



EPS

Escola Politècnica

UdG

Superior

Projecte/Treball Fi de Carrera

Estudi: Enginyeria Informàtica. Pla 1997

Títol: Entorn de programació pel robot Staubli mitjançant Matlab, aplicació a un sistema de recol·lecció de peces basat en visió artificial

Document: Memòria

Alumne: Ricard Batllori Niubó

Director/Tutor: Marc Carreras Pérez

Departament: Arquitectura i Tecnologia de Computadors

Àrea: ATC

Convocatòria (mes/any): 09/08

0. ÍNDEX

1. INTRODUCCIÓ.....	4
1.1. Antecedents	4
1.2. Objectius.....	4
1.3. Abast.....	4
1.4. Calendari del Projecte	5
1.5. Estructura de la memòria.....	5
2. ENTORN DE TREBALL.....	6
2.1. Manipulador Stäubli TX60.....	6
2.2. Armari de Control	7
2.2.1. Socket Ethernet	8
2.3. La MCP	8
2.3.1. La Pantalla de la MCP	9
2.3.2 El Teclat de la MCP.....	10
2.4. Càmera de vídeo i Targeta d'Adquisició.....	13
2.5. Llenguatge de Programació VAL3	15
2.6. Programes Stäubli	17
3. COMUNICACIONS	25
3.1. Introducció	25
3.2. Protocol de Comunicació	25
3.3. Funcions del Matlab	26
3.3.1. Connectar	26
3.3.2. ConnectarSim.....	26
3.3.3. Desconnectar.....	26
3.3.4. MoureJoint	27
3.3.5. Inici	27
3.3.6. MourePunt	27
3.3.7. MoureLineal.....	27
3.3.8. MoureManual.....	27
3.3.9. PosEina	27
3.3.10. PosEixos.....	28
3.3.11. Separar	28
3.3.12. Ajuntar	28
3.3.13. ObrirEina	28
3.3.14. TancarEina	28
3.3.15. ModificarVelocitat.....	28
3.4. Funcions del VAL3	29
3.4.1. MovimentJoint	29
3.4.2. MovimentPunt	29
3.4.3. MovimentLineal.....	29
3.4.4. PosEina	30
3.4.5. PosEixos.....	30
3.4.6. Obrir.....	30
3.4.7. Tancar	30
3.4.8. ModVelocitat	30
3.4.9. Separar	30

3.4.10. Ajuntar	30
3.5. Interfície Gràfica	31
3.5.1. Automàtic.....	31
3.5.2. Manual	32
3.5.3. Posició.....	33
3.6. Línia de Comandes.....	33
4. VISIÓ	35
4.1. Segmentació Utilitzant Color	35
4.1.1. Espai RGB	35
4.1.2. Espai HSV.....	36
4.1.3. Transformació entre Espais de Color.....	37
4.2. Programació en Matlab	38
4.3. Programació en C	39
4.3.1. LUT.....	39
4.3.2. Mex-Files	39
4.3.3. CalcLut.....	40
4.4. Localitzar la Posició dels Objectes.....	41
5. DETECCIÓ I RECOL·LECCIÓ DE PECES	43
5.1. Preparació Prèvia.....	43
5.2. Programació de l' Aplicació.....	43
6. CONCLUSIONS	47
6.1. Resum.....	47
6.2. Assoliment dels Objectius	48
6.3. Treball Futur.....	49
7. BIBLIOGRAFIA	50
ANNEX 1. MANIPULADOR STÄUBLI TX60 I ARMARI DE CONTROL CS8C ...	51
A1.1. Robot	51
A1.1.1. Espai de Treball	52
A1.1.2. Càrrega Nominal Transportable i Càrrega Màxima.....	53
A1.1.3. Velocitat Màxima.....	53
A1.1.4. Dimensions	53
A1.1.5. Repetibilitat.....	54
A1.1.6. Amplitud, Velocitat i Resolució	54
A1.1.7. Ambient de Treball	54
A1.1.8. Pes	54
A1.2. Armari de Control	54
A1.2.1. Localització de les Entrades – Sortides	54
A1.2.2. Enllaç Ethernet.....	55
A1.2.3. Port Sèrie.....	56
A1.2.4. Calculador	56
A1.2.5. Alimentació RPS 235.....	57
A1.2.6. Alimentació ARPS.....	57
A1.2.7. Amplificadors	57
A1.2.8. Mòdul de Potència (PSM).....	58
A1.2.9. Targeta RSI	58

ANNEX 2. LENGUATGE DE PROGRAMACIÓ VAL3	60
A2.1. Introducció.....	60
A2.2. Elements del Llenguatge VAL3	60
A2.2.1. Aplicacions	60
A2.2.2. Programes	62
A2.2.3. Les Biblioteques.....	63
A2.2.4. Tipus de Dades.....	64
A2.2.5. Constants.....	75
A2.2.6. Variables	75
A2.2.7. Tasques	77
A2.3. Instruccions de Control de Seqüència	77
A2.4. Instruccions de Moviment.....	78
A2.5. Instruccions de Control del Robot.....	79
A2.6. Instruccions de la Pàgina d'Usuari.....	80

1. INTRODUCCIÓ

1.1. Antecedents

En el laboratori docent de robòtica es disposa d'un manipulador industrial de 6 graus de llibertat de la marca Stäubli, una tarja d'adquisició Meteor i una càmera de vídeo en color. Actualment només està previst utilitzar aquest robot en assignatures específiques de robòtica industrial, en les que es dediquen varies sessions teòriques i pràctiques a l'aprenentatge del seu llenguatge de programació, el VAL3. El fet d'haver d'aprendre aquest llenguatge no permet utilitzar el robot en pràctiques d'assignatures que no estan tant orientades a la robòtica industrial, com l'assignatura de màster Autonomous Robots. En aquesta assignatura es realitzen totes les pràctiques amb l'entorn Matlab. Seria d'interès poder realitzar pràctiques amb el robot i una càmera de vídeo utilitzant únicament Matlab.

1.2. Objectius

Els objectius del projecte proposat són:

- 1- Realitzar un intèrpret de comandes en VAL3 que rebí les ordres a través d'una connexió TCP/IP.
- 2- Realitzar una toolbox de Matlab per enviar diferents ordres mitjançant una connexió TCP/IP.
- 3- Adquirir i processar mitjançant Matlab imatges de la càmera en temps real i detectar la posició d'objectes artificials mitjançant la segmentació per color.
- 4- Dissenyar i realitzar una aplicació amb Matlab que reculli peces detectades amb la càmera.

1.3. Abast

L'abast del projecte inclou:

- Estudi del llenguatge de programació VAL3 i disseny del intèrpret de comandes.
- Estudi de les llibreries de Matlab per comunicació mitjançant TCP/IP, per l'adquisició d'imatges, pel processament d'imatges i per la programació en C.
- Disseny de la aplicació recol·lectora de peces.
- Implementació de:
 - (a) un intèrpret de comandes en VAL3,
 - (b) la toolbox pel control del robot STAUBLI en Matlab i
 - (c) la aplicació recol·lectora de peces mitjançant el processament d'imatges en temps real també en Matlab.

