UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA ELETTRONICA

"SVILUPPO DI CODICE PER IL CONTROLLO DI STRUMENTI DI MISURA MEDIANTE IL TOOLBOX INTRSTRUMENT CONTROL DI MATLAB"

Relatore: Prof./Dott Caldognetto Tommaso

Laureando: Pizzolotto Andrea

Correlatore: Prof./Dott Stellini Marco

ANNO ACCADEMICO 2022 – 2023

Data di laurea 25 settembre 2023

Sommario

1 Introduzione: Motivazioni e obbiettivi

Matlab è un ambiente per il calcolo numerico utilizzato da milioni di persone nell'industria e nelle università per via dei suoi numerosi strumenti a supporto dei più disparati campi di studio applicativi.
Collegare quindi uno strumento di misura a Matlab consente di produrre ed elaborare i dati all'interno dello stesso ambiente software.
Un altro aspetto da non trascurare, e forse il più attraente, è la possibilità, per esempio, di effettuare centinaia di misure in modalità completamente automatica senza che un operatore debba effettuarle manualmente velocizzando così il processo e riducendo la possibilità di errore.

L'obbiettivo prefissato è quindi quello di accumulare le conoscenze necessarie (e fornire alcuni esempi concreti) per l'uso di tali strumenti.
Le prove sono state effettuate su due strumenti: un multimetro digitale (Keysight Ag34461A) e un oscilloscopio (Keysight DSOX1102G), ma ogni concetto e modalità di approccio alla scrittura del codice può essere esteso ad un qualsiasi altro strumento compatibile.

2 Passaggi preliminari

Prima di poter connettere gli strumenti al computer è necessario compiere alcuni passi di preparazione, più precisamente sarà necessario installare degli Add-Ons da Matlab e i driver necessari per gli strumenti che si desidera utilizzare. Alcune delle procedure successivamente descritte sono possibili solo su ambiente Windows.

2.1 Add-Ons di Matlab necessari

Gli Add-Ons necessari sono tutti installabili tramite l'Add-On Explorer di Matlab e sono:

1- Instrument Control Toolbox:

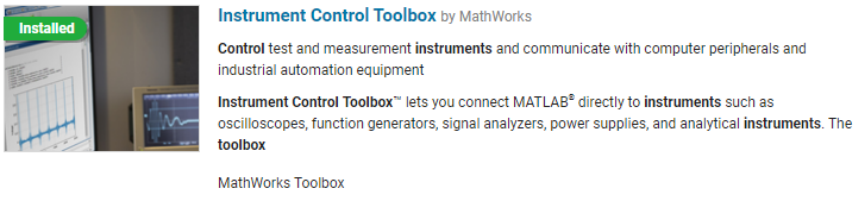


Figura 1 Pacchetto Instrument control toolbox visto dal menù Add-Ons di Matlab

2- Instrument Control Toolbox Support Package for National Instruments™ VISA and ICP Interfaces:

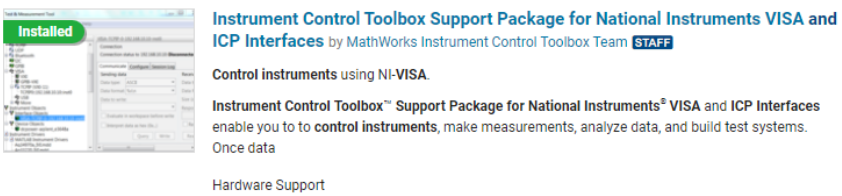


Figura 2 Pacchetto di supporto per interfacce National Instrument per l'Instrument control toolbox visto dal menù Add-Ons di Matlab

3- Instrument Control Toolbox Support Package for IVI and VXI*plug&play* Drivers:
Questo pacchetto è supportato solo su computer con sistema operativo Windows.
Contiene anche i driver Matlab degli strumenti supportati.

Instrument Control Toolbox Support Package for IVI and VXIplug&play Drivers by
MathWorks Instrument Control Toolbox Team STAFF

Communicate with **instruments** that use **IVI** or **VXIplug&play drivers**

Instrument Control Toolbox™ Support Package for IVI and **VXIplug&play Drivers** provides **support for** popular **instruments** that are delivered with **IVI**-C or **VXIplug&play drivers**. When this **support**

MathWorks Optional Feature

Figura 3 Pacchetto di supporto dei driver ivi per l'Instrument control toolbox visto dal menù Add-Ons di Matlab

2.2 Download e installazione driver strumenti

Prima di continuare bisogna accertarsi che i driver IVI del dispositivo che si intende utilizzare siano supportati dall' Instrument Control Toolbox Support Package for IVI and VXI*plug&play* Drivers verificando che sia presente nella lista di strumenti consultabile online al link:
https://it.mathworks.com/help/instrument/supported-ivi-drivers.html

Si può quindi procedere con l'installazione dei driver IVI degli strumenti che sono reperibili dal sito del produttore, sono riportati come esempio i driver necessari per il controllo dei dispositivi su cui sono state effettuate le prove.

KEYSIGHT InfiniiVision X-Series Oscilloscope IVI and MATLAB Instrument Drivers:
https://www.keysight.com/us/en/lib/software-detail/driver/infiniivision-xseries-oscilloscope-ivi-and-matlab-instrument-drivers-2019021.html

KEYSIGHT 3446x Digital Multimeters IVI and MATLAB Instrument Drivers:
https://www.keysight.com/us/en/lib/software-detail/driver/3446x-digital-multimeters-ivi-and-matlab-instrument-drivers-2352538.html

Nel caso lo strumento non sia tra quelli della lista dovrebbe comunque poter essere usato nel caso siano disponibili i driver IVI per lo strumento creando una "Driver Session" con iviconfigurationstore.
Si rimanda alla documentazione per un ulteriore approfondimento.
Ividev:
https://it.mathworks.com/help/instrument/ividev.html?searchHighlight=ividev&s_tid=srchtitle_support_results_1_ividev
e iviconfigurationstore:
https://it.mathworks.com/help/instrument/iviconfigurationstore.html?searchHighlight=iviconfigurationstore&s_tid=srchtitle_support_results_1_iviconfigurationstore .

2.3 Verifica corretta installazione dei driver

Lanciando sulla *Command Window* di Matlab il comando "`ividriverlist`" verrà restituito come output una tabella con i driver installati e i dispositivi supportati.

```
Command Window
>> ividriverlist

ans =

  15×4 table

          VendorDriver          MATLABDriver          IVIClass                                                                 SupportedModels
        __________________    __________________    __________________    ___________________________________________________________________________

     1  "Ag3446x"             "Ag3446x"             "IVIDmm"              {["34460A"    "34461A"    "34465A"    "34470A"
     2  "AgInfiniiVision"     "AgInfiniiVision"     "IVIScope"            {["C7302"    "C7302C"    "C7304"    "C7304C"    "C7312"    "C7312C"    "C7314"
     3  "IviACPwr"            "IviACPwr"            "IVIACPwr"            {[""
     4  "IviCounter"          "IviCounter"          "IVICounter"          {[""
     5  "IviDCPwr"            "IviDCPwr"            "IVIDCPwr"            {[""
     6  "IviDigitizer"        "IviDigitizer"        "IVIDigitizer"        {[""
     7  "IviDmm"              "IviDmm"              "IVIDmm"              {[""
     8  "IviDownconverter"    "IviDownconverter"    "IVIDownconverter"    {[""
     9  "IviFgen"             "IviFgen"             "IVIFgen"             {[""
    10  "IviPwrMeter"         "IviPwrMeter"         "IVIPwrMeter"         {[""
    11  "IviRfSigGen"         "IviRfSigGen"         "IVIRfSigGen"         {[""
    12  "IviScope"            "IviScope"            "IVIScope"            {[""
    13  "IviSpecAn"           "IviSpecAn"           "IVISpecAn"           {[""
    14  "IviSwtch"            "IviSwtch"            "IVISwtch"            {[""
    15  "IviUpconverter"      "IviUpconverter"      "IVIUpconverter"      {[""

fx >>
```

Figura 4 Lista driver installati

Se tutti i passaggi sono stati eseguiti correttamente si dovrà trovare tra i dispositivi supportati (sezione *SupportedModels* della tabella) il dispositivo che si intendeva utilizzare.

