

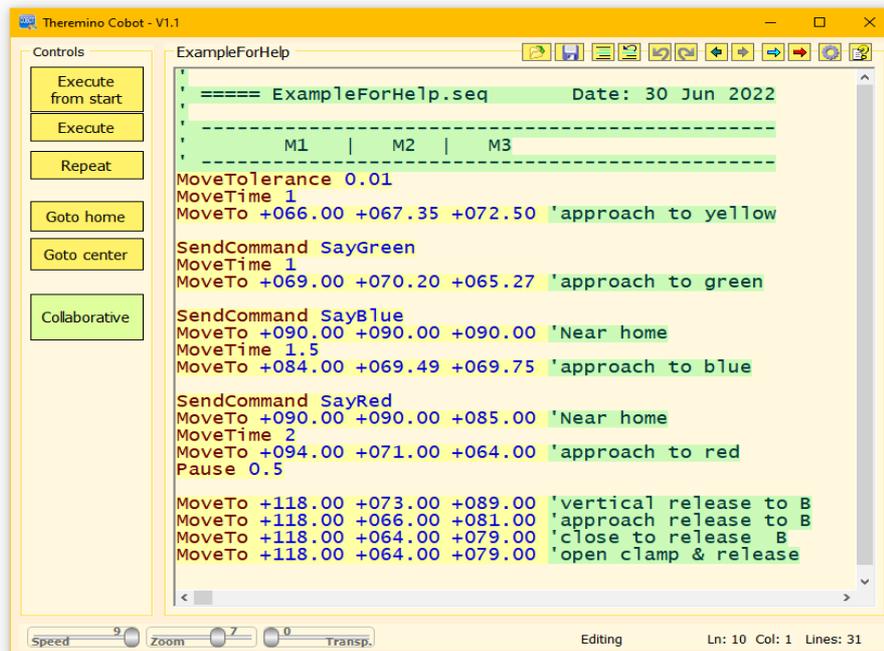
**theremino**  
•the•real•modular•in-out•

**Sistema** theremino

# **Theremino Cobot**

## **V1.2**

# La applicazione Theremino Cobot



Questa applicazione contiene il necessario per muovere un Cobot, cioè un braccio robotico o qualunque altro meccanismo progettato per collaborare con gli umani.

Si possono controllare dispositivi con uno o più motori, disposti in qualunque configurazione, e non è necessario specificare le caratteristiche dimensionali dei meccanismi.

Abbiamo volontariamente eliminato la descrizione tridimensionale con tutte le complicazioni che essa comporta e specifichiamo direttamente la posizione per ogni motore, in gradi, millimetri o nelle altre unità di misura che si preferiscono.

I calcoli tridimensionali sono sempre imprecisi, poiché si dovrebbero impostare tutti i parametri che contribuiscono agli errori. Ma sono parametri difficili da misurare e variano a seconda delle rotazioni e dei pesi trasportati.

Quindi deleghiamo i calcoli a un computer analogico, che è la meccanica stessa, che li compie con precisione assoluta. Tenendo conto di ogni fattore possibile, dalle forze in gioco alla gravità, ai cedimenti, alle imprecisioni strutturali e anche le imprecisioni causate dagli ingranaggi, dai motori e dai loro loop di retroazione.

Queste semplificazioni facilitano di molto l'uso della applicazione ma in alcuni casi si vorrebbero ottenere movimenti più lineari o un controllo maggiore delle operazioni. In tali casi leggete le [tecniche avanzate](#) alla fine di questo documento.

# Come è fatto un Cobot

I Cobot sono progettati per collaborare senza costituire un pericolo per gli esseri umani per cui hanno una struttura diversa dai robot industriali.

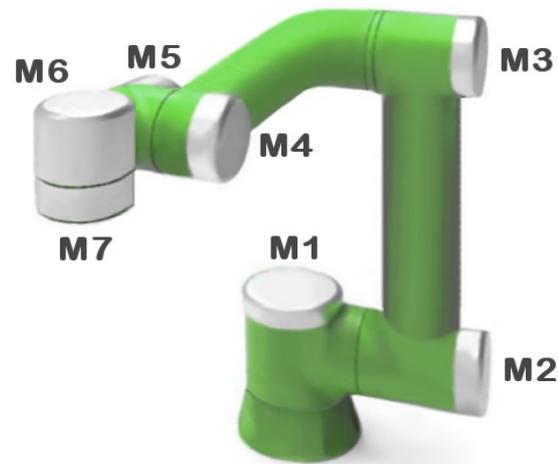
La applicazione Theremino Cobot può controllare qualunque numero di motori in qualunque configurazione meccanica essi siano disposti. Si potrebbero controllare motori separati, ad esempio un tappeto scorrevole e una ghigliottina per tagliare il prodotto che scorre. Oppure si potrebbe controllare un braccio in configurazione SCARA oppure un robot di tipo DELTA, ma normalmente si controlla un Cobot di tipo antropomorfo che è il più flessibile e quindi utilizzabile per molti compiti diversi nella media e piccola industria.

Ecco un esempio di braccio Cobot antropomorfo.

Questa immagine è solo uno schema, la forma potrebbe anche essere diversa ma i componenti principali saranno sempre gli stessi e anche le proporzioni saranno più o meno queste.

Ogni disco argentato rappresenta un giunto di rotazione e quindi anche un motore.

Diamo ad ogni parte un nome significativo:



- **M1 BASE** Rotazione sul piano orizzontale. Potrebbe anche ruotare di 360 gradi ma solitamente lavora solo in avanti e ha dei fermi laterali. Non ha bisogno di molta coppia ma deve essere preciso (poco backlash) perché i suoi errori vengono moltiplicati per tutta la lunghezza del braccio.
- **M2 SPALLA** Alza tutto il braccio. Questo è il motore che deve avere più coppia e deve anche essere molto preciso.
- **M3 GOMITO** Completa il lavoro della spalla e permette di raggiungere ogni posizione. Questo motore deve essere mediamente preciso.

I giunti **M4**, **M5**, **M6** e **M7** sono la **MANO**

- **M4 e M5 POLSO** Questi giunti ruotano su-giù e destra-sinistra, non devono essere precisi o avere molta coppia, ma devono essere piccoli e leggeri.
- **M6 ROTAZIONE** Questo giunto potrebbe fornire una rotazione continua, ad esempio per controllare un cacciavite. In questo caso M7 non ci sarà.
- **M7 PINZA** Questo non è un giunto rotante ma un attrezzo per prendere, avvitare, saldare, verniciare ecc...

# Progettare Cobot sicuri

Un Cobot deve poter lavorare fianco a fianco con un essere umano senza costituire un pericolo. Deve avere superfici arrotondate e in tutte le rotazioni possibili le sue parti non devono avvicinarsi troppo tra loro, altrimenti potrebbero pinzare la pelle o le dita come uno schiaccianoci.



La sicurezza di un Cobot deve essere intrinseca. La meccanica stessa non deve disporre di forza e di velocità in grado di fare del male.

Questo è un principio semplice, facile da capire, niente protezioni software o elettroniche.

Niente regole o protocolli che possono sbagliare e niente meccanismi che possono rompersi.

