configurazione `scope.fs = 1e6` merita anche un commento. La frequenza di campionamento di `Analog Discovery 2` può solo essere una *frazione* di 100 MSa/s, come 100, 50, 33.333... o 25 MSa/s. Nel caso in cui venga impostata una frequenza "proibita" come 60 MSa/s, il sistema arrotonderà al valore "ammesso" più vicino, ossia 50 MSa/s nello specifico caso citato. Il valore effettivamente impostato può essere verificato andando a rileggere il valore di `scope.fs`, per esempio con un `print(scope.fs)`.
- La configurazione di `scope.npt` in genere sarà entro il massimo numero di punti disponibile per il *buffer* di misura (8192 nella configurazione di default). Come vedremo, superare questo limite è possibile ma il PC dovrà continuamente "scaricare" i dati da `Analog Discovery 2` per mantenere libero il *buffer* di misura, con una sostanziale riduzione della massima frequenza di *sampling* praticamente utilizzabile[8].

[8] Indicativamente, sarà sconsigliabile andare sopra 1 MSa/s, pena una probabile corruzione della misura in quanto il computer non farà in tempo a scaricare i dati da `Analog Discovery 2` prima che vengano sovrascritti da nuove misure.

Nel codice a seguire, finiamo di configurare l'oscilloscopio ed effettuiamo una misura

Configurazione dei canali e acquisizione. Ognuno dei due canali `Ch1` e `Ch2` supporta varie configurazioni che illustriamo nel breve esempio a seguire. Come prima, illustriamo i passaggi chiave necessari, commentando in seguito le eventuali possibilità aggiuntive.

```
scope.ch1.rng = 5       # range Ch1 su [-2.5,+2.5]
scope.ch2.rng = 50      # range Ch2 su [-25,+25]
scope.ch2.avg = True    # attiva media su Ch2

scope.sample()          # Avvio acquisizione
print(scope.ch1.vals)   # stampa su shell
```

Note. `Analog Discovery 2` supporta *due* intervalli di misura: 5 o 50 V. Il sistema accetta in realtà qualsiasi valore, ma assegna il minimo intervallo compatibile. La proprietà `scope.ch2.avg` attiva la media *hardware*, che può essere molto importante quando un campionamento è inferiore a 100 MSa/s. Si supponga di lavorare a 20 MSa/s: di default `Analog Discovery 2` acquisisce *sempre* a 100 MSa/s ma registra solo *un valore ogni 5*, ignorando gli altri. Quando la media è attivata, il sistema considera tutti e 5 i valori misurati per ogni periodo di acquisizione, e ne calcola la medi.

Anche in questo caso aggiungiamo qualche commento esplicativo:

- Ogni canale ha *offset*, per esempio regolabile con `scope.ch1.offs = 3.0`. Questa configurazione trasla l'intervallo di misura a `[0.5,5.5]`, così che anche se il segnale da misurare oscilla poco attorno a 3 V, è possibile misurarlo senza passare al range meno preciso di 50 V.
- L'istruzione `scope.sample()` non fornisce un risultato diretto, ma semmai aggiorna `scope.ch1.vals`, un *array numpy* automaticamente aggiornato alla fine del misura.

Per finire, probabilmente vogliamo plottare o salvare i dati appena acquisiti.

Visualizzazione e salvataggio della misura. Per fare un grafico possiamo usare il vettore dei tempi (calcolato) `scope.time.vals`:

```
import matplotlib.pyplot as plt

plt.plot(scope.time.vals, scope.ch1.vals)
plt.xlabel("Tempo [s]")
plt.ylabel("Ch1 [V]")
```

Per salvare la misura in `python` esistono una moltitidine di metodi diversi, qui mostriamo come farlo con `numpy`, che è già importato come `np` dal modulo `tdwf`.

```
data = np.column_stack((scope.time.vals,scope.ch1.vals,scope.ch2.vals))
np.savetxt("nomefile.txt", data, delimiter="\t")
```

Nel caso voleste essere più descrittivi e aggiungere per esempio un `header`, considerate che è possible avere il *timestamp* della misura usando la variabile `scope.time.t0`, che viene aggiornata ad ogni acquisizione.