3 Avvio comunicazione con lo strumento

3.1 Visualizzazione lista dispositivi collegati

La funzione `ividevlist("Timeout",5)` (oppure solo `ividevlist()`) cerca dispositivi collegati al computer, dando la possibilità di impostare un tempo di attesa, e restituisce una tabella con l'elenco di dispositivi rilevati con alcune informazioni relative ad essi (nome dei pacchetti dei driver, il nome della risorsa il modello dello strumento e il suo numero di serie).

Un esempio di output può essere:

```
>> devlist = ividevlist("Timeout",5)
Warning: Timeout expired while searching for available resources.

devlist =

  1×5 table

        MATLABDriver                  ResourceName                      VendorDriver         Model        SerialNum
      _________________    ___________________________________________    _________________    ____________    _________

   1  "AgInfiniiVision"    "USB0::0x2A8D::0x1797::CN57136331::0::INSTR"    "AgInfiniiVision"    "DSO-X 1102G"    "CN571363

>>
```

Figura 5 Esempio output ividevlist

Le informazioni presenti nella tabella, come il nome del modello e il nome del pacchetto di driver, ci permettono di distinguere tra loro i vari dispositivi collegati.

3.2 Creazione oggetto "ividev"

Per controllare il dispositivo è necessario creare un oggetto "ividev" in Matlab con il costruttore: `ividev("MATLABDriver", "ResourceName")` che ha come parametri espliciti la stringa contenuta nel campo "`MATLABDriver`" e quella nel campo "`ResourceName`".

Esempi:

```
dso =
ividev("AgInfiniiVision","USB0::0x2A8D::0x1797::CN7136331::0::INSTR")
```
(esempio oscilloscopio Keysight InfiniiVision)

```
dmm = ividev("Ag3446x", "USB0::0x2A8D::0x1301::MY53224712::0::INSTR")
```
(esempio multimetro digitale Keysight serie Ag3446x)

C'è anche la possibilità di simulare la connessione con uno strumento, qual ora si volesse provare del codice (con alcune limitazioni), passando al costruttore come secondo parametro una stringa vuota e come terzo una variabile booleana "true" come negli esempi seguenti:

```
dso = ividev("AgInfiniiVision", "", Simulate = true)
```
(esempio oscilloscopio Keysight InfiniiVision)

```
dmm = ividev("Ag3446x", "", Simulate = true)
```
(esempio multimetro digitale Keysight serie Ag3446x)

Se l'operazione dovesse andare a buon fine viene restituita una sequenza di informazioni ulteriori sul dispositivo (per visualizzare l'output delle funzioni sulla *Command Window* bisogna omettere il ";" a fine riga) e la possibilità di visualizzare tutte le funzioni utilizzabili con lo stesso cliccando su "function" (sottolineato in rosso in Figura 2). È possibile poi utilizzare lo strumento di "help" per comprendere il funzionamento della funzione che si vuole utilizzare, in alcuni casi la documentazione fornita potrebbe non essere sufficiente per una comprensione completa e sarà quindi necessario effettuare delle prove.

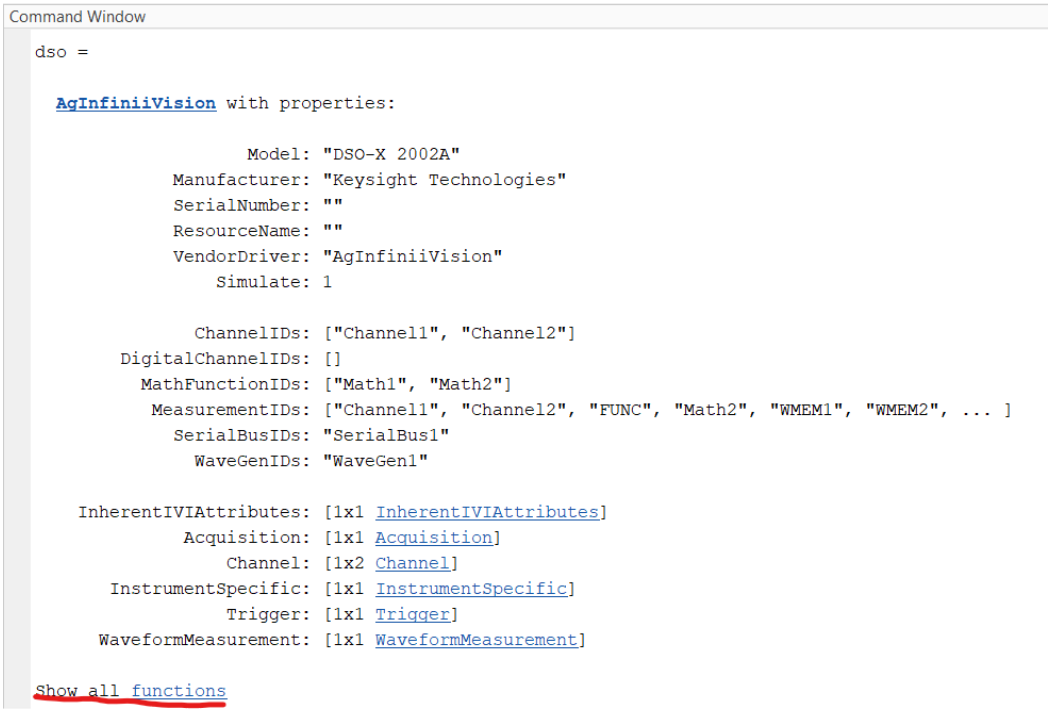


Figura 6 Esempio output costruttore ividev (strumento simulato)

La connessione è ora stabilita ed è possibile controllare lo strumento agendo sull'oggetto creato.
L'oggetto ividev è composto da variabili e "sotto oggetti" pubblici. Alcune di tali variabili se consultate forniscono informazioni sulle proprietà dello strumento o sul suo stato, altre se modificate ne permettono la configurazione. Una volta creato un oggetto in Matlab è possibile visualizzarne le variabili (doppio click sull'oggetto in questione nel *Workspace*). I

nomi delle variabili sono spesso esaustivi permettendo così di capire a che proprietà si riferiscano.

4 Introduzione alle modalità di controllo

Ci sono due modalità principali per il controllo: la prima che consiste nell'utilizzo delle funzioni preconfezionate disponibili e la seconda che prevede la modifica diretta delle variabili (pubbliche) dell'oggetto *ividev*.

Ogni strumento che condivide lo stesso pacchetto di driver ha a disposizione lo stesso set di funzioni e una struttura dell'oggetto *ividev*, creato alla connessione, uguale.

Nei successivi paragrafi verrà fornita una guida, come esempio, per il controllo di due serie di strumenti:

- Oscilloscopio Keysight serie InfiniiVision;
- Multimetro digitale Keysight Ag3446x.

Verrà esposto l'uso di alcune funzioni e di come modificare le variabili allo scopo di poter controllare gli strumenti in analisi.
Per il controllo dell'oscilloscopio è stata utilizzata prevalentemente la prima mentre per il multimetro la seconda.

5 Controllo oscilloscopio Keysight InfiniiVision

In seguito con "dso" si farà riferimento all'oggetto relativo all'oscilloscopio creato con le modalità precedentemente discusse. L'oggetto deve essere passato sempre come parametro esplicito alle funzioni.

Molte delle impostazioni effettuate in seguito con azione diretta sulle variabili possono essere effettuate anche con funzioni, tuttavia, con questo strumento, la prima modalità risulta spesso più comoda concedendo più libertà di controllo rimanendo comunque di semplice realizzazione.

5.1 Variabili

Effettuando un doppio click sull'oggetto (dopo la sua creazione), nel workspace, che si vuole analizzare si apre una finestra con l'elenco delle variabili e "sotto-oggetti" del quale è composto. Per far riferimento ad una variabile è necessario usare la sintassi "`oggetto.variabile`". Cliccando col tasto destro è anche possibile copiare la variabile con la giusta sintassi per poi copiarla nello script.