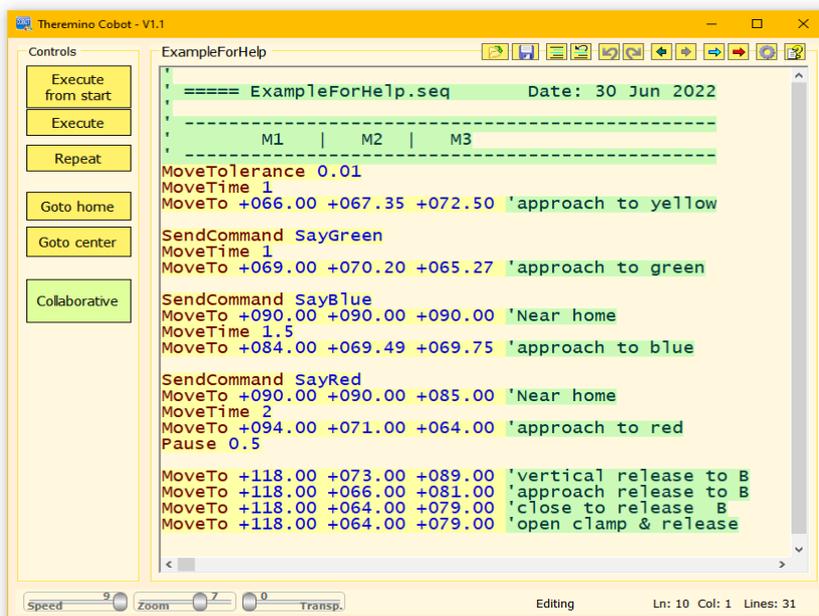
La sicurezza di un Cobot deve essere intrinseca e non basata su certificazioni, sistemi elettronici o software, perché il software può sbagliare, l'elettronica può rompersi e le certificazioni potrebbero essere incomplete o male interpretate.

I meccanismi di sicurezza basati su software, protocolli e certificazioni, oltre a non essere sufficienti, possono anche aumentare la pericolosità del sistema perché generano una falsa fiducia negli umani, che sono quindi portati a fidarsi ciecamente e rischiare.

Guardate [questo video](#) di un robot che ha rotto il dito a un bambino.

E leggete la nostra documentazione **Theremino Cobot Security** pubblicata in inglese, italiano e cinese.

# Utilizzare la applicazione Cobot



Questa applicazione è controllabile in molti modi, come vedremo nelle prossime pagine, ma il suo funzionamento di base è semplice.

Si scrivono le righe di comando nella lista e le si eseguono con i comandi di questa pagina.

Si modificano i valori nella lista guardando i movimenti del Cobot.

Oppure si inviano comandi da un'altra applicazione (che di solito è `Theremino_Automation`).

Ecco i principali comandi:



Premendo `Execute from start` si fa partire l'esecuzione dall'inizio.

Con `Execute` l'esecuzione inizia dalla riga con il cursore, cioè la riga selezionata con il mouse o la tastiera.

Se si abilita il pulsante `Repeat` allora dopo aver eseguito l'ultima riga l'esecuzione riparte dalla prima riga.

Con `Goto home` si fa muovere il Cobot alla posizione pre-selezionata nel modo `Collaborative`.

Con `Goto center` si fa muovere il Cobot alle posizioni pre-selezionate nel modo `Collaborative`.

Con `Collaborative` si accede al modo collaborativo che vedremo nelle prossime pagine.

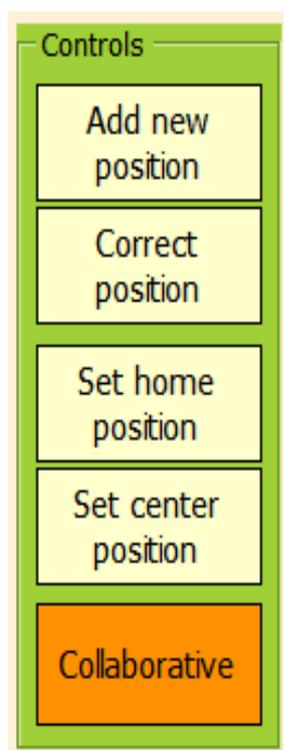
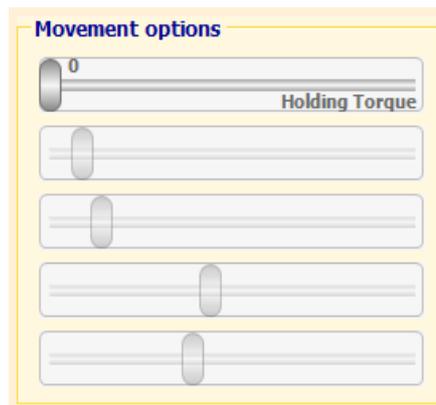
Per fermare l'esecuzione si preme nuovamente `Execute` o si fa click su una riga della lista dei comandi.

## Il modo "Collaborativo"

Quando si entra nel modo collaborativo nella finestra delle opzioni appare il cursore **Holding torque** che serve per limitare la coppia.

Spostando **Holding torque** verso sinistra la coppia viene limitata di molto e diventa possibile ruotare manualmente i giunti del Cobot.

I particolari di questo cursore e degli altri quattro sono spiegati nelle prossime pagine.



Premendo **Add new position** si aggiunge alla sequenza una riga **MoveTo** con la posizione attuale di tutti i motori.

Con **Correct position** si modifica la riga **MoveTo** selezionata con la posizione attuale dei motori.

Con **Set home position** si imposta la posizione **Home** con la posizione attuale dei motori **Nota 1**

Con **Set center position** si imposta la posizione **Center** con la posizione attuale dei motori.

Premendo il pulsante **Collaborative** si esce dal modo collaborativo e si torna al modo di normale funzionamento.

Il modo collaborativo serve solo per variazioni, dopo averle fatte premete il pulsante "Collaborative" e tornate al modo di normale funzionamento.

**Nota 1** - Se si utilizzano i motori oltre i 360 gradi la posizione Home deve stare nel primo giro (da 0 a 4095 per i FeeTech) e prima di spegnere il braccio si dovrebbe riportarlo alla posizione Home.

## Editare le sequenze di comandi

Durante il normale funzionamento, e senza entrare nel modo collaborativo, si possono **modificare i comandi** della sequenza con la **tastiera** e con il **Mouse**.

Se la sequenza è in esecuzione si ferma automaticamente appena si posa il cursore del mouse sulla lista dei comandi o si utilizzano le frecce SU e GIU.

Facendo click con il tasto destro sui comandi si apre un menu che vedremo nei dettagli nelle ultime pagine di questo documento.

I comandi validi, che vedremo nelle prossime pagine, vengono evidenziati con un colore di fondo giallo, mentre gli errori con un colore rosso.

## Aggiungere posizioni

Si possono aggiungere posizioni scrivendo manualmente i comandi **MoveTo** oppure copiando e incollando altre righe già pronte.

Si possono anche aggiungere posizioni utilizzando il modo Collaborative, eventualmente anche con un pulsante esterno mentre si muove manualmente il Cobot nelle posizioni desiderate.

## Eseguire i comandi riga per riga

Si possono provare i singoli comandi eseguendoli con un **doppio click del mouse** sulla prima parola della riga.

Se c'è una riga selezionata (con lo sfondo blu) allora si possono utilizzare le frecce SU e GIU della tastiera per eseguire le righe precedenti e seguenti.