1.4. Calendari del Projecte

La realització del projecte s'ha estructurat de la següent manera tal com es mostra a la figura 1:

- Estudi dels requeriments (desembre 07)
- Anàlisi [formació] (gener 08 – març 08)
- Disseny (febrer 08 – abril 08)
- Implementació (maig 08 – juliol 08)
- Proves (juny 08 – juliol 08)

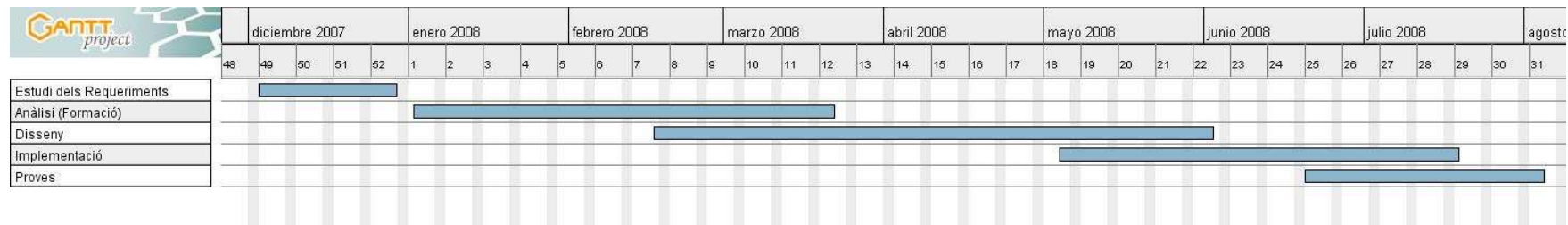


Figura 1. Diagrama de Gantt del projecte

1.5. Estructura de la memòria

Al capítol 2 s'explica l'entorn de treball que es disposa al laboratori de robòtica separant els elements hardware dels elements software. Al capítol 4 s'explica tot el que s'ha fet per tal de poder comandar el robot des del Matlab. Al capítol 4 hi ha l'explicació de com es fa la segmentació per color per tal de localitzar els objectes vistos per la càmera. Al capítol 5 s'explica el funcionament de l'aplicació de recollida de peces. Al capítol 6 s'expliquen les conclusions i el treball futur. Al capítol 7 es citen les fonts que s'han consultat per dur a terme aquest projecte.

Finalment hi ha 2 annexos: El primer explica més a fons el manipulador Stäubli TX60 i l'armari de control i el segon explica més a fons el llenguatge de programació VAL3.

2. ENTORN DE TREBALL

El projecte es desenvolupa al laboratori de robòtica on es disposa, a nivell de hardware, d'un manipulador de 6 graus de llibertat, concretament un Stäubli TX60, d'una càmera a color i d'un PC amb una tarja d'adquisició Meteor la qual està connectada a la càmera mitjançant un cable de vídeo compost. A nivell de software hi ha el Matlab, el llenguatge de programació VAL3 i el programa Stäubli Robotics Studio. Aquests elements es descriuen tot seguit.

2.1. Manipulador Stäubli TX60

El braç manipulador Stäubli TX60 que es pot veure a la figura 2.1 consta d'una cadena cinemàtica de sis sòlids rígids o elements units entre sí mitjançant sis articulacions de revolució (q_1, \dots, q_6) que ens aporten sis graus de llibertat amb els paràmetres següents:

EIX	1	2	3	4	5	6
Amplitud (°)	360	255	285	540	255	540
Distribució d'amplitud (°)	A ± 180	B ± 127.5	C ± 142.5	D ± 270	E $+ 133.5$ $- 122.5$	F ± 270
Velocitat nominal (°/s)	287	287	431	410	320	700
Velocitat màxima (°/s)	373	373	500	968	800	1125
Resolució angular ($^\circ \cdot 10^{-3}$)	0.057	0.057	0.057	0.114	0.122	0.172



Figura 2.1. Braç manipulador Stäubli TX60 utilitzat en el desenvolupament del projecte

2.2. Armari de Control

L'armari de control CS8C, tal com es pot observar a la figura 2.2 i a la figura 2.3, està format per un calculador (5), part intel·ligent de la instal·lació. El calculador guia al robot a través d'amplificadors de potència numèrics (1) dedicats a cada eix del braç. La conversió d'energia elèctrica l'efectuaran el bloc de potència PSM (7), l'alimentació RPS (2) i l'alimentació ARPS (3) que subministren els elements indicats més amunt les tensions necessàries pel seu correcte funcionament a partir de la tensió subministrada per la xarxa elèctrica.

Les indispensables funcions de seguretat elèctrica estan reunides a la targeta RSI (4).



Figura 2.2. Part del darrere de l'armari de control CS8C

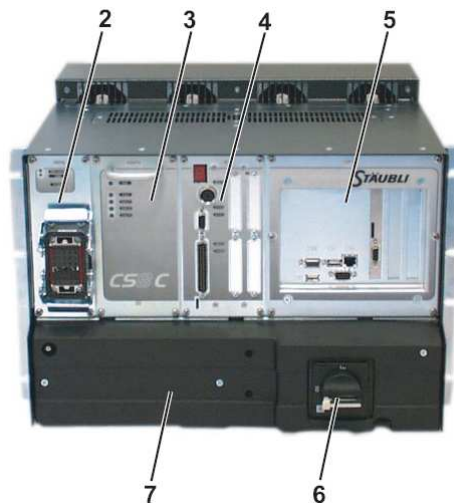


Figura 2.3. Part davantera de l'armari de control CS8C

La posada fora de tensió s'obté posant l'interruptor general (6) situat a la part del davant de l'armari de control en la posició 0. Només s'ha d'accionar quan s'ha aturat el funcionament del braç i s'ha tallat la potència de robot.

2.2.1. Socket Ethernet

El CS8C disposa de 2 ports Ethernet la direcció IP dels quals es pot modificar des del panell de control. També es pot obtenir automàticament una direcció IP mitjançant el protocol DHCP.

L'armari de control CS8C pot ser configurat per comunicar-se a través d'Ethernet mitjançant sockets. Suporta fins a 40 sockets simultanis en mode client i/o en mode servidor però no es suporten els sockets UDP.

Els paràmetres d'un socket servidor són:

- El port de connexió entre 0 i 65535.
- El nombre màxim de clients simultanis.
- El temps previ a l'activació d'error (temps màxim d'espera en lectura o en connexió). Un valor nul elimina el control del temps d'espera.
- El caràcter de final de cadena.

Els paràmetres d'un socket client són els mateixos amb, a més a més, la direcció IP del socket servidor. Un menú "Prueba" permet provar la connexió amb el servidor.

Un socket servidor s'activa ("obert") en el CS8C quan l'utilitza un programa VAL3, i es desactiva ("tancat") quan es desconnecta l'últim client. Quan s'arriba a la quantitat màxima de clients per un socket servidor, els altres clients que s'intenten connectar són acceptats però la comunicació és interrompuda immediatament pel servidor.

2.3. La MCP

L'armari de control disposa d'una MCP (Manual Control Pendant) que permet a l'usuari el control sobre ella. La MCP està formada per una pantalla i un teclat tal com es pot observar a la figura 2.7.



Figura 2.4. Manual Control Pendant

2.3.1. La Pantalla de la MCP

La pantalla de la MCP es divideix en tres zones tal com mostra la figura 2.5:

La Barra d'Estat

La barra d'estat (A) proposa les informacions següents sigui quin sigui l'estat de la navegació en curs:

- Indicador d'activitat del sistema (1). Quan l'indicador està present a la barra d'estat, el sistema no es troba disponible per l'operador.
- Indicador de presència de nous missatges d'informació (2). La seva presència indica que un o diversos missatges nous d'informació han estat emmagatzemats al registre d'esdeveniments. Aquest indicador apareix sempre parpellejant i es manté actiu mentre l'usuari no s'hagi adonat d'aquestes informacions.
- Indicador d'entrada (3). Apareix en mode parpelleig quan una aplicació VAL3 està en espera d'una entrada d'operador a la pàgina d'aplicació. Es manté actiu mentre l'aplicació estigui activa i no s'hagi efectuat l'entrada.
- Indicador de funcionament de l'autòmat programable (PLC) (4).
- Indicador de la velocitat de desplaçament del braç (5). S'aplica a tots els moviments (manuais i programats).
- Indicador de velocitat màxima en mode de prova.