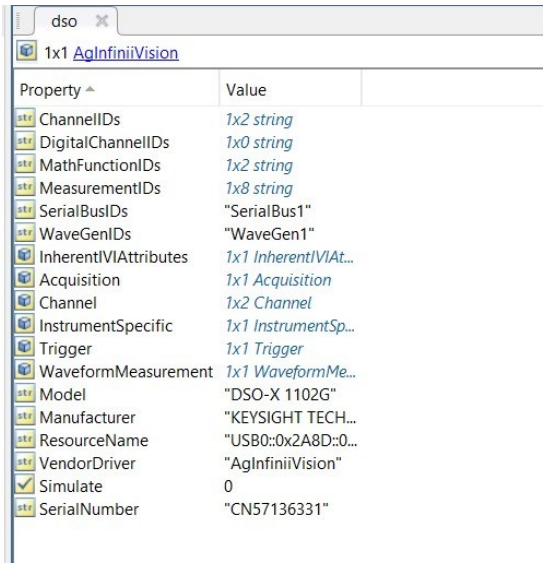


Figura 7 Variabili e "sotto oggetti" dell'oggetto ividev per oscilloscopi Keysight InfiniiVision (strumento fisico)

`dso.ChannelIDs` è un vettore di stringhe con gli id dei canali che serviranno per poter fare riferimento agli stessi canali il cui contenuto in questo caso è: (`"Channel1", "Channel2"`).

`dso.Channel` è un vettore riga di oggetti a cui ogni elemento fa riferimento ad un canale dell'oscilloscopio, nel caso di strumento con due canali come nella Figura 3 si ha canale 1: `dso.Channel(1,1)` e canale 2: `dso.Channel(1,2)` o alternativamente usando gli ID dei canali `dso.Channel("Channel1")` e `dso.Channel("Channel2")`.

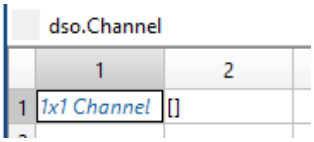


Figura 8 Vettore Channel (strumento simulato)

5.2 Variabili del canale

Effettuando un doppio click sull'oggetto Channel relativo al canale di cui si vuole visualizzarne le variabili si apre una finestra come in figura.

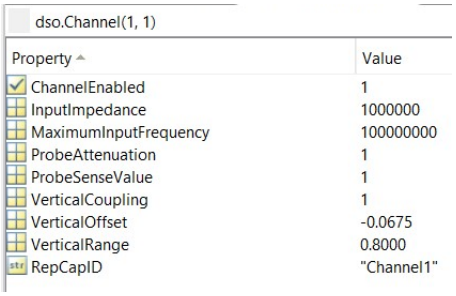


Figura 9 Variabili di canale (strumento fisico)

Modificare le variabili relative ai canali consente di impostarne le proprietà.

Tra cui (esempi per il canale 1 ma è lo stesso per ogni altro canale):

Enable canale: `dso.Channel(1, 1).ChannelEnabled` , variabile booleana a cui si può assegnare *true* (o 1) per abilitare il canale oppure *false* (o 0) per disabilitarlo.

Attenuazione sonda: `dso.Channel(1,1).ProbeAttenuation` variabile relativa all'attenuazione della sonda collegata al canale a cui semplicemente si deve assegnare il valore di attenuazione della sonda (per esempio 10).

Accoppiamento di canale AC/DC: `dso.Channel(1, 1).VerticalCoupling` , variabile relativa all'accoppiamento del canale.

Accoppiamento AC: `dso.Channel(1, 1).VerticalCoupling = "AC";`
o equivalentemente `dso.Channel(1, 1).VerticalCoupling = 0;`

Accoppiamento DC: `dso.Channel(1, 1).VerticalCoupling = "DC";`
o equivalentemente `dso.Channel(1, 1).VerticalCoupling = 1;`

Offset: `dso.Channel(1, 1).VerticalOffset` è la variabile che indica il valore dell'offset del canale in volt.

Range verticale: `dso.Channel(1, 1).VerticalRange` è la variabile che rappresenta il valore in volt del range di misura.

5.3 Acquisition

Acquisition è un oggetto che raggruppa le variabili inerenti all'acquisizione.

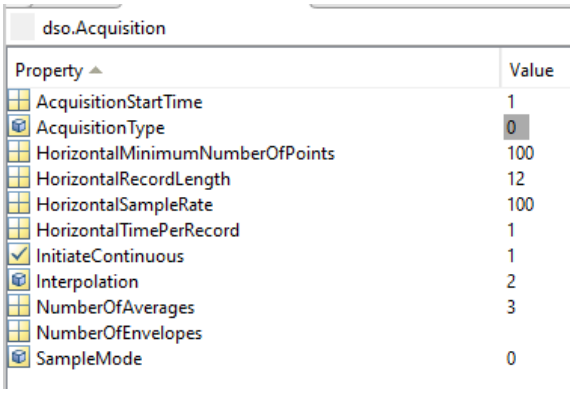


Figura 10 Variabili Acquisition (strumento simulato)

Tipo di acquisizione: è possibile impostare il tipo di acquisizione modificando il valore di dso.Acquisition.AcquisitionType tra:

normale: `dso.Acquisition.AcquisitionType = 0;` o
`dso.Acquisition.AcquisitionType = "NORMAL";`

alta risoluzione: `dso.Acquisition.AcquisitionType = 2;` O
`dso.Acquisition.AcquisitionType = "HI_RES";`

peak detect: `dso.Acquisition.AcquisitionType = 1;` o
`dso.Acquisition.AcquisitionType = "PEAK_DETECT";`

inviluppo: `dso.Acquisition.AcquisitionType = 3;` o
`dso.Acquisition.AcquisitionType = "ENVELOPE";`

media: `dso.Acquisition.AcquisitionType = 4;` o `dso.Acquisition.AcquisitionType = "AVERAGE";` con numero di medie `dso.Acquisition.NumberOfAverages` a sua volta modificabile.

Numero di campioni acquisizione: `dso.Acquisition.HorizontalRecordLength`, fornisce il numero di campioni acquisiti dall'oscilloscopio.

Durata intera acquisizione: `dso.Acquisition.HorizontalTimePerRecord` fornisce la durata totale dell'acquisizione.

5.4 Funzione actualRecordLength

Esempio di utilizzo: `numeroCampioni = actualRecordLength(dso);`
Restituisce il numero di campioni che l'oscilloscopio è programmato di acquisire.

5.5 Funzione readWaveform

```
[waveformArray, actualPoints] = readWaveform(dso, "Channel1",
recordLength, rangeMilliseconds);
```

Funzione che permette di acquisire un segnale in ingresso all'oscilloscopio
Restituisce il vettore di campioni e il numero di campioni dell'acquisizione rispettivamente waveformArray e actualPoints.
I parametri espliciti sono nell'ordine: l'oggetto *ividev*, l'ID del canale dal quale si vuole acquisire, il numero di campioni (restituito dalla funzione actualRecordLength(dso)) e la durata della finestra di acquisizione in millisecondi.

5.6 Creazione base dei tempi

Per poter relazionare i dati raccolti col tempo è necessario definire la base dei tempi.
Il vettore della base dei tempi dovrà avere esattamente lo stesso numero di elementi n del vettore di campioni e come suoi elementi il tempo da assegnare a ciascuno dei campioni. Per ottenere ciò si può fare nel modo seguente:

```
% waveformArray è il vettore contenente i campioni dell'acquisizione
n = length(waveformArray);

% dt = tempo intera acquisizione / numero di campioni dell'acquisizione che
% risulta essere quindi la distanza nel tempo dei campioni
dt = dso.Acquisition.HorizontalTimePerRecord/dso.Acquisition.HorizontalRecordLength;

% vettore base dei tempi
t = (0 : n-1) * dt;
```

Si nota che si sarebbe potuto usare il valore numero di campioni restituito dalla funzione *readWaveform*.

5.7 reset e autoSetup

Per entrambe le funzioni deve essere passato come unico parametro l'oggetto dso.