## Modificare le posizioni con il mouse

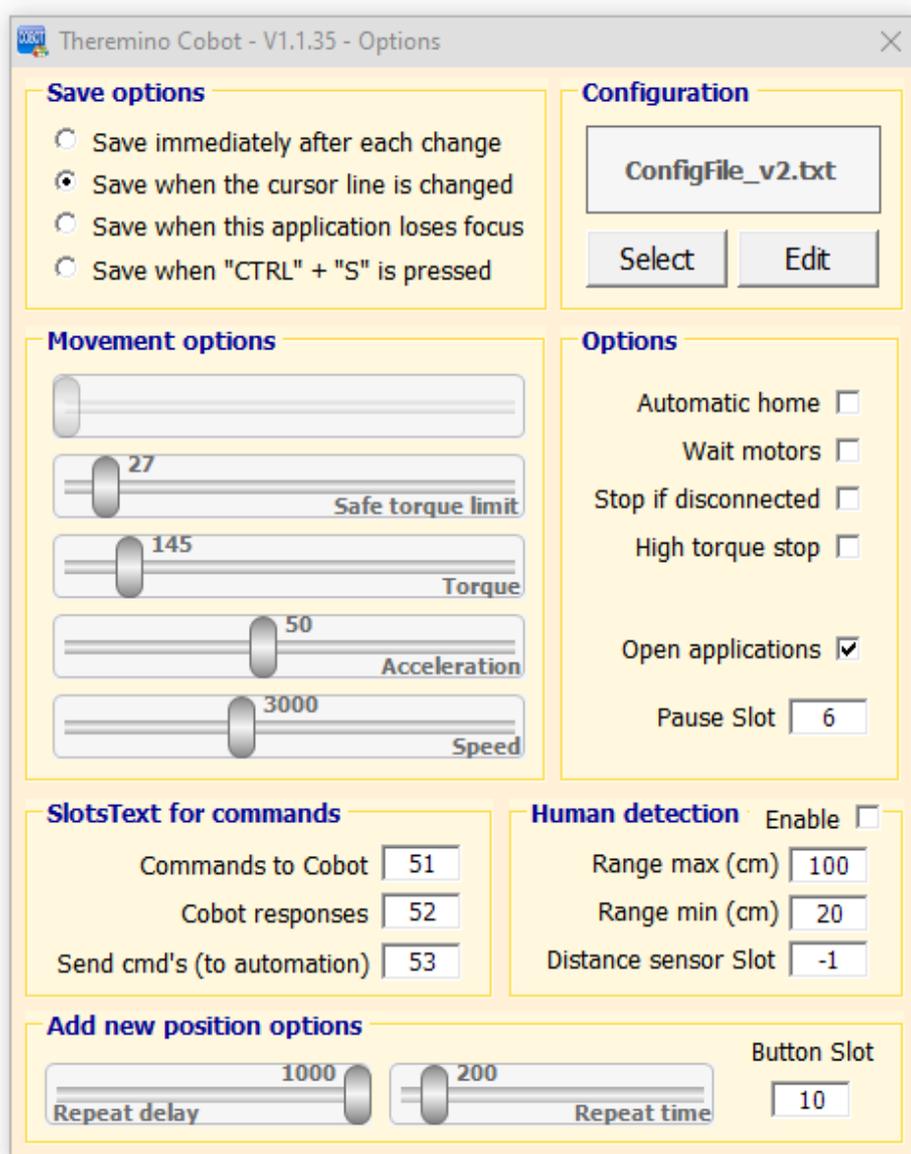
Si possono modificare i valori di rotazione delle righe **MoveTo** e vedere l'effettivo movimento nel Cobot mentre si fa la variazione.

- Con un **doppio click** selezionare un valore numerico di una riga MoveTo.
- Fino a che il valore numerico è selezionato la rotella del mouse lo modifica e si può vedere immediatamente il movimento guardando il Cobot e senza bisogno di stare attenti alla posizione del cursore del Mouse.

La velocità di variazione dei valori numerici è modificabile premendo i tasti **SHIFT**, **CTRL** e **ALT** mentre si utilizza la rotella del Mouse.

Normalmente la velocità quando non si premono tasti è di un numero intero ad ogni tacca della rotella del mouse, mentre le velocità con SHIFT, CTRL e ALT sono di 10, 0.1 e 0.01. Questi valori sono modificabili nella configurazione che è spiegata nelle prossime pagine.

# Il pannello delle opzioni

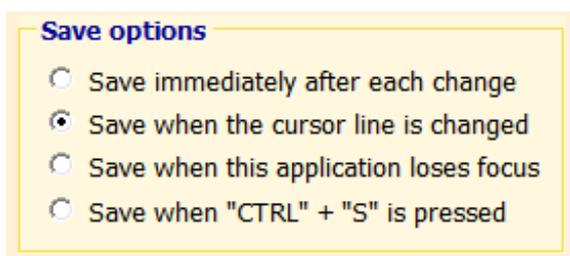


Questo pannello si apre premendo il pulsante con l'ingranaggio che si trova in alto a destra nella finestra principale.



Nelle prossime pagine vedremo il significato e l'utilizzo delle varie zone di questo pannello.

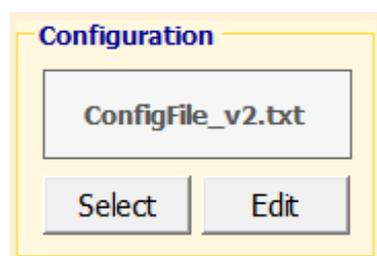
## Pannello opzioni - Save options



Queste opzioni determinano quando la lista dei comandi viene salvata su disco.

Si può salvare automaticamente in vari modi o quando si preme CTRL-S. In tutti i casi la lista viene anche salvata prima di caricarne un'altra da file o quando si chiude la applicazione.

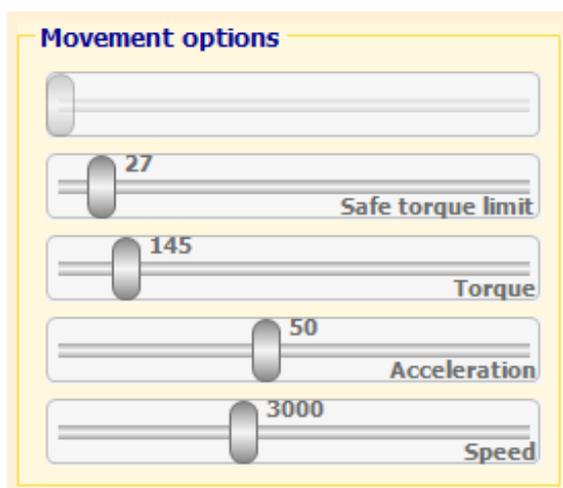
## Pannello opzioni - Configuration



Con questi pulsanti si sceglie il file di configurazione da utilizzare e lo si apre in un editor di testo (solitamente NotePad) per modificarlo.

I dettagli della configurazione sono spiegati nelle prossime pagine.

## Pannello opzioni - Movement options



Il primo cursore in alto, che qui è disabilitato, serve per limitare la coppia nel modo "Collaborative".

Se si fa click con il tasto destro i cursori vanno al valore di default. I valori di massimo e di default di questi cursori sono nella configurazione.

I valori **Torque**, **Acceleration** e **Speed** vengono utilizzati come limiti dentro al motore stesso.

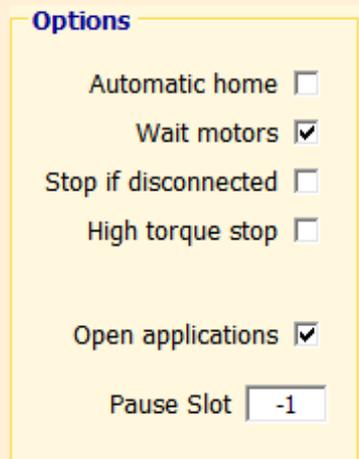
In alcuni motori (FeeTech) il limite di coppia non viene controllato bene nel firmware e quando lo si abbassa la accelerazione non funziona più bene per cui è bene tenerlo al massimo **Nota 1**

**Nota 1** - Se si deve tenere il Torque al massimo si può delegare la limitazione della coppia alla applicazione Cobot, che abbasserà il Torque del motore quando la coppia sale oltre al limite impostato in Safe torque limit

Non tutti i motori implementano nel firmware queste regolazioni, gli unici che utilizzano tutte le caratteristiche della applicazione Cobot sono i FeeTech e i TMOT (Theremino Motors) (in progettazione nel 2022).