La Pàgina de Treball

La pàgina de treball (B) és la zona de la pantalla situada entre la barra d'estat i la zona de menús contextuais. És damunt d'aquesta pàgina on s'intercanvien totes les informacions relatives a l'aplicació en curs (visualització, finestres d'informacions, entrades). La pàgina de treball té sempre un títol situat a la línia immediatament a sota de la barra d'estat.

Els Menús Contextuais

Els menús contextuais (C) permeten efectuar una acció específica a l'element seleccionat o a la pàgina de navegació. Per activar l'acció, n'hi ha prou amb prémer la tecla situada a sota de l'etiqueta corresponent.

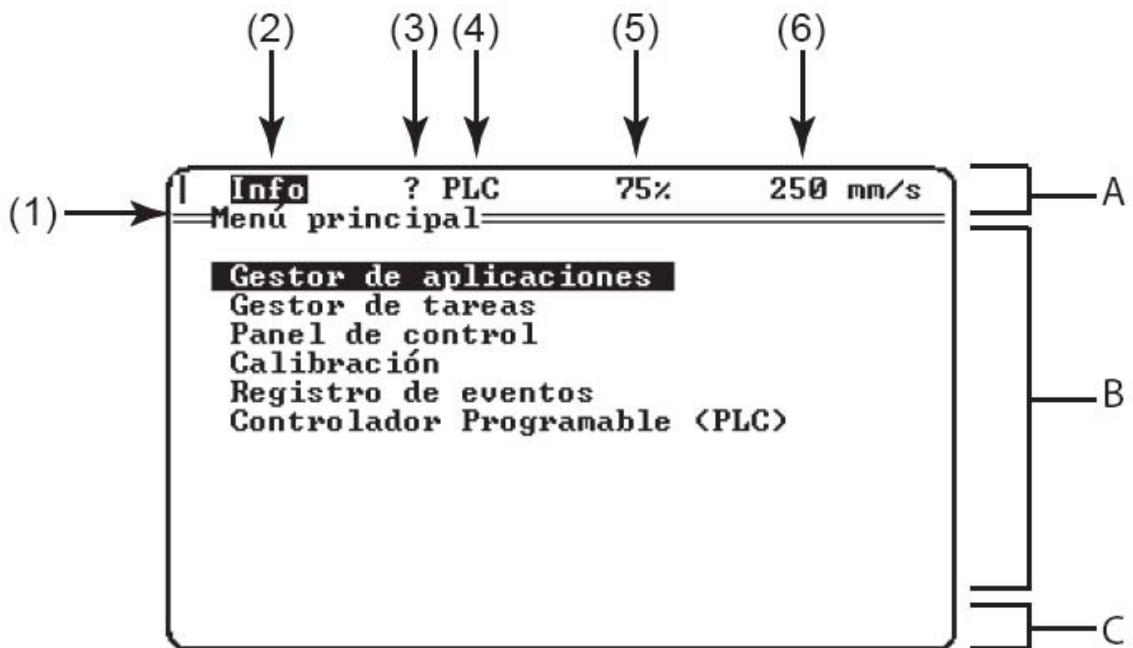


Figura 2.5. Pantalla de la MCP

2.3.2 El Teclat de la MCP

Tot seguit es farà una explicació de les principals funcions del teclat de la MCP segons la distribució que es pot veure a la figura 2.6.



Figura 2.6. Distribució del teclat de la MCP

Botó de Selecció del Mode de Marxa (1)

Aquest polsador permet seleccionar un dels 4 modes de marxa (mode prova, mode manual, mode local i mode desplaçat). El mode seleccionat està indicat al voltant del botó davant dels pictogrames dels modes de marxa.

Botó d'Aplicació de Potència del Braç (2)

Aquest botó lluminós permet aplicar o tallar la potència al braç. L'indicador verd encès fix indica que l'aplicació de potència és efectiva. En mode manual o en mode de prova, si la MCP no està col·locada sobre del seu suport, s'ha d'utilitzar el botó de validació (11).

Botó de Validació (11)

Aquest botó té tres posicions i acciona contactes que estan:

- Oberts quan el botó no està activat.
- Tancats a la posició mitja.
- Oberts en una posició completament enfonsada que correspon a una crispació de l'usuari. Aquests contactes es mantindran oberts fins que es deixi anar el botó.

Aquest botó permet autoritzar l'aplicació de potència al braç en mode manual solsament quan està a la posició mitja. Les altres 2 posicions impedeixen l'aplicació de potència o provoquen el tall de potència si el braç està sotmès a tensió en el mode manual. En el mode automàtic, la posició del botó no es té en compte.

Aturada d'Emergència (3)

L'aturada d'emergència només s'ha d'utilitzar en cas de necessitat absoluta per una aturada no prevista en una aplicació.

Tecles de moviments (4)

Aquestes tecles estan actives en el mode manual i permeten generar moviments del braç per eix o en els punts de referència segons el mode de desplaçament escollit.

Tecles d'elecció del mode de desplaçament (5)



Quan el braç està sotmès a potència i en el mode manual, cadascuna d'aquestes 4 tecles permet seleccionar el mode de desplaçament escollit (Joint, Frame, Tool o Point). L'indicador associat a la tecla indica el mode en curs.

Tecla reglatge de velocitat (6)



Aquesta tecla permet fer que variï la velocitat dins dels límits imposats pel mode de desplaçament. Pot ser desactivada segons el perfil d'usuari en curs. La indicació de la velocitat es mostra a la barra d'estat de la pantalla de la MCP.

Tecles de menús contextuais (7)

S'utilitzen per validar els menús contextuais mostrats al seu damunt.

Tecles alfanumèriques (8)

Aquestes tecles serveixen per entrar dades a l'aplicació.

Tecles d'interfície i de navegació (9)



Combinada amb determinades tecles permet utilitzar altres funcions.



Interromp l'entrada i restitueix el valor inicial del camp o surt de la pàgina en cus.



Una pressió sobre la tecla Help permet accedir en qualsevol moment a l'ajuda en línia. En el mode "ajuda en línia", les funcions dels menús contextuais estan desactivades. En canvi, la pressió sobre una de les tecles de menú contextual acciona la visualització d'una finestra d'explicació sobre la funció que tingui associada.



Aquesta tecla permet tornar al menú principal. Segons el perfil d'usuari, pot estar inactiva.



Una pressió sobre aquesta tecla fa aparèixer la pàgina d'aplicació VAL3.



Permet passar ràpidament d'un camp a un altre.



Aquestes tecles, a més de les funcions de navegació clàssiques, posseeixen algunes funcions que pertanyen exclusivament a la interfície de l'armari de control CS8C.



Permet accedir a la segona funció de la tecla.



Aquesta tecla posseeix la funció clàssica d'esborrat del caràcter situat a l'esquerra del cursor.

Tecles de comandament de les aplicacions (10)



Aquestes tecles s'utilitzen per posar en marxa / aturar una aplicació i per validar els moviments del braç.

Tecles d'activació de sortides digitals (12)



En mode manual, aquestes tecles fan que es commuti l'estat de les sortides digitals que hi estan associades.

Tecles de jog (13)



Aquestes tecles estan actives en el mode manual i permeten generar moviments del braç, per eix o en els punts de referència segons el mode de desplaçament escollit (Joint, Frame, Tool), amb una sola mà.

2.4. Càmera de vídeo i Targeta d'Adquisició

La càmera de vídeo que s'utilitzarà en aquest projecte (figura 2.7) és una càmera de Sony SSC-DC198P que disposa d'un CCD de transferència interlineal tipus 1/3. Té un sistema d'exploració de 625 línies. Té una resolució de 768 x 576 píxels i pot adquirir imatges a 25 fps. El sistema de senyal que utilitza és el PAL estàndard i transmet les imatges per cable de vídeo compost.