`reset(dso)` effettua un reset delle impostazioni dell'oscilloscopio, utile per portare lo strumento in una condizione conosciuta qual ora non si sappia con che impostazioni potrebbe esser stato lasciato dall'ultimo utilizzo.

`autoSetup(dso)` corrisponde all'*autoscale*, lo strumento viene automaticamente impostato in funzione agli ingressi per permetterne la migliore visualizzazione.

5.8 Gestione errori

Si hanno a disposizione alcune funzioni che permettono di interrogare lo strumento quanto riguarda eventuali errori che si sono verificati e poterne leggere i messaggi.

`[errorCode, errorMsg] = error_query(dso)` restituisce il codice identificativo dell'errore e il messaggio di errore rispettivamente;

`errorMsg = error_message(dso)` restituisce il messaggio di errore;

clearError(dso) elimina lo stato di errore.

5.9 Esempio semplice acquisizione

Si hanno ora tutti gli strumenti necessari a comprendere l'esempio di semplice acquisizione. Lo script ha lo scopo di iniziare una comunicazione con lo strumento ed effettuare una semplice acquisizione di un segnale in ingresso al canale 1 della durata di un millisecondo con impostazioni automatiche.

```
%% Stampa lista dispositivi connessi al pc
devlist = ividevlist("Timeout",5)

%% creazione oggetto ividev (inserimento nome risorsa manuale)
dso = ividev("AgInfiniiVision", "USB0::0X2A8D::0X1797::CN57136331::0::INSTR");

%% reset dello strumento e autoscale
reset(dso)
autoSetup(dso)

%% attivazione canale 1 del dso
dso.Channel("Channel1").ChannelEnabled = true;
% impostazione attenuazione della sonda
dso.Channel("Channel1").ProbeAttenuation = 10;

%% Acquisizione
% record length (numero di campioni acquisiti)
recordLength = actualRecordLength(dso);
% durata acquisizione
rangeMilliseconds = 1;
% lettura dati dall'oscilloscopio
% actualPoints rappresenta il numero di campioni effettivi provenienti
% dall'acquisizione
[waveformArray,actualPoints] =
readWaveform(dso,"Channel1",recordLength,rangeMilliseconds);


%% Visualizzazione risultati

% definizione del numero di elementi dell'array dei campioni
n = length(waveformArray);

% calcolo tempo fra i campioni
% tempo intera acquisizione / numero di campioni dell'acquisizione che
% risulta essere quindi la distanza nel tempo dei campioni
dt = dso.Acquisition.HorizontalTimePerRecord/dso.Acquisition.HorizontalRecordLength;

% creazione base dei tempi
t = (0 : n-1) * dt;

% plot dell'acquisizione
plot(t, waveformArray);
grid on;
xlabel('Time (ms)');
ylabel('Volts (V)');

% verifica errori con stampa sulla Command Window
[errorCode, errorMsg] = error_query(dso);
fprintf('ErrorQuery: %d, %s\n', errorCode, errorMsg);
```

5.10 Variabili trigger

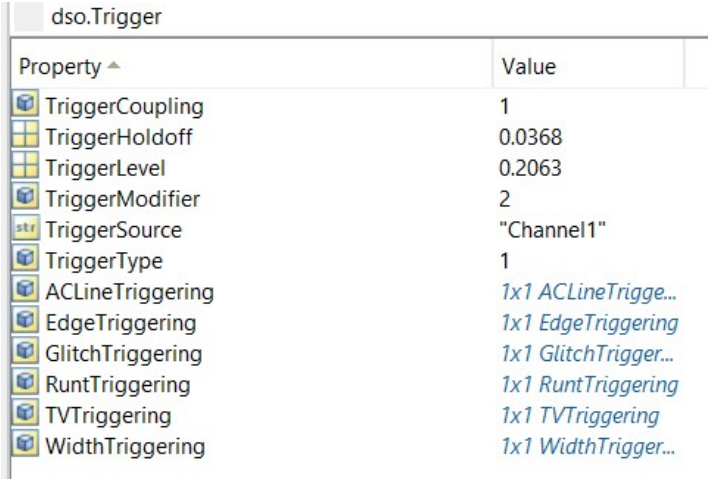


Figura 11 Variabili trigger (strumento fisico)

Accoppiamento trigger: `dso.Trigger.TriggerCoupling` variabile che permette l'impostazione dell'accoppiamento del trigger.
tutte le possibilità sono: "AC" (o 0), "DC" (o 1), "HF_REJECT" (o 2), "LF_REJECT" (o 3), "NOISE_REJECT" (o 4).

Tempo di hold-off: la variabile `dso.Trigger.TriggerHoldoff` rappresenta appunto il tempo di hold-off espresso in millisecondi impostabile semplicemente tramite l'assegnazione del valore desiderato.

Livello trigger: la variabile `dso.Trigger.TriggerLevel` rappresenta il livello a cui è impostato il trigger espresso in volt, qual ora si volesse impostare un diverso livello di trigger è semplicemente necessario modificare il valore della variabile con il valore in volt desiderato.

Sorgente trigger: rappresentato dalla stringa dso.Trigger.TriggerSource indica l'ID del canale attualmente impostato come sorgente di trigger, qual ora si volesse cambiare sorgente sarà sufficiente assegnare l'ID del canale desiderato. Esempio di impostazione del canale 1 come sorgente di trigger: `dso.Trigger.TriggerSource = "Channel1"`.

5.11 Funzione `readWaveformMeasurement(dso, ChannelID, misura, rangeMilliseconds)`

Permette di ottenere il valore di misure automatiche svolte direttamente dall'oscilloscopio. Misura è la stringa identificativa della misura. Gli altri parametri sono già stati analizzati per funzioni precedenti.

Tutte le stringhe corrispondenti alle possibili misure:

```
AMPLITUDE, DUTY_CYCLE_NEG, DUTY_CYCLE_POS, FALL_TIME, FREQUENCY,
MEASUREMENT_AREA2, MEASUREMENT_BURST_WIDTH2, MEASUREMENT_DELAY2,
MEASUREMENT_FALLING_EDGE_COUNT2, MEASUREMENT_FALLING_PULSE_COUNT2,
MEASUREMENT_FREQUENCY_COUNTER2, MEASUREMENT_PHASE2,
MEASUREMENT_RISING_EDGE_COUNT2, MEASUREMENT_RISING_PULSE_COUNT2,
MEASUREMENT_STANDARD_DEVIATION2, MEASUREMENT_TIME_AT_MAX2,
MEASUREMENT_TIME_AT_MIN2, MEASUREMENT_TIME_AT_VALUE2,
MEASUREMENT_TIME_OF_EDGE2, MEASUREMENT_VALUE_AT_TIME2, OVERSHOOT, PERIOD,
PRESHOOT, RISE_TIME, VOLTAGE_AVERAGE, VOLTAGE_CYCLE_AVERAGE,
VOLTAGE_CYCLE_RMS, VOLTAGE_HIGH, VOLTAGE_LOW, VOLTAGE_MAX, VOLTAGE_MIN,
VOLTAGE_PEAK_TO_PEAK, VOLTAGE_RMS, WIDTH_NEG,  WIDTH_POS.
```

Esempio uso

Misura tensione efficace segnale in ingresso al canale 1 considerando un intervallo di tempo della misura di un millisecondo:

```
vrms = readWaveformMeasurement(dso, “Channel1”, “VOLTAGE_RMS”, 1);
```

5.11 Generatore di funzioni interno

dso.InstrumentSpecific.WaveGen

Property	Value
OutputMode	
WaveGenAdditionalNoise	0
WaveGenAmplitude	0.5000
WaveGenDutyCycle	50
WaveGenFrequency	1000
WaveGenFunction	0
WaveGenHighVoltage	0.2500
WaveGenLowVoltage	-0.2500
WaveGenOffsetVoltage	1
WaveGenOutputEnabled	1
WaveGenOutputInverted	
WaveGenOutputLoadImpedance	0
WaveGenPeriod	1.0000e-03
WaveGenPulseWidth	1.0000e-04
WaveGenRampSymmetry	50
Arbitrary	*1x1 Arbitrary*
Modulation	*1x1 Modulation*
Track	*1x1 Track*
RepCapID	"WaveGen1"

Figura 12 Variabili generatore di funzioni interno (strumento simulato)

Abilitazione generatore di funzioni: dso.InstrumentSpecific.WaveGen.WaveGenOutputEnabled, variabile booleana true o false (alternativamente 1 o 0) che permette l’attivazione o la disattivazione dell’output del generatore di funzioni.