# Pannello opzioni - Options

Queste opzioni controllano alcuni comportamenti di ordine generale della applicazione.



The screenshot shows a yellow-bordered box titled "Options". Inside, there are the following controls:

- Automatic home
- Wait motors
- Stop if disconnected
- High torque stop
- Open applications
- Pause Slot

- **Automatic home** - Abilita il movimento automatico alle posizioni Home ad ogni avvio e chiusura della applicazione **Nota 1**
- **Wait motors** - Normalmente lo si tiene abilitato e lo si disabilita solo per provare le sequenze di comandi senza motori.
- **Stop if disconnected** - Se lo si abilita ferma l'esecuzione della sequenza se la tensione dei motori scende sotto ai 2 Volt **Nota 2**
- **High torque stop** - Se lo si abilita ferma l'esecuzione della sequenza se la coppia diventa eccessiva.

- **Open applications** - Se si abilita questa opzione allora all'avvio e alla chiusura verranno avviate e chiuse anche tutte le applicazione con nome Theremino\_xx.exe. Le applicazioni da avviare devono essere situata nella stessa cartella e nelle sottocartelle del file Theremino\_Cobot.exe.

Solitamente è bene creare una cartella "Apps" per contenere tutte le applicazioni da avviare.

Se si utilizza la applicazione Theremino\_Automation allora è meglio lasciare ad essa anche questo compito **Nota 1** e disabilitare questa opzione.

- **Pause Slot** - Si utilizza per comandare la pausa con un pulsante esterno. Impostando un numero di Slot valido l'esecuzione viene messa in pausa se il valore di tale Slot supera il 500. L'esecuzione riparte quando il valore scende sotto al 500. Impostando -1 questa opzione è inattiva.

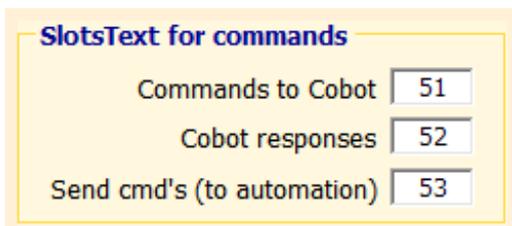
## Nota\_1

Se si utilizza la applicazione Theremino\_Automation allora si ottiene un funzionamento più robusto pilotando tutto da Automation e disabilitando le opzioni **Automatic home** e **Open applications**. Inoltre con Automation si potrebbero gestire meglio anche gli eventi di disconnessione e mandare un comando **StopExecution** quando necessario. In tal caso è bene disabilitare anche l'opzione **Stop if disconnected**.

## Nota\_2

La tensione dei motori viene letta dagli Slot relativi ad ogni motore. I valori degli Slot sono nella configurazione.

## Pannello opzioni - SlotsText for Commands



**SlotsText for commands**

Commands to Cobot

Cobot responses

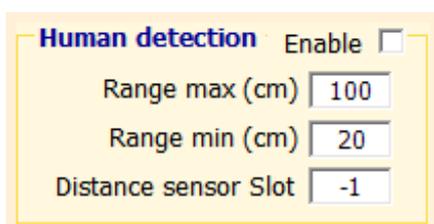
Send cmd's (to automation)

Qui si impostano gli Slot Text **Nota 1** per ricevere comandi da applicazioni esterne, per dare risposte e anche per comandare applicazioni esterne (solitamente Theremino\_Automation).

Se non li si utilizzano è bene disabilitarli impostandoli con il valore -1

**Nota\_1** - Gli Slot Text servono per comunicare messaggi di testo tra applicazioni.

## Pannello opzioni - Human detection



**Human detection** Enable

Range max (cm)

Range min (cm)

Distance sensor Slot

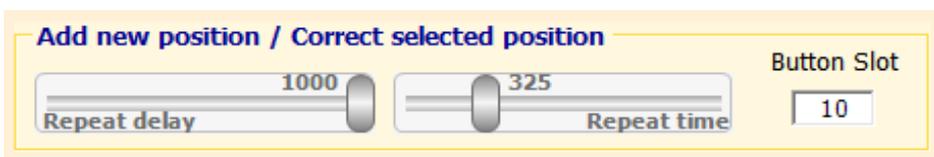
Con queste opzioni si regolano l'inizio e la fine della zona di sicurezza progressiva. Sotto a Range min la velocità viene limitata al minimo. Sopra a Range max non viene limitata. Tra i due estremi si ha una limitazione progressiva della velocità.

Per far funzionare questo meccanismo si imposta il Distance sensor Slot con un valore di Slot valido e poi si collega a tale Slot un sensore di distanza.

Per la massima affidabilità consigliamo di utilizzare sensori multipli [come spiegato in questa sezione del nostro sito](#).

Aumentando il numero di sensori (anche fino a 17 con poca spesa), e dirigendoli in varie direzioni e si ottiene la certezza che almeno un sensore individuerà la presenza di un umano nella zona di sicurezza.

## Pannello opzioni - Add new position / Correct selected position



**Add new position / Correct selected position**

Repeat delay

Repeat time

Button Slot

Queste opzioni regolano il ritardo iniziale e il tempo di ripetizione (in millisecondi) e lo Slot da utilizzare per un pulsante esterno. Poi si imposterà la applicazione HAL in modo che premendo tale pulsante il valore nello Slot superi il 500.

Quando si premerà il pulsante verrà aggiunta una nuova riga alla lista dei comandi con la posizione attuale di tutti i motori, e se lo si tiene premuto a lungo verranno aggiunte molte righe con ritardo e cadenza regolate dai due cursori.

Se c'è una riga selezionata (evidenziata con lo sfondo blu) il pulsante esterno la modifica invece di aggiungere una nuova riga.

Per disabilitare questa opzione si imposta lo Slot con il valore -1

# Capire e modificare la configurazione

Molte operazioni che spiegheremo nelle prossime pagine dipendono dai valori scritti nella configurazione. Nelle prossime pagine spiegheremo le sezioni della configurazione, una per una.

## Tipo di motore e parametri per ognuno dei motori

**Motor** - Il numero di righe che iniziano con "Motor" determina il numero di motori

**Name** - Il tipo di motore determina alcune caratteristiche per i movimenti

**Slots** - Gli slot di questo motore iniziano da questo **numero base**

**Min** - Limita il movimento del motore (deve essere minore di Max)

**Max** - Limita il movimento del motore (deve essere maggiore di Min)

```
=====
= Type      Name      Slots  Min    Max
=====
Motor      STS3215   100    -30000 +30000
Motor      STS3215   200    -30000 +30000
Motor      STS3215   300    -30000 +30000
Motor      STEPPER   400     +0      +360
Motor      SERVO     500     +50     +950
'Motor     SERVO     600     +50     +950
'Motor     Sente150  700    -860000 860000
=====
```

## Slot per i parametri dei motori

Questi sono gli incrementi nella mappa dei registri del motore da sommare al **numero base** del motore.

```
=====
= MOTOR SLOT INCREMENTS
=====
SlotDestination 1
SlotPosition    2
SlotVelocity    3
SlotTorque      4
SlotVoltage     5
SlotTemperature 6
SlotMoving      7
SlotCurrent     8
SlotTorqueLimit 20
SlotAccLimit    21
SlotSpeedLimit  22
SlotOffOnCenter 23
=====
```

## Numero di cifre intere e decimali per i numeri della istruzione MoveTo

Esempio con 3 e 2 : `MoveTo +123.45 -123.45 ----- +000.00`

=====

= DIGITS (3 to 9) and DECIMALS (0 to 9) for edit numbers

=====

Digits 3

Decimals 2

## Velocità di regolazione delle posizioni con la rotella del mouse

Modificando questi moltiplicatori si possono ottenere le velocità preferite.