Figura 2.7. Càmera de vídeo que s'ha utilitzat al projecte

Per tal de poder adquirir les imatges de la càmera, s'ha instal·lat a l'ordinador una targeta d'adquisició Matrox Meteor II PCI (figura 2.8), que ens permet processar les imatges enviades per la càmera.



Figura 2.8. Targeta d'adquisició d'imatges Matrox Meteor II

Està dissenyada per capturar tant imatges en color com en blanc i negre. Pot transferir les imatges adquirides a la memòria de l'ordinador per tal de processar-les o a la targeta gràfica per tal de mostrar-les directament. Disposa d'una interfície sèrie RS-232 per si es volgués controlar remotament una càmera. Els formats de vídeo que suporta són els estàndards PAL i NTSC. Disposa d'una memòria RAM de 4 MB.

2.5. Llenguatge de Programació VAL3

2.5.1. Introducció

El llenguatge VAL3 és un llenguatge de programació d'alt nivell creat pel comandament dels robots Stäubli a l'àmbit d'aplicacions industrials de manipulació i d'assemblatge.

2.5.2. Elements del Llenguatge VAL3

Els elements constitutius del llenguatge VAL3 són:

- Les aplicacions
- Els programes
- Les biblioteques
- Els tipus de dades
- Les constants
- Les variables (globals, constants, paràmetres)
- Les tasques

Aplicacions

Una aplicació VAL3 és un software autònom de programació del robot i de les entrades-sortides vinculades a un CS8 i està constituïda pels elements següents:

- un conjunt de programes: les instruccions VAL3 que s'han d'executar
- un conjunt de variables globals: les dades de l'aplicació
- un conjunt de biblioteques: les instruccions i les dades externes utilitzades per l'aplicació

Quan una aplicació s'està executant, també conté:

- un conjunt de tasques: els programes en curs d'execució

Programes

Un programa és una seqüència d'instruccions VAL3 per executar i està constituït pels següents elements:

- una seqüència d'instruccions: les instruccions VAL3 que s'han d'executar
- un conjunt de variables locals: les dades internes del programa
- un conjunt de paràmetres: les dades facilitades al programa quan es crida

El programa **start()** és el programa que es crida quan es posa en marxa una aplicació VAL3 i no pot tenir paràmetres.

El programa **stop()** és el programa que es crida quan s'atura una aplicació VAL3 i tampoc pot tenir paràmetres.

Les Biblioteques

Una biblioteca VAL3 és un software reutilitzable per una aplicació o altres biblioteques VAL3 que està constituïda pels elements següents:

- un conjunt programes: les instruccions VAL3 que s'han d'executar
- un conjunt de variables globals: les dades de la biblioteca
- un conjunt de biblioteques: les instruccions i dades externes utilitzades per la biblioteca

Quan la biblioteca s'està executant, també pot contenir:

- un conjunt de tasques: els programes propis de la biblioteca que s'està executant

Tipus de Dades

El tipus d'una constant o d'una variable VAL3 és una característica que permet que el sistema controli les instruccions i els programes a la seva disposició. Totes les constants i variables VAL3 tenen un tipus.

El llenguatge VAL3 suporta els següents tipus senzills:

- el tipus *bool*: pels valors booleans (cert / fals)
- el tipus *num*: pels valors numèrics
- el tipus *string*: per les cadenes de caràcters
- el tipus *dio*: per les entrades-sortides tot o res
- el tipus *aio*: per les entrades-sortides numèriques (analògiques o digitals)
- el tipus *sio*: per les entrades-sortides en connexió sèrie i socket Ethernet

Un tipus estructurat és un tipus que reuneix diverses dades separades en camps. Els camps de tipus estructurats són accessibles individualment pel seu nom.

El llenguatge VAL3 suporta els següents tipus estructurats:

- el tipus *trsf*: per les transformacions geomètriques cartesianes
- el tipus *frame*: pels plans geomètrics cartesianes
- el tipus *tool*: per les eines ajustades en un robot
- el tipus *point*: per les posicions cartesianes d'una eina
- el tipus *joint*: per les posicions articulars del robot
- el tipus *config*: per les configuracions del robot
- el tipus *mdesc*: pels paràmetres de desplaçament del robot

Constants

Una constant és una dada definida directament en un programa VAL3, sense declaració prèvia. Una constant té un tipus que ve determinat implícitament pel sistema.

Variables

Una variable és una dada a la qual es fa referència en un programa pel seu nom i es caracteritza per:

- el seu nom: una cadena de caràcters
- el seu tipus: un dels tipus VAL3 descrits anteriorment
- la seva mida: per un vector, el seu nombre d'elements
- el seu abast: el o els programes que poden utilitzar la variable

El nom d'una variable és el d'una cadena de 1 a 15 caràcters entre “a..zA..Z0..9_”.

Tasques

Una tasca és un programa que s'està executant i es caracteritza pels elements següents:

- un nom: un identificador únic de tasca dins la biblioteca o l'aplicació
- una prioritat o un període: paràmetre per la seqüenciació de les tasques
- un programa: punt d'entrada (i sortida) de la tasca
- un estat: actiu o aturat
- la propera instrucció a executar (i el seu context)

2.6. Programes Stäubli

Per tal de crear l'aplicació per comunicar l'armari amb l'ordinador s'han utilitzat un conjunt de programes de la marca Stäubli que m'han permès programar l'interpret amb el llenguatge VAL3 i fer diverses proves de funcionament amb el simulador abans d'executar-ho amb el robot real. Aquests programes estan agrupats en el que s'anomena Stäubli Robotics Studio (SRS) i aquesta versió és concretament la 3.1. Tot seguit s'aniran explicant els diferents programes dels que disposa.

2.6.1. VAL3 Studio

VAL3 Studio (figura 2.9) constitueix un entorn de desenvolupament per les aplicacions VAL3. Permet:

- Crear una aplicació VAL3.
- Modificar una aplicació VAL3 (dades, programes, biblioteques...).
- Imprimir una aplicació VAL3.
- Comprovar la sintaxis d'una aplicació VAL3.

Creació d'Aplicacions VAL3

La creació d'una aplicació es fa basant-se en un model. Els models són aplicacions, emmagatzemades en un directori específic de la cèl·lula (USR/TEMPLATES). Per crear una aplicació, executa el menú “ARCHIVO / APLICACIÓN NUEVA”. Escull un nom per l'aplicació i, tot seguit, un model de la llista, i valida l'opció fent clic sobre “OK”.

Un nom d'aplicació no ha d'excedir els 15 caràcters i ha de començar per una lletra o un subratllat (a-z,A-A,_,_).

La nova aplicació es crea a la cèl·lula en curs (USR/USRAPP).