Tipo di funzione: dso.InstrumentSpecific.WaveGen.WaveGenFunction, variabile che permette la modifica del tipo di funzione in output al generatore di funzioni.
Le principali alternative sono (con valori da assegnare alla variabile):

sinusoide: 0 oppure “WAVEFORM_GENERATOR_FUNCTION_SINUSOID”;
onda rettangolare: 1 oppure “WAVEFORM_GENERATOR_FUNCTION_SQUARE”;

triangolare: 2 oppure "WAVEFORM_GENERATOR_FUNCTION_RAMP";
impulso: 3 oppure "WAVEFORM_GENERATOR_FUNCTION_PULSE";
rumore: 4 oppure "WAVEFORM_GENERATOR_FUNCTION_NOISE";

Impostazione basso e alto livello:
Basso livello: `dso.InstrumentSpecific.WaveGen.WaveGenLowVoltage` è la variabile che indica il valore del livello basso in volt;
Alto Livello: `dso.InstrumentSpecific.WaveGen.WaveGenHighVoltage` è la variabile che indica il valore del livello alto in volt.
Ampiezza: `dso.InstrumentSpecific.WaveGen.WaveGenAmplitude` è la variabile che indica l'ampiezza del segnale di uscita.

Frequenza: `dso.InstrumentSpecific.WaveGen.WaveGenFrequency` è la variabile che indica la frequenza del segnale in Hertz.

Periodo: `dso.InstrumentSpecific.WaveGen.WaveGenPeriod` è la variabile che indica il periodo del segnale in millisecondi.

Offset: `dso.InstrumentSpecific.WaveGen.WaveGenOffsetVoltage` variabile che indica l'Offset del segnale in uscita in volt.

Impedenza di carico:
`dso.InstrumentSpecific.WaveGen.WaveGenOutputLoadImpedance` è la variabile che indica l'impedenza di carico.
Possibilità disponibili e corrispondenti valori da assegnare alla variabile:
1- Alta impedenza (1 Mohm): 0 oppure "WAVEFORM_GENERATOR_OUTPUT_LOAD_IMPEDANCE_ONE_MEG";
2- 50 ohm: 1 oppure "WAVEFORM_GENERATOR_OUTPUT_LOAD_IMPEDANCE_FIFTY".

Duty cycle: `dso.InstrumentSpecific.WaveGen.WaveGenDutyCycle` duty cycle in percentuale, utilizzabile nel caso di onda rettangolare.

Simmetria rampa: `dso.InstrumentSpecific.WaveGen.WaveGenRampSymmetry`, percentuale tempo di salita rispetto al totale. (Utilizzabile sono nel caso di onda triangolare).

Durata impulso: `dso.InstrumentSpecific.WaveGen.WaveGenPulseWidth`, durata impulso in millisecondi (utilizzabile soltanto nel caso di impulso).

Modulazione:

dso.InstrumentSpecific.WaveGen.Modulation

Property	Value
ModulationEnabled	0
ModulationFunction	0
ModulationRampSymmetry	50
ModulationType	0
AM	*1x1 AM*
FM	*1x1 FM*
FSK	*1x1 FSK*

Figura 13 Variabili modulazione generatore di funzioni (strumento simulato)

Attivazione modulazione:
`dso.InstrumentSpecific.WaveGen.Modulation.ModulationEnabled` variabile booleana true o false;

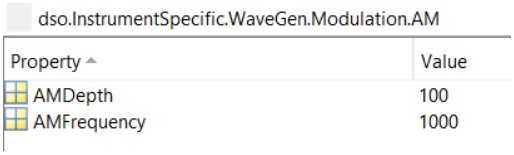
Tipo di modulazione: `dso.InstrumentSpecific.WaveGen.Modulation.ModulationType`
Variabile che permette di impostare il tipo di modulazione.

Possibilità:
modulazione AM: 0 oppure `"MODULATION_TYPEAM"`;
modulazione FM: 1 oppure `"MODULATION_TYPEFM"`;
modulazione FSK: 2 oppure `"MODULATION_TYPEFSK"`.

Funzione modulante:
`dso.InstrumentSpecific.WaveGen.Modulation.ModulationFunction` permette la scelta del tipo della funzione modulante, stesse regole per la scelta del tipo di segnale precedentemente analizzata;

Simmetria rampa modulazione:
`dso.InstrumentSpecific.WaveGen.Modulation.ModulationRampSymmetry`
impostazione simmetria rampa modulante (nel caso il tipo di modulante sia rampa).

Altre impostazioni sono possibili a seconda del tipo di modulazione scelta, cliccando sul "sotto-oggetto" relativo alla modulazione impostata, esempio:
Profondità e frequenza modulazione AM.


dso.InstrumentSpecific.WaveGen.Modulation.AM

Property	Value
AMDepth	100
AMFrequency	1000

Figura 14 Variabili modulazione AM (strumento simulato)

5.12 Esempio misura automatica con uso di due canali dell'oscilloscopio e del generatore di funzioni interno: risposta in frequenza di un amplificatore

Nota: per il corretto funzionamento dello script il segnale di ingresso all'amplificatore deve essere collegato al canale 1, mentre quello di uscita al canale 2.

Funzione:

```
function [magnitudedb, phase, f] = frequencyResponseFunction(startFreq, endFreq, amp, nMeas, nPer,
resourceName)

    % Creazione Array risultati
    freqRespM = ones(nMeas, 1);
    freqRespMdb = freqRespM;
    freqRespPh = zeros(nMeas, 1);

    % Base delle frequenze
    df = (endFreq - startFreq)/(nMeas-1);
    f = (startFreq: df :endFreq);

    % creazione oggetto ividev
    % strumento fisico collegato
    dso = ividev("AgInfiniiVision",resourceName);

    % strumento simulato
    % dso = ividev("AgInfiniiVision", "", Simulate = true);

    reset(dso)
    autoSetup(dso)

    dso.Channel("Channel1").ProbeAttenuation = 1;
    dso.Channel("Channel2").ProbeAttenuation = 1;

    % setup del generatore di funzioni
    % disabilitazione output
    dso.InstrumentSpecific.WaveGen.WaveGenOutputEnabled = 0;
    % impostazione segnale sinusoidale
    dso.InstrumentSpecific.WaveGen.WaveGenFunction = 0;
    % impostazione livello basso e livello alto
    dso.InstrumentSpecific.WaveGen.WaveGenLowVoltage = -amp;
    dso.InstrumentSpecific.WaveGen.WaveGenHighVoltage = amp;
    % impostazione frequenza del segnale alla frequenza di partenza
    dso.InstrumentSpecific.WaveGen.WaveGenFrequency = startFreq;
    % abilitazione output
    dso.InstrumentSpecific.WaveGen.WaveGenOutputEnabled = 1;
```

```
    % ciclo for con numero di iterazioni pari al numero di misure
    for i = 1 : nMeas

        % impostazione della frequenza generatore di funzioni
        dso.InstrumentSpecific.WaveGen.WaveGenFrequency = f(i);

        % autoSetup oscilloscopio + impostazioni
        autoSetup(dso)

        dso.Channel("Channel1").ChannelEnabled = true;
        dso.Channel("Channel2").ChannelEnabled = true;
        % tipo di acquisizione
        dso.Acquisition.AcquisitionType = "HI_RES";
        % impostazione sorgente di trigger canale 1
        dso.Trigger.TriggerSource = "Channel1";
        % livello del trigger in volt
        dso.Trigger.TriggerLevel = 0;

        % impostazione accoppiamento ac del canali 1 e 2
        dso.Channel(1, 1).VerticalCoupling = 0;
        dso.Channel(1, 2).VerticalCoupling = 0;

        % calcolo durata acquisizione in millisecondi partendo dal numero
        % di periodi da considerare e la frequenza del segnale di ingresso
        rangeMillisecondi = 1000 * nPer/f(i);

        [waveformArrayCh1, l1] = readWaveform(dso,"Channel1",actualRecordLength(dso),rangeMillisecondi);
        [waveformArrayCh2, l2] = readWaveform(dso,"Channel2",actualRecordLength(dso),rangeMillisecondi);

        % calcolo rms delle due acquisizioni
        rmsIn = rms(waveformArrayCh1);
        rmsOut = rms(waveformArrayCh2);
        % % alternativa misura Vrms con uso della funzione:
        % % readWaveformMeasurement
        % % misura diretta rms due canali
        % rmsIn = readWaveformMeasurement(dso, "Channel1", "VOLTAGE_RMS", rangeMillisecondi);
        % rmsOut = readWaveformMeasurement(dso, "Channel2", "VOLTAGE_RMS", rangeMillisecondi);

        % calcolo rapporto modulo
        freqRespM(i) = rmsOut / rmsIn;
        % ... in db
        freqRespMdb(i) = 20*log10(freqRespM(i));


        % phaseDiff funzione per il calcolo della fase usata come esempio (risultati non corretti)
        freqRespPh(i) = phaseDiff(waveformArrayCh1, waveformArrayCh2);
        % % alternativa misura fase con uso della funzione
        % % readWaveformMeasurment
        % % calcolo sfasamento NON TESTATO
        % freqRespPh(i) = readWaveformMeasurement(dso, "Channel2", "MEASUREMENT_PHASE2",
        % rangeMillisecondi) - readWaveformMeasurement(dso, "Channel1", "MEASUREMENT_PHASE2",
        % rangeMillisecondi);

    end

    % disabilitazione del generatore di funzioni
    dso.InstrumentSpecific.WaveGen.WaveGenOutputEnabled = false;

    clear dso

    % assegnazione risultati misure
    magnitudedb = freqRespMdb;
    phase = freqRespPh;

end
```