=====

= EDIT SPEED MULTIPLIER

=====

= Normally 1 (with Feetech or Theremino motors)

= Set to 10, 100 or more to increase the mouse wheel effect

=====

EditSpeedMultiplierSHIFT 10

EditSpeedMultiplierCTRL 0.1

EditSpeedMultiplierALT 0.01

EditSpeedMultiplier 1

## Tolleranza per i comandi GotoCenter e Home

Queste è la tolleranza che verrà utilizzata dopo i comandi GotoCenter e GotoHome

=====

= END MOVING TOLERANCE

=====

= Before to update Acc Or Speed and after GotoCenter and Home

=====

EndMovingTolerance 10

## Valori di default per i cursori di Torque, Acceleration e Speed

Quando si fa click con il pulsante destro del Mouse i cursori vanno al valore stabilito da queste righe.

=====

= SLIDERS

=====

SafeTorqueMax 500 ' usually 200 to 400

SafeTorqueDefault 150 ' usually 150

TorqueDefault 500 ' min=1 max=1000

AccDefault 50 ' min=1 max=255

SpeedDefault 3000 ' min=1 max=65535

# Le sequenze di comandi

## Comandi per muovere i motori

<code>MoveTo n n n n n..</code>	I numeri n sono le destinazioni in step, mm o gradi
<code>MoveTolerance n</code>	Il numero n indica la tolleranza in step, mm o gradi
<code>MoveSpeed n</code>	Interpola con velocità step, mm o gradi per secondo
<code>MoveTime n</code>	Interpola con tempo fisso in secondi

Il comando `MoveTolerance` provoca una attesa, alla fine di ogni `MoveTo`, fino a che tutti i motori sono arrivati alla destinazione entro la tolleranza specificata.

L'ultimo comando `MoveTolerance`, `MoveSpeed` o `MoveTime` che si incontra, viene utilizzato per tutte le righe seguenti fino a che non se ne incontra un altro.

I comandi `MoveSpeed` e `MoveTime` provocano una interpolazione dei movimenti. Il percorso verso la destinazione viene inviato cento volte al secondo suddividendo la distanza totale in molti piccoli tratti. In questo modo tutti i motori arrivano alla destinazione insieme.

Con alcuni motori che comunicano piano e non hanno la accelerazione (ad esempio i Sentel) la interpolazione funziona male. Per disabilitarla si scrive `MoveTime 0`

## Regolazione di coppia, accelerazione e velocità

<code>SafeTorque n</code>	Solo per gli smart motor (non steppers e servo)
<code>Torque n</code>	Solo per gli smart motor (non steppers e servo)
<code>Acceleration n</code>	Solo per gli smart motor (non steppers e servo)
<code>Speed n</code>	Solo per gli smart motor (non steppers e servo)

## Controllo del

<code>Pause n</code>	Pausa per "n" secondi
<code>Restart</code>	Riavvia il programma dalla prima linea
<code>Stop</code>	Ferma l'esecuzione del programma

## Comandi di comunicazione con le altre applicazioni

<code>Slot x = n</code>	Scrivi il valore n nello slot x
<code>SendCommand stringa</code>	Invia un comando ad Automation o altre applicazioni

Tutti i numeri indicati con "n" possono anche essere frazionari e per i decimali si può utilizzare indifferentemente il punto o la virgola.

I tempi in secondi possono anche indicare frazioni, fino ai millesimi di secondo.

# I comandi spiegati uno per uno

## MoveTo

Muove i motori verso le destinazioni indicate.

Ecco alcuni esempi:

```
MoveTo +000 +000 +000 ' Tutti e tre i motori a 0 gradi
```

```
MoveTo +360 ---- ---- ' Motore 1 a 360 gradi e i motori 2 e 3 restano dove sono
```

```
MoveTo +89.9 +11.1 +3.00 ' Motori 1, 2 e 3 con una cifra intera e due decimali
```

```
MoveTo -89.9 -11.1 -3.00 ' Gli stessi valori della riga precedente ma in negativo
```

Nella configurazione (spiegata nelle pagine precedenti) si possono indicare quanti decimali e quante cifre intere visualizzare.

Con una buona impostazione del numero di cifre e di decimali si può mantenere un buon allineamento delle colonne per tutti i numeri, anche nei casi di numeri positivi, negativi e con molti decimali.

I trattini indicano che la destinazione del motore è la stessa che già aveva in precedenza. Quindi si impostano i trattini per i motori che non devono muoversi.

## MoveTolerance

La tolleranza viene usata dalle istruzioni `MoveTo` e serve per attendere il completamento dei movimenti di tutti i motori.

Il programma prosegue alla riga successiva solo se la differenza tra la destinazione e la posizione attuale è minore della tolleranza indicata.

Ecco alcuni esempi:

```
MoveTolerance 0.5 ' Attendi che tutti i motori siano a mezzo grado dalla destinazione
```

```
MoveTolerance 15 ' Procedi velocemente approssimando il percorso
```

## MoveSpeed

Il programma esegue ogni istruzione `MoveTo` con velocità prefissate.

Ogni istruzione `MoveTo` invierà ai motori numerose destinazioni intermedie, una ogni dieci millisecondi, fino al completamento del tempo.

Il tempo viene calcolato con la formula:  $\text{Percorso}$  (in unità, cioè step / gradi o millimetri) fratto  $\text{Velocità}$  (in unità per secondo), utilizzando il percorso del motore che ha più strada da fare.

Se tutti i motori sono abbastanza veloci per seguire queste indicazioni allora tutti i motori arriveranno insieme alla destinazione.

Ecco alcuni esempi:

```
MoveSpeed 50 ' Invia molte posizioni intermedie per rispettare la velocità indicata
```

```
MoveSpeed 999 ' Procedi molto velocemente
```

Probabilmente in quest'ultimo caso i motori resteranno indietro e quindi il programma attenderà che tutti i motori siano arrivati entro la `MoveTolerance`, prima di passare alla riga seguente.

## MoveTime

Il programma esegue ogni istruzione `MoveTo` in un tempo prefissato.

Ogni istruzione `MoveTo` invierà ai motori numerose destinazioni intermedie, una ogni dieci millisecondi, fino al completamento del tempo.

Se tutti i motori sono abbastanza veloci per seguire queste indicazioni allora tutti i motori arriveranno insieme alla destinazione.

Ecco alcuni esempi:

```
MoveTime 2 ' Interpola con molte posizioni intermedie per due secondi
```

```
MoveTime 0.01 ' Procedi molto velocemente.
```

Probabilmente in quest'ultimo caso i motori resteranno indietro e quindi il programma attenderà che tutti i motori siano arrivati entro la `MoveTolerance`, prima di passare alla riga seguente.

Con alcuni motori che comunicano piano e non hanno la accelerazione (ad esempio i Sentel) la interpolazione funziona male e il motore va a scatti.

Per disabilitarla si scrive `MoveTime 0`

## SafeTorque

Questa istruzione imposta il cursore SafeTorque che regola la coppia di sicurezza. Comando valido solo per gli Smart Motor (non stepper e servo).

Attualmente questo comando è in costruzione e dipende molto dal tipo di motori utilizzati. Non in tutti i casi funzionerà come previsto.