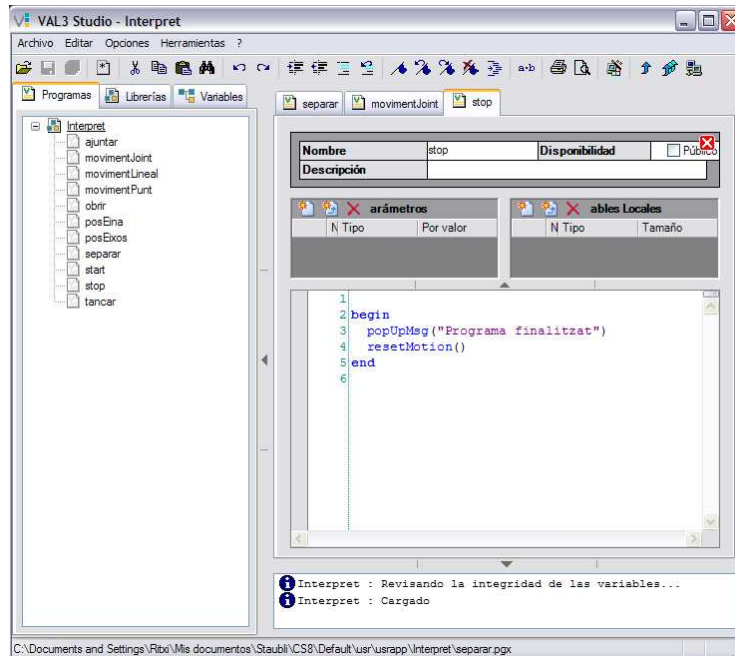


Figura 2.9. VAL3 Studio

Edició de les Dades

A l'arbre de navegació, escull la pestanya “VARIABLES” (ALT+1). Tot seguit fes un doble clic sobre la dada que desitgi. Les dades a l'arbre de navegació estan classificats per ordre alfabètic. Els tipus geomètrics (*frame*, *tool*) estan classificats en un “arbre de pertinença”, és a dir, que un punt de referència conté tots els punts i els punts de referència que es refereixen a ell (ídem en el cas de les eines). Les icones a l'arbre de navegació es mostren agrissats si la dada és privada i normals si la dada és pública.

Per crear una dada nova, fes que aparegui el menú contextual (fes un clic amb el botó de la dreta) en el tipus de dada que es desitgi i, tot seguit, escullí “NUEVO...”. Fes de seguida un doble clic sobre la nova dada per editar-la. Una zona d'edició a la part alta permet modificar les propietats de la dada, o sigui, el seu nom i el seu abast. A sota d'aquesta zona, es presenten els valors de la dada.


Per eliminar una dada, selecciona'l a l'arbre de navegació i, tot seguit, visualitza'l al menú contextual (fes un clic amb el botó de la dreta) i escullí “ELIMINAR” o prem la tecla “DEL”.

En el cas de les dades geomètriques de tipus *point*, *frame* o *tool*, és possible desplaçar-los arrossegant-los amb el ratolí. Si el *frame* o el *tool* desplaçat conté dades, també es desplaçaran.

Edició dels Programes

Per accedir a la llista de programes, escull la pestanya “PROGRAMAS” (ALT+2) a l'arbre de navegació. La icona a l'arbre de navegació està agrisada si el programa és privat i apareix normal si el programa és públic.

L'entrada del codi es fa a través de la zona d'edició central. Aquesta posa color a les paraules clau, gestiona l'auto-decalat, l'acabament (mitjançant l'accés directe des del teclat CTRL+SPACE) i l'ajuda en línia (F1 a la contrasenya). És possible editar diversos programes alhora. Una barra de pestanyes permet navegar entre tots els programes oberts.

Per crear un nou programa, fes clic sobre el botó  de la barra d'eines, o executa el menú “ARCHIVO / NUEVO PROGRAMA” (CTRL+N) o el menú contextual (fes clic amb el botó dret) “NUEVO PROGRAMA” en un programa dins de l'arbre de navegació.

Edició de les Biblioteques

Per accedir a la llista de les biblioteques utilitzades per una aplicació, escull la pestanya “LIBRERÍAS” (ALT+3) a l'arbre de navegació. Cada nus “LIBRERÍA” presenta la llista de les seves dades i programes públics.

Per afegir una biblioteca, premi el botó  situat a la zona “EDICIÓN DE LAS LIBRERÍAS”. Per eliminar una biblioteca, seleccioni la línia corresponent a la biblioteca i premi el botó  situat a la zona “EDICIÓN DE LAS LIBRERÍAS”.

Impressió d'una Aplicació VAL3

Per imprimir una aplicació, selecciona-la a l'arbre de navegació i, tot seguit, escull el menú “ARCHIVO / IMPRIMIR” (CTRL+P) o “ARCHIVO / VISTA PREVIA”. Per obtenir una vista prèvia, és indispensable una versió mínima d'Internet Explorer 5.5.

L'accés ràpid des del teclat CTRL+P en un programa VAL3 només imprimeix el codi del programa.

Comprovació de la Sintaxis

Aquesta eina comprova la sintaxis de cada aplicació i presenta visualment els errors a la finestra de missatges. Al fer la comprovació, les biblioteques són igualment comprovades, encara que no estiguin carregades al VAL3 Studio.

Per comprovar la sintaxis d'una aplicació, escull el menú “COMPRUEBE LA SINTAXIS” en el menú contextual del nus “APLICACIÓN” de l'arbre de navegació. Per comprovar totes les aplicacions carregades al VAL3 Studio, premi la tecla F5.

Si apareixen errors, és possible situar-se al programa, fent un doble clic sobre el primer. Per desplaçar-se d'un error a un altre, premi la tecla F4 o SHIFT+F4.

Si un error es troba en una dada, VAL3 Studio es situarà sobre la dada, però no sobre el valor de la dada que ha generat l'error.

2.6.2. CS8 Emulator

Aquesta eina permet llançar un emulador CS8 (SHIFT + CTRL + E). L'emulador es comporta com un controlador i ofereix la mateixa interfície gràcies a una simulació del comandament manual tal com es pot veure a la figura 2.10.

En la que es pot:

- Carregar, executar i depurar una aplicació VAL3.
- Veure i commutar les E/S.
- Veure els esdeveniments.
- Simular els moviments del braç.

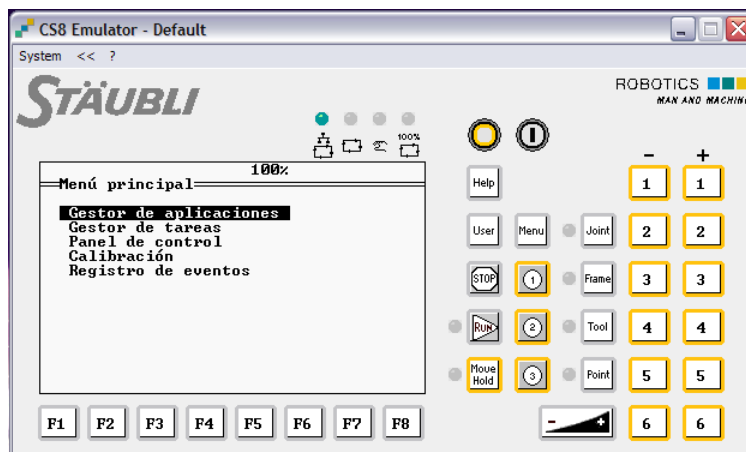


Figura 2.10. CS8 Emulator

2.6.3. 3D Studio

Introducció

3D Studio és un entorn 3D en línia que, connectat a un controlador o a un emulador, permet:

- Veure el robot i els moviments del robot.
- Veure la posició cartesiana actual de l'eina vinculada al robot i la posició articular.
- Veure el traçat del robot.

Les Finestres d'Eines

Totes les finestres d'eines del 3D Studio poden ser flotants o fixes a la vora de la finestra principal. Les finestres són:

- Connexió: Permet administrar la connexió en un CS8 i la freqüència d'actualització.
- Eina: Permet definir l'eina vinculada al robot.
- Cartesiana: Mostra la posició cartesiana actual del robot.
- Angular: Mostra la posició articular actual del robot.
- Pantalla: Permet començar, aturar i esborrar el traçat.
- Accessos directes: Mostra els accessos directes del teclat.

Un cop efectuada la disposició de les finestres, es pot gravar aquesta configuració per les sessions futures. Per això, executar el menú "ARCHIVO / PREFERENCIAS / GUARDAR ORGANIZACIÓN DE VENTANAS".