Task 3 Usare gli *snippet* di codice forniti per costruire un notebook `my-adc.ipynb` che combini i vari passaggi. Eseguire i seguenti test sperimentali:

- cortocircuitare `Ch1+` e `Ch1-` $\implies$ ci aspettiamo di misurare dei valori prossimi a zero;
- lasciare `Ch1+` flottante e toccarlo con un dito $\implies$ ci aspettiamo di misurare un rumore a 50 Hz.

Si noti come il segnale misurato sia discretizzato: quanto è la risoluzione approssimativa (basta una cifra significativa) della misura? Come dipende dalla scelta dell'intervallo di misura impostato da `rng`? Dipende dall'impostazione usata per `avg`? Questo valore è consistente con l'informazione data secondo cui questo `ADC` fa delle misure a 14 bit? Verificare che queste specifiche vengono indicate nel *datasheet* di `Analog Discovery 2`, in termini di `V/div` scelto nel programma `WaveForms`.

3.3 Configurare il generatore di funzioni con `python`

In maniera simile a quanto fatto per l'acquisizione, rivediamo i passaggi di base richiesti per controllare il canale `W1` del generatore di funzione, fra cui:

1. La creazione di un generatore di funzione tramite la classe `tdwf.WaveGen` e la sua configurazione;
2. Il controllo (avvio/interruzione) delle uscite analogiche.

Gli *snippet* a seguire mostrano alcuni semplici codici per configurare l'uscita analogica `W1` di `Analog Discovery 2`. Per ora ignoreremo la seconda uscita `W2`.

Configurazione del generatore di funzioni - Opzione 1. Il modulo `tdwf` implementa un primo metodo chiamato `.config()` che permette di fare una configurazione cumulativa del canale di *output*:

```
wgen = tdwf.WaveGen(ad2.hdwf)              # inizializzazione generatore
wgen.w1.config(offs=1, func=tdwf.funcDC)   # => imposta valore DC a 1V
wgen.w1.start()                            # => avvia riproduzione
```

Note. Nel metodo `.config()` il parametro `offs` regola un *offset* generale del `DAC`, che in questo caso sarà l'intero segnale. Il parametro `func` permette di scegliere il tipo di forma d'onda, e quando impostato alla costante `tdwf.funcDC` corrispondente ad una forma d'onda nulla.

Configurazione del generatore di funzioni - Opzione 2. Esattamente lo stesso risultato può essere ottenuto anche andando a cambiare le proprietà del canale una per volta:

```
wgen = tdwf.WaveGen(ad2.hdwf)  # inizializzazione generatore
wgen.w1.offs = 1               # => imposta offset a 1V
wgen.w1.func = tdwf.funcDC     # => imposta forma d'onda costante
wgen.w1.start()                # => avvia riproduzione
```

La stessa procedura può essere usata per controllare la seconda uscita, andando ad agire su `wgen.w2`.

La configurazione completa della forma d'onda del canale di output include le seguenti proprietà:

Proprietà	Descrizione	Default
`w#.offs`	*offset* della forma d'onda, ossia una costante additiva	0 V
`w#.ampl`	ampiezza (di picco) della forma d'onda	1 V
`w#.freq`	frequenza della forma d'onda	1 kHz
`w#.phi`	fase della forma d'onda	0 deg
`w#.duty`	simmetria della forma d'onda (o anche *duty cycle*)	50%
`w#.func`	tipo di fonda d'onda (vedere sotto)	`tdwf.funcSine`
`w#.data`	array di valori fra -1 e $+1$ che descrive una forma d'onda arbitraria	`[]`

L'argomento `func` controlla il tipo di forma d'onda e merita una discussione a parte. In questo caso è possibile impostare varie costanti predefinite, fra queste le più importanti sono:

Costante	Significato
`tdwf.funcDC`	Onda nulla $\implies$ l'output costante e uguale all'*offset*
`tdwf.funcSine`	Sinusoide (questa è la configurazione di default)
`tdwf.funcSquare`	Onda quadra, con *duty* controllato dal valore di `duty`
`tdwf.funcTriangle`	Onda triangolare, che con `duty` pul diventare un dente di sega
`tdwf.funcCustom`	Onda arbitraria in base al contento dell'array `w#.data`

Task 4 Funziona tutto? Inserire il controllo del canale di uscita `W1` in un nuovo notebook `my-dac+adc.ipynb`. Collegare `Ch1+` a `W1` e `Ch1-` a una delle terre. Impostare qualche forma d'onda o un voltaggio `DC` e verificare se `Ch1` mostra quanto attesto. Usare un multimetro per verificare l'accuratezza dei voltaggi `DC`. Valutare la consistenza dei risultati sulla base delle accuratezze dichiarate nei manuali/datasheet.