Uso della funzione tramite script

```
%% dati della misura
% frequenza di partenza (Hz)
startFreq = 100;
% frequenza fine misura (Hz)
endFreq = 5000;
% ampiezza ingresso (V)
amp = 0.5;
% numero di misure
nMeas = 50;
% numero periodi acquisizione
% ovvero numero di periodi su cui si effettua la misura
nPer = 3;

%% chiamata funzione risposta in frequenza
% risultati: [risposta in frequenza del modulo, risposta in frequaenza
% della fase, base delle frequenze]
% parametri: (frequenza iniziale, frequenza finale, ampiezza segnale in
% ingresso all'amplificatore, numero di misure, numero di periodi su cui si
% effettua la misura, nome della risorsa oscilloscopio)
[freqRespMdb, freqRespPh, f] = frequencyResponseFunction(startFreq, endFreq, amp,
nMeas, nPer, "USB0::0x2A8D::0x1797::CN57136331::0::INSTR");

%% plot dell'acquisizione
figure(1)
tiledlayout(2,1)
title(1, "Risposta in frequenza")

ax1 = nexttile;
semilogx(ax1, f, freqRespMdb, "-")
title(ax1, "Modulo")
xlabel(ax1, "Frequenza (Hz)")
ylabel(ax1, "Modulo (dB)")
grid minor

ax2 = nexttile;
semilogx(ax2, f, freqRespPh, "-")
title(ax2, "Fase")
xlabel(ax2, "Frequenza (Hz)")
ylabel(ax2, "Fase (°)")
grid minor
```

Nota: il codice della funzione effettua un'acquisizione e a partire dai dati calcola l'amplificazione in decibel dell'amplificatore collegato e la differenza di fase tra il segnale in uscita e quello in ingresso. Nei commenti è stata inserita una versione del codice che permetterebbe di accedere direttamente alle misure di tensione efficace e della fase che però non è stata testata.
Accedendo alle misure per la fase direttamente dall'oscilloscopio potrebbe portare a risultati corretti senza dover implementare una funzione per il calcolo della fase più complicata.

Esempio risultati per il modulo:

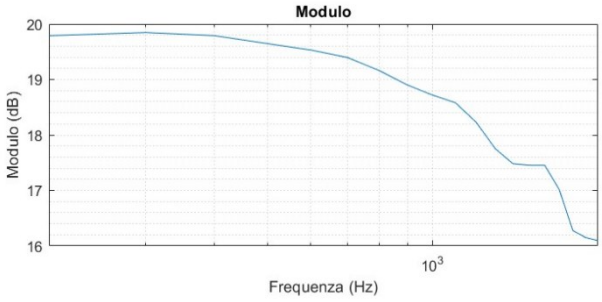


Figura 15 Esempio diagramma del modulo della risposta in frequenza ricavato con il codice dell'esempio precedente

6 Controllo multimetro digitale keysight Ag34461A

Durante la trattazione si farà riferimento all'oggetto *ividev* creato per il multimetro digitale con "dmm".
Verranno in seguito riportate le informazioni necessarie per effettuare la configurazione e vari tipi di acquisizione con lo strumento.
L'esposizione circa controllo di questo strumento, come precedentemente accennato, è incentrata su l'uso di funzioni già disponibili.

6.1 Misure automatica senza la necessità di alcuna impostazione manuale:

Tensione continua: dcVoltageMeasureAuto(dmm) [Volt];
Tensione alternata (tensione efficace): acVoltageMeasureAuto(dmm) [Volt];
Corrente continua: dcCurrentMeasureAuto(dmm) [Ampere];
Corrente alternata (corrente efficace): acCurrentMeasureAuto(dmm) [Ampere];
Resistenza: resistanceMeasureAuto(dmm) [ohm].

Sono funzioni simili tra loro, come unico parametro hanno l'oggetto "ividev" relativo allo strumento e restituiscono semplicemente il valore della misura.
La misura effettuata in questo modo è completamente automatica, è infatti effettuabile immediatamente dopo la creazione dell'oggetto "ividev".

6.2 Misure con impostazioni manuali

Prima di tutto è importante ricordare che dopo ogni modifica di un'impostazione è necessaria la chiamata della funzione **`initiate(dmm)`** prima di effettuare una qualsiasi misura. La funzione `initiate` fa uscire lo strumento dallo stato di "idle" (attesa) indotto dalle funzioni di configurazione.

fetch(dmm, timeoutMilliseconds): funzione che effettua l'acquisizione con la configurazione impostata manualmente. (Funzione `read(dmm, timeoutMilliseconds)` è equivalente).

Configurazione automatica:
Tensione alternata: acVoltageConfigureAuto(dmm);
Tensione continua: dcVoltageConfigureAuto(dmm);

Corrente alternata: acCurrentConfigureAuto(dmm);
Corrente continua: dcCurrentConfigureAuto(dmm);
Resistenza: resistanceConfigureAuto(dmm).

Lo strumento imposta in maniera automatica il range di misura per effettuare la misura specificata dal nome stesso della funzione.

Reset dello strumento: per resettare le impostazioni dello strumento è sufficiente eseguire la funzione reset(dmm).

Configurazione manuale:
Tensione alternata: acVoltageConfigure(dmm, range, precisione);
Tensione continua: dcVoltageConfigure(dmm, range, precisione);
Corrente alternata: acCurrentConfigure(dmm, range, precisione);
Corrente continua: dcCurrentConfigure(dmm, range, precisione).
Resistenza: resistanceConfigure(dmm, range, precisione).
In questo caso si dovrà specificare l'oggetto come sempre ma in particolare il range di misura e la precisione del multimetro utilizzato.

Il range di misura può assumere solo valori compatibili con gli effettivi range di misura dello strumento fisico. Se si è impostato un range di misura troppo piccolo rispetto all'input lo strumento dà errore di "overload".

In maniera equivalente alle funzioni appena analizzate si può usare quella generica:
`configureMeasurement(dmm, "tipo_di_misura", range, precisione);`

dove il tipo di misura è uno dei parametri, le opzioni sono:

"VAL_2_WIRE_RES" (misura di resistenza a due cavi), "VAL_4_WIRE_RES" (misura di resistenza a 4 cavi), "AC_CURRENT", "AC_PLUS_DC_CURRENT", "AC_PLUS_DC_VOLTS", "AC_VOLTS", "DC_CURRENT", "DC_VOLTS", "FREQ", "PERIOD", "TEMPERATURE".