Ecco alcuni esempi:

```
SafeTorque 300 ' Tutti i motori con coppia di sicurezza 300
```

## Torque

Questa istruzione imposta il cursore Torque che invia la coppia massima a tutti i motori. Comando valido solo per gli Smart Motor (non stepper e servo).

Ecco alcuni esempi:

```
Torque 300 ' Tutti i motori con coppia massima 300
```

## Acceleration

Questa istruzione imposta il cursore Acceleration che invia la accelerazione a tutti i motori. Comando valido solo per gli Smart Motor (non stepper e servo).

Ecco alcuni esempi:

```
Acceleration 20 ' Tutti i motori con accelerazione 20
```

## Speed

Questa istruzione imposta il cursore Speed che invia la velocità a tutti i motori. Comando valido solo per gli Smart Motor (non stepper e servo).

Ecco alcuni esempi:

```
Speed 1000 ' Tutti i motori a velocità 1000
```

## Pause

Questa istruzione ferma l'esecuzione per il tempo indicato in secondi e frazioni.

Ecco alcuni esempi:

```
Pause 12 ' Attendi per 12 secondi
```

```
Pause 0.5 ' Attendi per mezzo secondo
```

```
Pause 1.53 ' Attendi per 1 secondo e 53 centesimi
```

## Restart

Questa istruzione riavvia il programma

Ecco un esempio:

```
Restart          ' Riparti dalla prima riga del programma
```

## Stop

Questa istruzione ferma il programma

Ecco un esempio:

```
stop            ' Ferma l'esecuzione del programma
```

## Slot

Questa istruzione scrive un numero immediato in uno Slot

Ecco alcuni esempi:

```
slot 3 = 1000   ' Scrivi il numero 1000 nello Slot 3
```

```
slot 9 = 0.33   ' Scrivi il numero 0.33 nello Slot 9
```

```
slot 999 = 10   ' Scrivi il numero 10 nello Slot 999
```

## SendCommand

Questa istruzione invia una stringa di testo nello SlotText indicato nel pannello delle opzioni con: `Send cmd's (to automation)`

Dopo aver inviato il comando la istruzione SendCommand attende che il comando sia stato ricevuto (cioè che lo SlotText sia di nuovo vuoto).

Ecco alcuni esempi:

```
SendCommand Beep          ' Invia il comando "Beep"
```

```
SendCommand Beep multiple ' Invia il comando "Beep multiple"
```

# Slot e SendCommand

Con questa istruzione si scrivono gli Slot numerici.

Gli Slot sono il centro della comunicazione del sistema theremino e chi legge queste istruzioni **dovrebbe** già sapere cosa sono.

In caso contrario si consiglia di [leggere questa sezione](#) e la seguente sugli SlotText, e magari anche tutta la [pagina sulle comunicazioni](#), dall'inizio alla fine.

Ecco alcuni esempi che scrivono in uno Slot

`slot 1 = 12`

Il valore 12 viene scritto nello Slot 1

`slot 2 = 1000`

Nello Slot 2 viene scritto il valore numerico 1000

E questa è una istruzione che scrive negli SlotText utilizzando la funzione SendCommand che è spiegata nelle ultime pagine di questo documento.

`SendCommand Beep`

Fate attenzione che queste istruzioni hanno una sintassi semplificata rispetto alle istruzioni Slot della applicazione Automation.

- Si scrivono solo valori immediati, non formule o funzioni matematiche.
- Non si utilizzano parentesi.
- Non si leggono valori dagli Slot, si scrivono solo numeri immediati.
- Non si scrivono gli SlotText ma si utilizza la funzione SendCommand spiegata nelle ultime pagine di questo documento

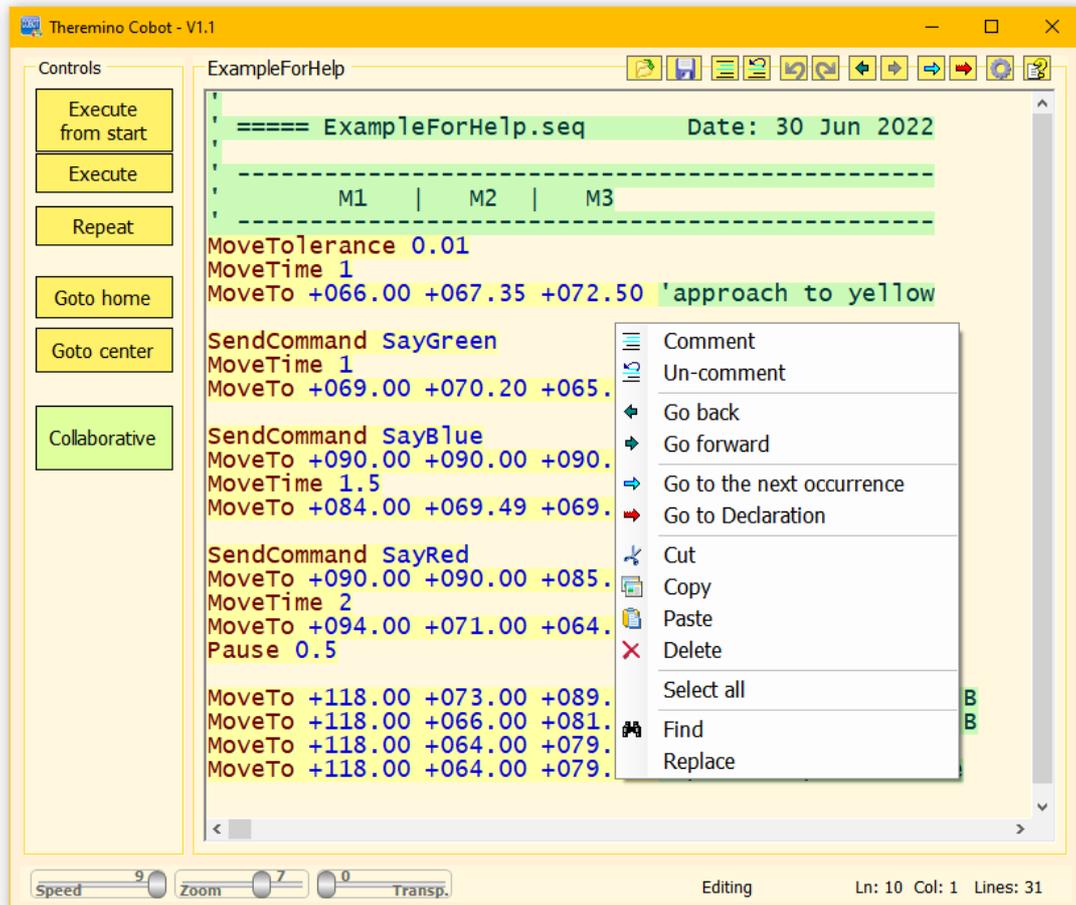
Attenzione a non confondere gli Slot con gli SlotText, hanno indirizzi simili (da 0 a 999), ma scrivono e leggono in zone di memoria diverse.

Inoltre gli Slot contengono numeri (interi o a virgola mobile), mentre gli SlotText contengono stringhe di caratteri (fino a 100000 caratteri).

E infine gli SlotText sono utilizzabili solo per comunicare tra applicazioni e non per comunicare con gli HAL e i moduli Master o Arduino.

# Il menu del programma

Facendo click sulla zona del programma, con il tasto destro del Mouse (oppure toccando lo schermo tattile senza staccare il dito per due secondi), si apre il menu che si vede qui sotto.



**"Comment"** e **"Uncomment"** servono per commentare (aggiungere l'apice iniziale) a intere zone del programma. Oppure per eliminare i commenti.