4 Primo semplice esperimento con la `breadboard`

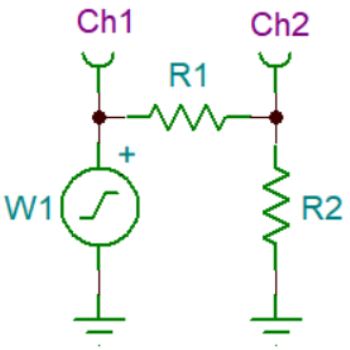

Figura 4.1: *Partitore.*

Per concludere, faremo alcune misure sul circuito di Fig. 4.1, ricordandoci che l'ovvio obiettivo di una qualsiasi misura è: (1) **stimare dei parametri**, ossia produrre dei numeri, e (2) capire **quanto di possiamo fidare** delle stime ottenute... e non tanto (3) verificare che "tutto torna", cosa sempre gradita e auspicabile ma anche completamente inutile se svincolata dal resto. Il numero da produrre qui sarà il rapporto fra le due resistenze, che è ovviamente connesso ai voltaggi in Fig.4.1 da

$$V_{\text{OUT}} = V_{\text{IN}} \frac{R_2}{R_1 + R_2}. \tag{4.1}$$

Il circuito è evidentemente banale e poco misterioso, considerate tutto questo soprattutto come una scusa per famigliarizzare con l'uso e le proprietà dei sistemi hardware (scheda `DAQ` e `breadboard`) e software (`LabVIEW`) per l'acquisizione dati, e con le procedure di salvataggio, caricamento e analisi dei dati con `MATLAB`. Conoscere gli strumenti che si usano è infatti fondamentale e in seguito vi sarà probabilmente utile sapere quanto vale o come ricavare, per esempio, la risoluzione, il livello di rumore, o in genere le caratteristiche della `DAQ` che usate nelle misure.

Task 5 Montare sulla `breadboard` un partitore con rapporto di divisione non molto minore di 1, scegliendo resistenze con un valore nominale dell'ordine dei kΩ. Collegare l'ingresso a `W1` e fare alcune acquisizioni su `IN` e su `OUT` in Fig.4.1, usando il vostro `my-dac+adc.ipynb`. L'obiettivo è esplorare il tipo di dato prodotto dalla `DAQ` ma per fissare le idee suggeriamo di:

- usare sampling a 8192 punti e 1 MSa/s;
- verificare la dipendenza da `scope.ch1.rng`, con un voltaggio di `W1` a piacere.

Inserire nel *logbook* il codice di analisi `MATLAB`, attingendo liberamente dal *Livescript* di esempio `Tutorial_FilePlotFit_Basic.mlx`). Infine, generare due grafici finali `DistribuzioneIN.pdf` e `DistribuzioneOUT.pdf` che descrivano/analizzino la distribuzione dei dati.

Arrivati a questo punto, sarebbe utile poter usare la `DAQ` in maniera più completa, per esempio acquisendo due canali e automatizzando un po' l'acquisizione. Quando arrivate a questo punto, vi forniremo un notebook `python` preconfezionato chiamato `sweepbias.ipynb`, per mostrarvi qualche esempio pratico di come confezionare un notebook di misura. In seguito ci aspettiamo che sviluppiate un minimo di indipendenza nel modificarli in base alle vostre esigenze.

Task 6 Usare il codice fornito per misurare la risposta del partitore, scegliendo gli intervalli di voltaggio e i parametri che volete. Analizzare il risultato con `MATLAB` producendo un grafico finale con fit `Partitore.pdf`. Confrontare il rapporto R_1/R_2 misurato con le tolleranze attese per i componenti usati. L'errore statistico delle misure mediate può essere o calcolato o magari misurato (per esempio facendo una misura a `W1` e guardandone la distribuzione).