6.3 Gestione degli errori:

[errorCode, errorMsg] = error_query(dmm): funzione che restituisce il codice identificativo dell'errore e il messaggio di errore.

errorMsg = error_message(dmm): funzione che restituisce solo il messaggio di errore.

clearError(dmm): funzione che resetta lo stato di errore.

6.4 Esempio di misura multimetro digitale

```
%% Stampa lista dispositivi connessi al pc
devlist = ividevlist("Timeout",5)

%% creazione oggetto ividev (inserimento nome risorsa manuale)
dmm = ividev("Ag3446x", "USB0::0x2A8D::0x1301::MY53224712::0::INSTR")

%% configurazione misura DC
range = 1; %range misura in volt
precisione = 0.001; %precisione misura
configureMeasurement(dmm, "DC_VOLTS", range, precisione);

%% abilitazione multimetro digitale
initiate(dmm);

%% misura
mis = fetch(dmm, 10)
```

7 App Designer

7.1 Perché creare una app

La creazione di un'interfaccia grafica permette un uso più *user-friendly* dello strumento collegato con Matlab rispetto alla modifica manuale dei valori delle variabili di uno script, consente infatti l'automatizzazione o la semplificazione di alcune fasi del collegamento evitando che il codice debba essere ogni volta modificato per inserimento dei parametri richiesti dall'elaborazione. Supponendo che in un laboratorio ci sia la necessità di effettuare una misura ricorrente, la creazione di una semplice interfaccia grafica in cui ci siano dei campi numerici in cui inserire i dati necessari e un bottone per l'avvio della misura consentirebbe di ridurre il tempo necessario per effettuare la misura e ridurre la probabilità di errore da parte dell'operatore mantenendo il codice invariato, l'involontario inserimento di errori nel codice potrebbe portare ad una perdita di tempo e risorse per il suo ripristino.

Come esempio è stata realizzata una app con interfaccia grafica per il controllo dell'oscilloscopio Keysight DSOX1102G, tutte le immagini riportate in seguito derivano dall'ambiente di lavoro relativo allo sviluppo di tale app.

7.2 Ambiente di lavoro

Matlab mette a disposizione un tool per lo sviluppo di app.
L'interfaccia del tool si compone di due finestre ovvero la *Design View* e la *Code View*, si può passare da una all'altra selezionando una delle due opzioni in alto a destra.

La *Design View* ci consente di comporre l'interfaccia dell'app con gli strumenti nella Component Library (finestra a sinistra) semplicemente agendo col mouse.

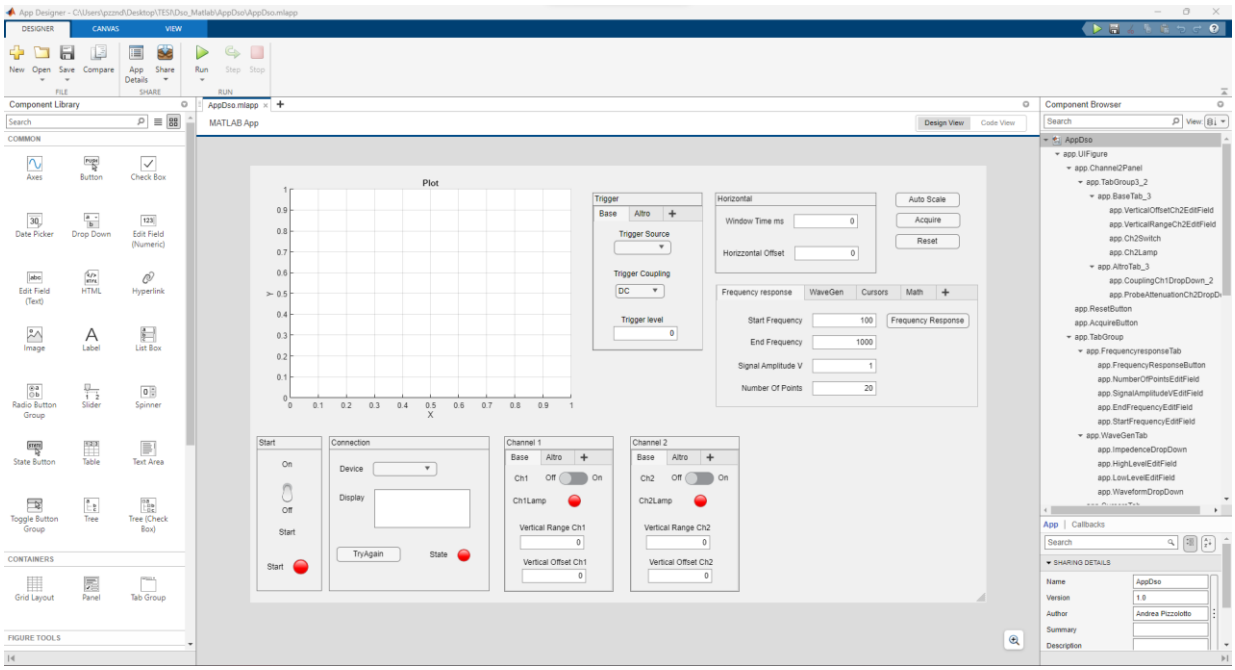


Figura 16 Design View

La *Code View* ci permette di definire il comportamento effettivo dei componenti della nostra interfaccia e quindi le funzionalità dell'app. Le sezioni grigie sono automaticamente generate e sono la diretta conseguenza delle azioni effettuate nella *Design View*, le sezioni bianche rappresentano il codice scritto a mano.