Per recuperar una configuració de les finestres prèviament guardada, executar el menú "ARCHIVO / PREFERENCIAS / REORGANIZAR VENTANAS".

Connectar-se a un CS8/Emulador

3D Studio funciona únicament en mode en línia. Per connectar-se, fer clic al botó "CONECTAR" de la finestra d'eina "CONEXIÓN". Tot seguit escollir la direcció IP del controlador (o introduir una nova direcció), escriure el nom de l'usuari i la seva contrasenya. Per un emulador, introduir la direcció IP del PC.

La connexió s'efectua en un port TCP. El número del port pot modificar-se mitjançant el menú "ARCHIVO / PREFERENCIAS / PARÁMETROS".

Si el número de port es canvia al 3D Studio, també és necessari actualitzar el fitxer de configuració del controlador (NETWORK.CFX) que es troba al directori /USR/CONFIGS (modificar l'atribut "VALUE" del paràmetre "PORT" a la secció <SOAPSERVER>).

Un cop connectat, la finestra de "CONEXIÓN" pren el nom del robot.

Mostrar la Posició del Robot

Un cop connectat a un controlador, l'actualització de la posició del robot comença automàticament.

La freqüència d'actualització s'ajusta amb el cursor de la finestra de "CONEXIÓN". El "LED" a la dreta del cursor permet adonar-se si se sobrepassa la capacitat d'actualització, és a dir, que el PC no pot dibuixar la vista 3D en el temps assignat. En aquest cas, augmentar l'interval amb el cursor.

Per aturar l'actualització, fer clic al botó "PARAR" de la finestra de "CONEXIÓN".

Per tornar a iniciar l'actualització de la posició del robot, fer clic al botó “COMENZAR” de la finestra de “CONEXIÓN”. A la figura 2.11 es pot veure com mostra la posició actual del robot.

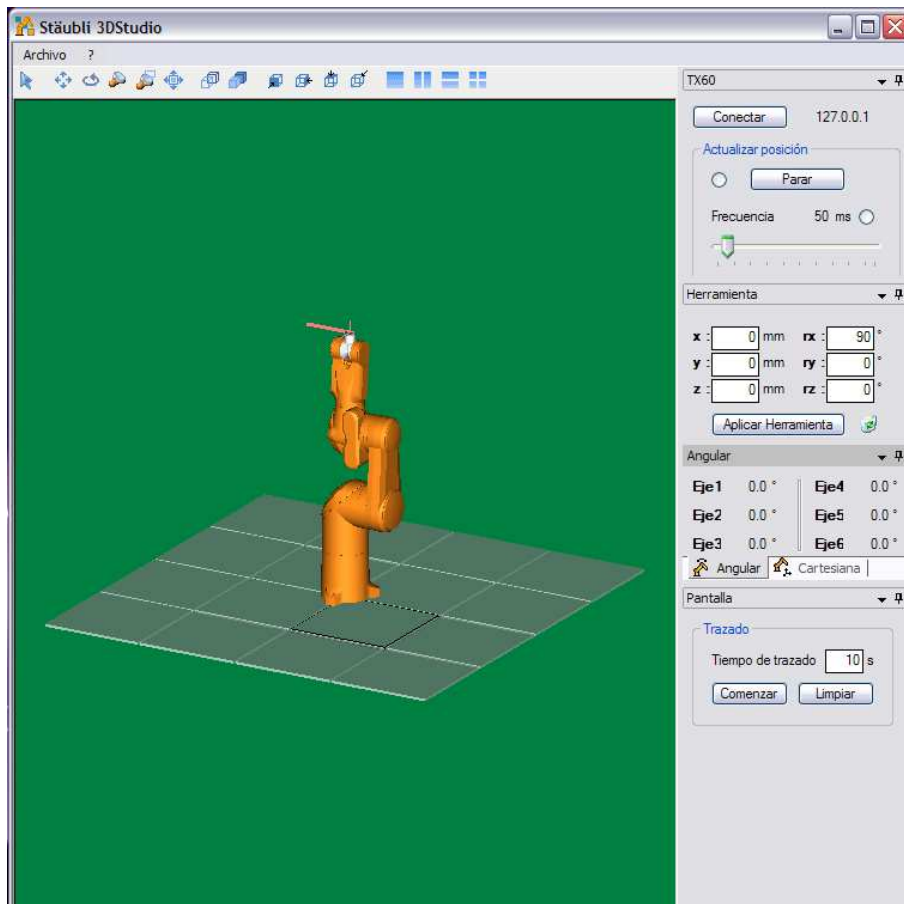



Figura 2.11. 3D Studio

Definir una Eina

Per visualitzar el traçat del robot, s'ha de definir una eina corresponent a l'eina vinculada al robot. Per això, introduir els valors de l'eina a la finestra “HERRAMIENTA” i fer clic al botó “APLICAR HERRAMIENTA”.

A la vista 3D, el símbol  que representa l'eina ha de desplaçar-se en funció dels valors introduïts.

Per anul·lar la definició d'una eina, fer clic al botó a la finestra “HERRAMIENTA”.

Mostrar la Pantalla

Aquest mode de funcionament permet visualitzar el traçat a l'extrem de l'eina sobre els desplaçaments del robot.

Per iniciar la visualització del traçat (pantalla):

- Introduir el temps durant el qual s'ha de visualitzar el traçat (Per exemple: temps de cicle).
- Fer clic al botó “COMENZAR” de la finestra “PANTALLA”. Si es visualitza un traçat antic, s'esborra i comença el nou traçat pel període de temps especificat.

Per aturar la visualització del traçat, prémer el botó “PARAR” de la finestra “TRAZADO”.

Per esborrar el conjunt del traçat, fer clic al botó “LIMPIAR” de la finestra “PANTALLA”.


2.6.4. *Transfer Manager*

L'eina “TRANSFER MANAGER” permet actualitzar una cèl·lula de l'emulador des d'un controlador i al contrari. Executa l'eina a partir de la barra d'eines o del menú “HERRAMIENTAS / TRANSFER MANAGER” (SHIFT + CTRL + T).

Per això és necessària una connexió al controlador. Un cop connectada, l'eina mostra, al panell esquerre, la cèl·lula de l'emulador i, al panell dret, la del controlador.

Per seleccionar elements (aplicacions VAL3, PLC, E/S...) s'ha de fer clic a les caselles de selecció corresponents.

Les fletxes a cada costat del botó “TRANSFERIR” indiquen el sentit en el qual s'efectua la transferència.

 Per transferir des de l'emulador cap al controlador.

 Per transferir des del controlador cap a l'emulador.

Para iniciar la transferència, premi el botó “TRANSFERIR”.

El botó “REFRESCAR” permet posar al dia la vista corresponent.

El botó “BORRAR” permet eliminar els elements seleccionats a la vista.

2.6.5. *CS8 Remote Maintenance*

Aquesta eina permet connectar-se a un controlador a través de la interfície simulada del comandament manual.

1. Per connectar-se, llença l'eina fent clic sobre el botó o utilitzant el menú “HERRAMIENTAS / CONTROLADOR / CS8 REMOTE MAINTENANCE” (SHIFT+CTRL+R).
2. Introdueix les informacions de connexió.
3. Valida.

Un cop connectat, el comandament manual simulat presenta exactament la mateixa interfície que la del comandament manual real. Els 2 comandaments manuals es mantenen connectats en paral·lel, i totes les accions efectuades sobre un d'ells es reflecteixen a l'altre. Les accions que no admet el comandament manual simulat són les següents:

- Mode sotmès a potència.
- Canvi dels modes de funcionament.
- Iniciar una aplicació VAL3 amb el botó “START”.
- Aturar una aplicació VAL3 amb el botó “STOP”.
- Canviar el mode de desplaçament manual.
- El botó “MOVE / HOLD”.
- Les tecles de desplaçament manual.
- Canviar la velocitat.
- Els botons 1, 2 i 3.