**"Go Back"** e **"Go Forward"** spostano il cursore, e anche la pagina visibile, sulle sezioni di programma visitate in precedenza.

**"Go to the next occurrence"** cerca altre occorrenze della parola selezionata.

**"Go to declaration"** cerca la parola selezionata solo nelle righe di dichiarazione.

I comandi **"Cut"**, **"Copy"**, **"Paste"**, **"Delete"** e **"Select All"**, copiano, incollano, cancellano e selezionano parti del programma. Al loro posto si potrebbero anche usare i tasti CTRL-X, CTRL-C, CTRL-V, CANC e CTRL-A.

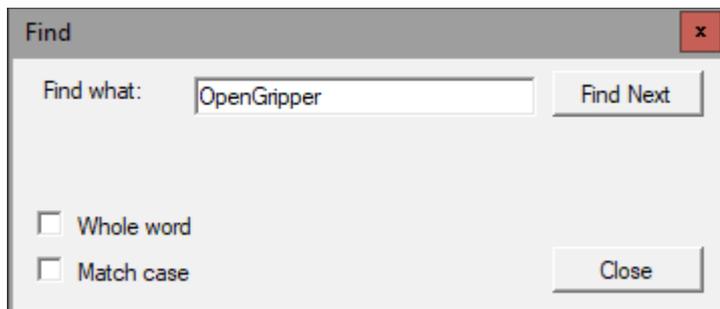
**"Find"** (o CTRL-F) e **"Replace"**, aprono la finestra per cercare e sostituire parole e frasi.

# Find e Replace

Le ultime voci in basso del menu della applicazione aprono due finestre simili tra loro.



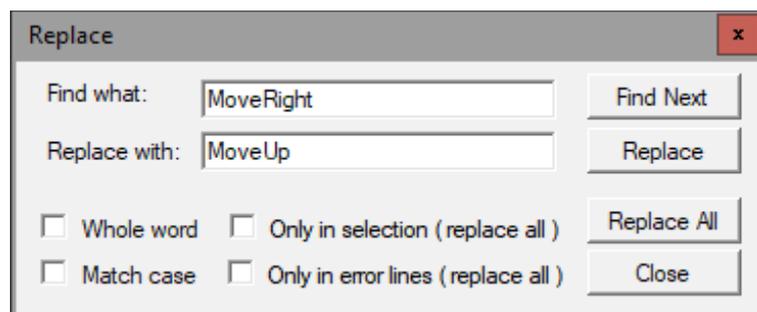
## La finestra "FIND"



Con questa finestra si cercano parole o frasi nel testo del programma.

- ◆ Se si abilita "Whole word", la parola deve essere completa.
- ◆ Se si abilita "Match case", la parola deve corrispondere anche come maiuscole e minuscole.
- ◆ Con "Find next" (o con F3), si passa alla prossima occorrenza della parola cercata. Se si arriva alla fine del programma la ricerca riparte dall'inizio.

## La finestra "REPLACE"

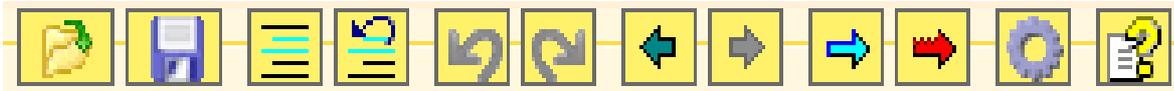


Questa finestra ha le stesse opzioni della precedente, ma permette anche di sostituire la parola (o la frase) con un'altra.

Se si preme "Replace" si effettua una sola sostituzione. Invece con "Replace All" si sostituiscono tutte le occorrenze.

La sostituzione può avvenire in tutto il programma o soltanto nella zona selezionata, oppure soltanto nelle linee che contengono errori (o "warnings").

# I controlli della barra superiore



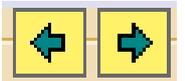
I primi due pulsanti caricano e salvano le sequenze di comandi.



Questi due pulsanti commentano e de-commentano le righe selezionate.



Le due frecce blu servono per tornare indietro nelle modifiche al programma e per ricostruire le modifiche eliminate.



Le due FRECCE scure spostano il cursore, e anche la pagina visibile, sulle sezioni di programma visitate in precedenza.



La FRECCIA azzurra cerca tutte le occorrenze di funzioni, variabili o anche semplici parole.



La FRECCIA rossa cerca solo nelle dichiarazioni (Button, Key, Label e Variable) (ereditata da Automation e attualmente poco utilizzabile).

*Le funzioni di ricerca sono comode, basta selezionare una parola o anche solo posizionare il cursore su di essa e poi premere ripetutamente la freccia.*



L'ingranaggio apre la finestra delle opzioni.

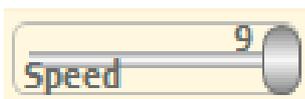


Il punto interrogativo apre il file di istruzioni (Help) nella lingua prescelta. Per far funzionare questo comando si deve copiare i file di Help delle lingue preferite nella cartella "Docs".

Se il file di Help non viene trovato allora appare un messaggio che suggerisce di aprire la cartella Docs e copiarvi il file.

Oppure si può scegliere di selezionare un file di Help nella lingua preferita posizionato nella cartella "Docs" o in qualunque altra cartella. *Per cambiare il file selezionato cliccare il pulsante con il tasto destro del mouse.*

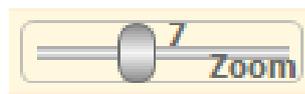
## I controlli della barra inferiore



Questo cursore regola la velocità di esecuzione.

Le velocità vanno da "1" (una istruzione al secondo), fino a "8" (diecimila istruzioni al secondo), e a "9" (la massima velocità permessa dal sistema).

Mentre si scrive il programma è bene utilizzare una velocità media. Di solito la velocità "5" (20 istruzioni al secondo), che è abbastanza lenta da poter seguire visivamente l'esecuzione del programma.



Il cursore ZOOM stabilisce la dimensione del testo, sia nella finestra del programma, sia in quella di Debug.



Questo cursore regola la trasparenza della finestra principale e permette di vedere anche sotto di essa.

Executing line 15

La parte centrale della barra inferiore mostra lo stato attuale che può essere "Executing line nn", "Ready", "Editing" e altri comandi che indicano condizioni speciali. Questi stessi messaggi vengono anche inviati nello Slot di testo **CobotResponses** e possono essere letti da Theremino Automation o da altre applicazioni. Vedere le pagine sulle [comunicazioni con Automation](#).

Ln: 17 Col: 1 Lines: 18

La parte destra della barra inferiore mostra informazioni sul programma:

- Il numero totale di linee
- La linea dove si trova il cursore (partendo da linea 1)
- La colonna dove si trova il cursore (partendo da colonna 1)

# Tecniche avanzate

Come abbiamo già visto la applicazione Cobot è volutamente semplice e questo facilita di molto l'utilizzo nella maggioranza dei casi, ma esistono alcuni casi che non sono gestibili con semplicità e che richiedono tecniche speciali.

Nelle pagine seguenti spiegheremo alcune tecniche che facilitano le seguenti operazioni:

- Effettuare movimenti lineari
- Muovere con le coordinate ortogonali X, Y e Z
- Inviare comandi alla applicazione Theremino Automation
- Ricevere comandi dalla applicazione Theremino Automation o da altre applicazioni del sistema theremino.

# Movimenti lineari

Utilizzando MoveSpeed o MoveTime il movimento viene spezzato in molti piccoli segmenti, tutti i motori vengono guidati cento volte al secondo (se la linea di comunicazione utilizzata lo permette) e tutti arriveranno insieme alla destinazione.