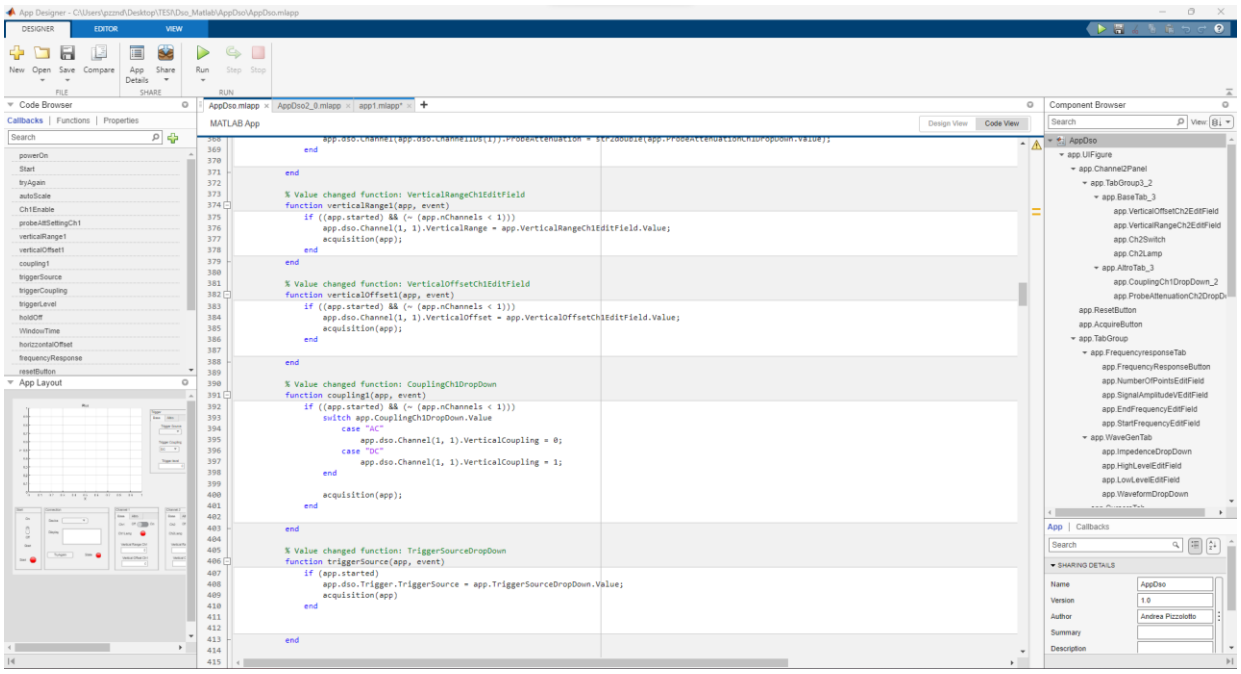


Figura 17 Code View

7.3 Proprietà dei componenti dell'interfaccia grafica

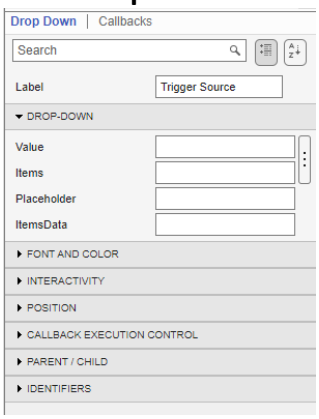


Figura 18 Proprietà componente Drop Down

Cliccando una volta col tasto sinistro del mouse un componente dell'interfaccia nella Design View nel riquadro in basso a destra sarà possibile visualizzarne le proprietà ed eventualmente modificarle.
Tali proprietà sono modificabili anche via codice in maniera da poter adattare l'interfaccia alle esigenze previste dal funzionamento dell'applicazione utilizzando la sintassi `oggetto.proprieta = nuovaProprieta`.

7.4 Implementazione funzionalità

L'implemetazione effettiva delle funzionalità dell'applicazione è possibile grazie a delle particolari funzioni dette *Callbacks* che permettono di programmare la risposta a degli eventi che possono essere semplicemente l'avvio dell'applicazione o un'azione, da parte dell'utente, su un componente dell'interfaccia grafica. È possibile anche creare funzioni che possono essere chiamate in altre sezioni dell'app *(Function)* e la dichiarazione di variabili globali *(Property).*

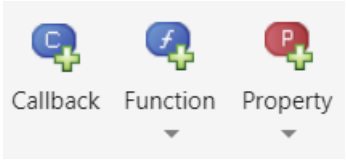


Figura 19 Bottoni per la creazione di nuovi callbacks, funzioni e proprietà

Cliccando sulla sezione *Callbacks* dell'oggetto selezionato si apre la finestra:

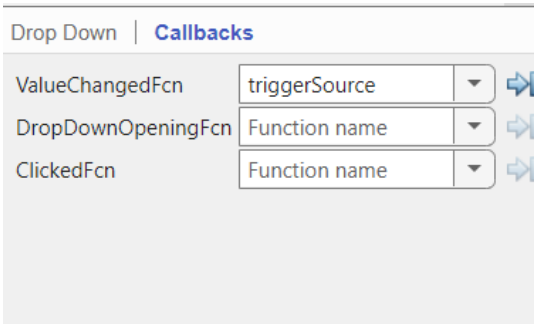


Figura 20 Callbacks relativi ad un componente (Drop Down) dell'interfaccia grafica realizzata come esempio

dove sono elencate le varie azioni effettuabili sul componente e dove è possibile quindi selezionare la funzione di *Callback* corrispondente. Il componente preso in considerazione è un menù a tendina che permette di selezionare la sorgente di trigger, una volta che la sorgente selezionata cambia viene chiamata la funzione di *Callback* triggerSource che ha come scopo quello di impostare la nuova sorgente di trigger. Qual ora la funzione di *Callback* non fosse ancora stata creata è sufficiente inserire il nome che si intende darle nel riquadro opportuno ed essa verrà automaticamente creata. Può essere importante considerare che diversi componenti dell'interfaccia grafica possono fare riferimento alla stessa funzione di *Callback*.

La sezione *Code Browser* della *Code View* permette la visualizzazione di funzioni, Callbacks e proprietà, permette anche la loro eliminazione o rinominazione (la rinominazione effettuata in questo modo verrà riportata su tutto il codice in maniera automatica).

7.5 Esempio acquisizione con due canali tramite applicazione:

Combinando le conoscenze acquisite quanto riguarda il controllo di strumenti e la creazione di un'interfaccia grafica è possibile realizzare una app come quella realizzata per esempio.
L'app in questione è in grado di connettersi allo strumento (Keysight DSOX1102G), gestirne le impostazioni ed effettuare misure automatiche come la risposta in frequenza.
I risultati delle misure vengono stampati per renderne possibile la visualizzazione e un'eventuale analisi

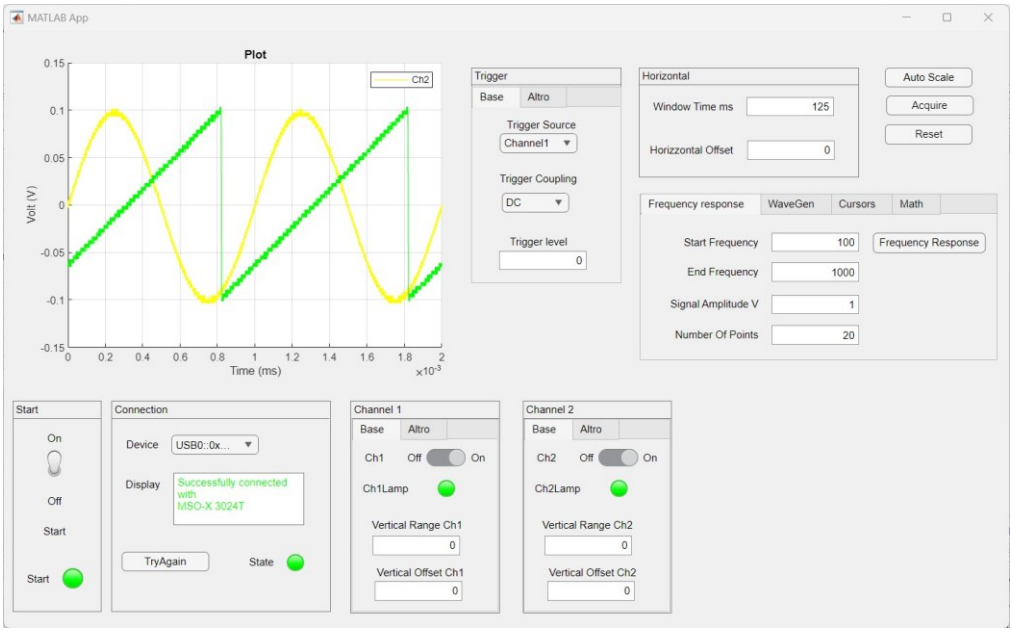


Figura 21 Schermata acquisizione a due canali con interfaccia creata come esempio

8 Link di riferimento

Descrizione passaggi preliminari:
https://it.mathworks.com/help/instrument/install-instrument-control-toolbox-support-package-for-ivi-and-vxiplugplay-drivers.html

Driver Keysight DSOX1102G:
https://www.keysight.com/us/en/lib/software-detail/driver/infiniivision-xseries-oscilloscope-ivi-and-matlab-instrument-drivers-2019021.html

Driver Keysight Ag3446x:
https://www.keysight.com/us/en/lib/software-detail/driver/3446x-digital-multimeters-ivi-and-matlab-instrument-drivers-2352538.html

Driver ivi supportati da Matlab (da cui è possibile reperire alcuni esempi di codice):
https://it.mathworks.com/help/instrument/supported-ivi-drivers.html

Documentazione App Designer:
https://it.mathworks.com/help/matlab/app-designer.html

9 Conclusioni: Risultati raggiunti

Con le informazioni riportate nel documento è possibile capire come collegare strumenti, tra quelli compatibili, a Matlab e come sfruttarli per eseguire misure. Anche se si decidesse di usare strumenti che non siano tra i due testati è possibile comunque adattare le modalità di approccio alla connessione e alla scrittura del codice Matlab.
In aggiunta viene data anche un'introduzione all'uso dell'App Designer con il fine di creare interfacce grafiche. Per motivi di brevità tale parte di relazione ha solo lo scopo di dare le informazioni di base per la comprensione del funzionamento del tool.