Per aturar el telemanteniment, tanca el comandament manual simulat.

Port TCP de Telemanteniment

Per connectar-se a un controlador, l'eina de telemanteniment utilitza un port TCP. Per defecte, aquest port és el 800. En cas que, per raons tècniques, aquest estigués ja utilitzat, és possible canviar-lo.

1. Del costat del PC, canvia el port al sol·licitar les informacions de connexió.
2. Fes clic sobre la casella de selecció “PUERTO”; entra el número de port que es desitgi.
3. Del costat del controlador, entra al “PANEL DE CONTROL / CONFIGURACIÓN DEL CONTROLADOR” i canvia el “PUERTO MANTENIMIENTO REMOTO”. En el moment de procedir a canviar el número de port, si certes eines de telemanteniment estiguessin ja connectades, es tallaria la connexió.

2.6.6. PLC Studio

Aquesta eina és una plataforma de desenvolupament de PLC conforme a la norma IEC-61131. Suporta els llenguatges d'aquesta norma, és a dir: IL (INSTRUCTIONS LIST), ST (STRUCTURED TEXT), SFC (GRAFCET), FBD (FUNCTIONAL BLOC DIAGRAM) i LD (LADDER).

L'eina permet crear i modificar una aplicació PLC, però també fer-se càrrec a distància d'un CS8 que executa un PLC amb l'objectiu de depurar-lo. L'aplicació PLC desenvolupada amb aquesta eina es comporta com un autòmat extern, és a dir, que la comunicació amb el mon VAL3 es fa a través de les E/S de la cèl·lula.

3. COMUNICACIONS

3.1. Introducció

El projecte s'ha dividit en dos blocs o "toolboxes" (si s'utilitza l'expressió del Matlab) bastant diferenciats:

El primer consisteix en crear un protocol de comunicació entre el Matlab i el llenguatge de programació VAL3 per tal que quan s'envii una instrucció des del Matlab, es tradueixi i la pugui interpretar el robot directament. És a dir, es tracta de que el robot es pugui comandar des del propi Matlab a través de la línia de comandes o a través d'una interfície gràfica. Aquest bloc s'ha anomenat **comunicacions**.

El segon bloc es centra en la visió per computador per tal que es pugui saber quina és la posició dels objectes que veu la càmera utilitzant la segmentació per color. Aquest segon bloc s'ha anomenat **visió**.

3.2. Protocol de Comunicació

Les instruccions que s'enviaran des del Matlab a l'armari de control es transmetran a través d'una connexió Ethernet i amb objectes TCP/IP. El llenguatge VAL3 per tal de poder treballar amb connexions Ethernet ha de crear un socket que és on rebrà les dades i a través del qual enviarà les dades. Aquest socket ha de tenir un port concret (en aquest cas 1000) i també un temps d'espera concret a partir del qual si no ha rebut res es tanca la connexió. En aquest cas el valor serà 0, que vol dir que sempre estarà actiu fins que no es tanqui.

El llenguatge VAL3 presenta una limitació en la comunicació a través d'Ethernet, aquesta és que les dades que envia i rep són en format byte, o sigui de 0 a 255 (o de -128 a 128 si s'utilitzen nombres negatius). Les instruccions que s'enviaran a través del Matlab ocuparan més d'un byte, ja que pel sol fet de girar a -160° una articulació del robot ja n'ocuparia més d'un. Per aquest motiu s'ha dissenyat un senzill protocol de comunicacions entre el Matlab i el VAL3. Es tracta de separar el nombre que es vol enviar en 4 bytes de la següent manera:

Primer: signe. 0 si és positiu i 1 si és negatiu.

Segon: quocient de la divisió del nombre entre 255.

Tercer: resta de la divisió del nombre entre 255.

Quart: nombres decimals, que com a molt podrien ser 2 (si fossin més decimals ja no hi cabrien en un byte i per la informació que s'envia, amb precisió de 2 decimals n'hi ha suficient).

Per exemple, el nombre -835,64 seria transmès així:

Primer byte: 1.

Segon byte: $835/255 = 3$.

Tercer byte: $835\%255 = 70$.

Quart byte: 64.

A part d'això a l'hora d'enviar les dades des del Matlab es farà sempre a través d'un vector de 25 posicions (25 bytes). La primera posició serà per fer saber a l'interpret de VAL3 quina instrucció ha d'executar i la resta seran per indicar-li els graus de les 6 articulacions del robot o la posició i l'orientació de la mà del robot. Hi haurà instruccions en les quals només es tindrà en compte la primera posició i la resta seran sempre 0. Això passarà quan les instruccions que s'enviïn no siguin de moviment, per exemple, per saber la posició on es troba o per obrir o tancar l'eina del robot.

En el cas que, per exemple, es volgués enviar el robot a una determinada posició movent-lo per eixos, l'estructura del vector seria així:

Codi instrucció	Eix 1 (1r byte)	Eix 1 (2n byte)	Eix 1 (3r byte)	Eix 1 (4t byte)	Eix 2 (1r byte)	Eix 2 (2n byte)	Eix 2 (3r byte)	Eix 2 (4t byte)	Eix 3 (1r byte)	Eix 3 (2n byte)	Eix 3 (3r byte)	Eix 3 (4t byte)	Eix 4 (1r byte)	Eix 4 (2n byte)	Eix 4 (3r byte)	Eix 4 (4t byte)	Eix 5 (1r byte)	Eix 5 (2n byte)	Eix 5 (3r byte)	Eix 5 (4t byte)	Eix 6 (1r byte)	Eix 6 (2n byte)	Eix 6 (3r byte)	Eix 6 (4t byte)

3.3. Funcions del Matlab

S'han creat tot un conjunt de funcions amb el Matlab, algunes de les quals són per establir comunicació amb l'armari del robot i d'altres que equivalen a instruccions del llenguatge VAL3. Són les següents:

3.3.1. Connectar

Tal com indica el seu nom serveix per poder establir la comunicació entre el Matlab i l'armari de control del robot. El Matlab crea un objecte TCP/IP amb la direcció IP de l'armari (84.88.155.222) i amb el nombre de port on es troba el socket (1000). Un cop creat l'objecte, l'obre, retorna l'objecte i a partir d'aquí ja es poden començar a enviar ordres a través de l'objecte retornat.

3.3.2. ConnectarSim

És com el connectar amb la diferència que en comptes de connectar-se amb l'armari es connecta amb el simulador del robot que està instal·lat al propi ordinador. Ho fa creant un objecte TCP/IP amb la direcció IP local (127.0.0.1) i amb el nombre de port on es troba el socket (1000). Un cop creat l'objecte, l'obre, retorna l'objecte i a partir d'aquí ja es poden començar a enviar ordres a través de l'objecte retornat.

3.3.3. Disconnectar

Envia l'ordre de disconnectar al interpret de VAL3 a través de l'objecte TCP/IP que se li ha passat per paràmetre i llavors el tanca.

3.3.4. MoureJoint

S'envien a través de l'objecte TCP/IP que se li passa per paràmetre els angles on s'han de moure les articulacions del robot. Abans d'enviar-les es transformen els angles que se li han passat per paràmetre utilitzant el protocol de comunicació. A la primera posició del vector s'hi passa un 1 per tal que l'interpret del llenguatge VAL3 sàpiga quina instrucció ha d'executar. Si no es pot moure a la posició desitjada retorna error.

3.3.5. Inici

Fa que el robot es situï a la seva posició inicial. S'envia la mateixa ordre a l'armari que amb la funció moureJoint però amb els valors dels angles a 0.