MoveSpeed e MoveTime migliorano il controllo del percorso ma se la destinazione è lontana si ottengono comunque movimenti visibilmente curvi. Se è necessario minimizzare questo problema si possono utilizzare le tecniche seguenti:

- Diminuire MoveSpeed (o aumentare MoveTime) per controllare se le curve indesiderate sono prodotte dalla troppa velocità. Per alcuni motori, che devono fare un movimento maggiore di altri, la velocità potrebbe essere maggiore del limite impostato nei motori (nell'HAL per gli Stepper o nei cursori della applicazione Cobot per i motori programmabili). In questi casi alcuni motori restano indietro e il percorso segue curve ancora maggiori di quelle inevitabili, causate dalla geometria della meccanica.
- Se diminuire la velocità non basta allora si consiglia di alzarla nuovamente e poi spezzare i movimenti in più righe per creare punti di percorso più ravvicinati. Si devono quindi impostare brevi tappe dove si vogliono movimenti e posizioni precise e viceversa tappe lontane una dall'altra nelle zone dove non importa la precisione del percorso.
- Posizionare le tappe del percorso in punti adeguati. Un punto dovrà essere all'inizio della zona dove si vuole avere la massima precisione e gli altri, vicini tra loro, lungo tutta la zona che deve essere precisa.

## Estensione 3D

In alcuni casi i metodi proposti potrebbero non bastare, ad esempio per effettuare una saldatura lunga e perfettamente dritta.

Stiamo quindi preparando anche la possibilità di effettuare i calcoli tridimensionali e quindi specificare la posizione della punta del braccio nelle tre coordinate spaziali X, Y e Z, ma tenete conto che:

- La configurazione delle caratteristiche diventerà notevolmente più complessa e anche la gestione del braccio non sarà più semplice come è ora.
- Se le richieste saranno numerose probabilmente questa opzione sarà disponibile nel 2023.

## Comandi da Cobot a Automation

In Automation la `Label Event_CommandsFromCobot` viene utilizzata per ricevere comandi dalla applicazione `Theremino_COBOT`, ma potrebbe ricevere comandi anche da qualunque altra applicazione in grado di scrivere negli "Slot di testo".

Quando una applicazione esterna scrive una stringa con un comando nello `Slot_CommandsFromCobot` l'evento viene notificato chiamando questa `Label`.

Il programma di Automation viene interrotto, qualunque cosa stesse facendo e le istruzioni dopo alla `Label` vengono eseguite fino al `Return`.

Il testo del comando si legge con la funzione `"CommandText"` e poi lo si decodifica con una struttura `"Select"`, come nell'esempio seguente:

```
Variable Numeric Slot_CommandsFromCobot = 53
Stop

Label Event_CommandsFromCobot
  Select CommandText
    Case "Beep"
      ExecBeep
    ,
    Case "Beep multiple"
      ClearCommand
      ExecBeepMultiple
    ,
    CaseElse
      Print "Unrecognized command: " + CommandText
  EndSelect
  ClearCommand
Return

Label ExecBeep
  Beep 440 300
  Wait Seconds 0.5
Return

Label ExecBeepMultiple
  For v1 = 1 To 3
    Beep 880 400
    Wait Seconds 0.5
  Next
Return

Label ClearCommand
  SlotText(Slot_CommandsFromCobot) = ""
Return
```

Notare che nel `Case "Beep multiple"` è stata aggiunta una istruzione `ClearCommand` prima della esecuzione del comando. Questa istruzione dice alla applicazione COBOT di proseguire immediatamente, senza attendere la fine della esecuzione del comando.

Nella prossima pagina c'è un esempio di sequenza che invia questi comandi.

# I comandi da COBOT a Automation

Questo capitolo spiega come inviare comandi dalla applicazione COBOT verso la applicazione Automation. L'esempio seguente è una sequenza che muove tre motori e tra un movimento e il successivo invia i comandi "Beep" e "Beep multiple".

```
MoveTolerance 1
MoveTime 0.5
MoveTo +00137.000 +00300.000 +00022.000
SendCommand Beep
MoveTo +00153.000 +00303.000 +00025.000
SendCommand Beep multiple
```

Il file che contiene questa sequenza viene eseguito dalla applicazione Theremino\_COBOT e deve stare nella sua cartella "Sequences".

I comandi che si inviano alla applicazione Automation possono essere una qualunque stringa di testo, anche separata da spazi, e vengono decodificati senza tenere conto delle maiuscole o minuscole.

Eventuali caratteri TAB o Spazio multipli vengono trasformati in spazi singoli dalla applicazione COBOT prima di inviarli e i TAB e gli spazi iniziali e finali vengono eliminati.

L'utente stesso può scrivere i suoi comandi, ricordando che:

- I comandi vanno scritti nella sequenza della applicazione COBOT.
- Le istruzioni per decodificarli nell'evento CommandsFromCobot di Automation.

Nella applicazione Cobot si devono impostare gli Slot di testo da utilizzare per i comandi. Aprire il pannello delle opzioni (con l'ingranaggio in alto a destra) e individuare la sezione "Slots for Text-Commands" che si trova in basso.

Gli Slot di testo che si impostano nella applicazione Cobot devono essere gli stessi che si dichiarano nelle variabili di Automation con i nomi: **Slot\_CommandsToCobot**, **Slot\_CobotResponses** e **Slot\_CommandsFromCobot**.

Si possono anche inviare comandi che premono i pulsanti, inviando il loro nome, questi metodi sono spiegati meglio nella documentazione di Automation.

Vedere anche gli esempi nella cartella  
"Demo Programs \ SlotText Commands"

Nella prossima pagina vedremo che si possono inviare comandi anche dalla applicazione Automation verso la applicazione Cobot.

# I comandi da Automation a COBOT

Per inviare comandi verso la applicazione Cobot si impostano le variabili **Slot\_CommandsToCobot** e **Slot\_CobotResponses** e poi le si utilizzano in Automation (o altre applicazioni) con i comandi che scrivono e leggono gli Slot di testo.

**Comandi da Automation a COBOT** ( stringhe di testo nello **Slot\_CommandsToCobot** )

- **ExitFromEdit** ( esci dallo stato di "edit" )
- **StartExecution** ( uscita automatica dallo stato di "edit" )
- **StopExecution** ( uscita automatica dallo stato di "edit" )
- **GotoHome** ( uscita automatica dallo stato di "edit" )
- **GotoCenter** ( uscita automatica dallo stato di "edit" )
- **EnableRepeat**
- **DisableRepeat**
- **SelectLine nnn**
- **ExecuteSelectedLine**
- **ArrowUP**
- **ArrowDOWN**
- **EnableCollaborative**
- **DisableCollaborative**
- **AddNewPosition**
- **CorrectSelectedPosition**
- **DeleteSelectedLine**
- **SetHomePosition**
- **SetCenterPosition**
- **SetHoldingTorque nnn**
- **SetSafeTorqueLimit nnn**
- **SetTorque nnn**
- **SetAcceleration nnn**
- **SetSpeed nnn**
- **LoadSequence SeqName.seq** ( il nome "xxxxx.seq" non deve contenere spazi )

La applicazione Cobot elimina il testo dallo Slot dopo averlo ricevuto e interpretato. La applicazione che ha inviato il comando deve attendere che il comando sia stato ricevuto prima di inviarne altri.

**Risposte dal Cobot** ( stringhe di testo nello **Slot\_ResponsesFromCobot** )

- **Editing**
- **Ready**
- **Execution running at line nnn**
- **Motors disconnected**
- **Human detected - Speed reduced**
- **Human too near - Execution stopped**
- **Too much torque - Execution stopped**
- **Motors disconnected - Execution stopped**
- **Execution paused**