3.3.6. MourePunt

S'envia a través de l'objecte TCP/IP que se li passa per paràmetre el punt i l'orientació on s'ha de moure la mà del robot. Abans d'enviar-los es transformen els paràmetres utilitzant el protocol de comunicació. A la primera posició del vector s'hi passa un 2. Si no es pot moure a la posició desitjada retorna error.

3.3.7. MoureLineal

S'envia a través de l'objecte TCP/IP que se li passa per paràmetre el punt i l'orientació on s'ha de moure la mà del robot. Abans d'enviar-los es transformen els paràmetres utilitzant el protocol de comunicació. A la primera posició del vector s'hi passa un 3. La diferència que hi ha amb el MourePunt és que obliga el robot a seguir una línia recta a l'hora de desplaçar-se des de l'origen al punt indicat. Si no es pot moure a la posició desitjada retorna error.

3.3.8. MoureManual

Simula el moviment manual del robot com si es fes utilitzant la MCP. Rep per paràmetre el tipus de moviment que ha de fer (angular, cartesià, lineal) i l'increment de moviment que ha de fer, llavors suma l'increment a la posició actual on es troba i crida la funció corresponent per tal de fer el moviment. Si no es pot moure a la posició desitjada retorna error.

3.3.9. PosEina

Retorna la posició i l'orientació de l'eina del robot. Per fer-ho envia l'ordre a l'armari (passant un 4 a la primera posició del vector) a través de l'objecte TCP/IP que rep per paràmetre i espera a rebre les dades. Les dades les rep amb cada component separada en 4 bytes i llavors les ha d'ajuntar per poder-les retornar correctament.

3.3.10. PosEixos

Retorna els angles de les articulacions del robot. Per fer-ho envia l'ordre a l'armari (passant un 5 a la primera posició del vector) a través de l'objecte TCP/IP que rep per paràmetre i espera a rebre les dades. Les dades les rep amb cada component separada en 4 bytes i llavors les ha d'ajuntar per poder-les retornar correctament.

3.3.11. Separar

Transforma un nombre real en 4 bytes per tal de seguir el protocol de comunicació. Si el signe és negatiu el primer byte és 1, sinó és 0. Es divideix el nombre entre 255 i el quocient es posa en el segon byte i la resta en el tercer. Al quart byte s'hi posen els 2 primers decimals.

3.3.12. Ajuntar

És la funció complementària a l'anterior. Transforma els quatre bytes en un nombre real. Si el primer byte és 1 el signe serà negatiu. Multiplicant el segon byte per 255 i sumant-li el tercer es recupera la part sencera del nombre. I amb el quart byte es recupera la part decimal.

3.3.13. ObrirEina

Obre l'eina del robot. Per tal de fer-ho envia l'ordre a través de l'objecte TCP/IP a l'armari passant un 6 a la primera posició del vector.

3.3.14. TancarEina

Tanca l'eina del robot. Per tal de fer-ho envia l'ordre a través de l'objecte TCP/IP a l'armari passant un 7 a la primera posició del vector.

3.3.15. ModificarVelocitat

Serveix per modificar la velocitat i l'acceleració del robot. Se li passen per paràmetre l'acceleració, la desacceleració i la velocitat articular màxima autoritzada i aquestes s'envien a l'armari seguint el protocol de comunicació i passant un 8 a la primera posició del vector per tal que es modifiquin aquests paràmetres.

A part d'aquestes funcions s'ha creat una interfície gràfica senzilla però útil per tal de facilitar al màxim la comunicació amb el robot que s'explicarà més endavant al seu corresponent apartat.

3.4. Funcions del VAL3

Pel que fa al VAL3 s'ha creat una aplicació per tal d'interpretar les ordres que arriben del Matlab i poder-li enviar aquesta informació al robot. Aquesta aplicació consta d'un programa principal i d'un conjunt de funcions que s'aniran cridant quan sigui necessari.

El programa principal està pensat per treballar en el mode remot. Quan comença comprova si la potència del braç robòtic està activada i si no ho està l'activa. Es crea un enllaç entre el socket on es rebran i s'enviaran les dades amb una variable (sio) que el VAL3 pugui treballar que s'anomena dades. S'esborra la pantalla d'usuari i se li canvia el títol a "Intèrpret de Comandes".

A partir d'aquí entra en un bucle del que només sortirà quan se li envii aquesta ordre des del Matlab. Dins d'aquest bucle estarà tota l'estona a l'espera de rebre ordres des del Matlab i depenent de quina rebi executarà una funció o una altra. Seguint el protocol de comunicació aquesta decisió es prendrà segons el primer valor del vector de dades que es rebi:

1. movimentJoint
2. movimentPunt
3. movimentLineal
4. posEina
5. posEixos
6. obrir
7. tancar
8. modVelocitat
50. sortir del bucle

Tot seguit hi ha una explicació de les funcions:

3.4.1. MovimentJoint

Transforma els paràmetres que rep segons el protocol de comunicació en els angles on es mouran les articulacions del robot. Comprova que s'hi pugui desplaçar i s'hi desplaça. En cas contrari envia un error al Matlab.

3.4.2. MovimentPunt

Transforma els paràmetres que rep segons el protocol de comunicació en el punt i l'orientació on es mourà el robot. Comprova que s'hi pugui desplaçar i s'hi desplaça. En cas contrari envia un error al Matlab.

3.4.3. MovimentLineal

Transforma els paràmetres que rep segons el protocol de comunicació en el punt i l'orientació on es mourà el robot. Comprova que s'hi pugui desplaçar i s'hi desplaça. En cas contrari envia un error al Matlab. La diferència que hi ha amb el MovimentPunt és que aquest obliga al robot a moure's des de l'origen al destí seguint una línia recta.

3.4.4. PosEina

Envia al Matlab la posició i l'orientació de l'eina del robot seguint el protocol de comunicació.

3.4.5. PosEixos

Envia al Matlab els angles de les articulacions del robot seguint el protocol de comunicació.

3.4.6. Obrir

Obre la pinça del robot.

3.4.7. Tancar

Tanca la pinça del robot.

3.4.8. ModVelocitat

Modifica l'acceleració, la desacceleració i la velocitat articular màxima autoritzada segons els paràmetres que rep del Matlab.

3.4.9. Separar

Separa els nombres reals en 4 bytes per tal de seguir el protocol de comunicació. Si el signe és negatiu el primer byte és 1, sinó és 0. Es divideix el nombre entre 255 i el quocient es posa en el segon byte i la resta en el tercer. Al quart byte s'hi posen els 2 primers decimals.

3.4.10. Ajuntar

És la funció complementària a l'anterior. Converteix els quatre bytes en un nombre real. Si el primer byte és 1 el signe serà negatiu. Multiplicant el segon byte per 255 i sumant-li el tercer es recupera la part sencera del nombre. I amb el quart byte es recupera la part decimal.

3.5. Interfície Gràfica

S'ha dissenyat una interfície gràfica per tal de poder fer més senzilla encara la comunicació amb el robot Staubli. Per tal d'iniciar aquesta interfície s'ha d'escriure "staubli" a la línia de comandes del Matlab. A continuació s'explica com treballar amb aquesta interfície començant amb els botons comuns als diferents modes de moviment:

- Connectar: Serveix per establir comunicació amb l'armari del robot. Per tal que funcioni hi ha d'haver l'interpret de VAL3 funcionant.
- Desconnectar: Serveix per tancar la comunicació amb l'armari del robot i tancar l'interpret de comandes de VAL3.
- Obrir: Prement aquest botó fa que s'obri la pinça del robot.
- Tancar: Prement aquest botó fa que es tanqui la pinça del robot.
- Guardar: Si es vol guardar una determinada posició per accedir-hi més tard amb el mode de desplaçament "posició" s'ha de guardar prement aquest botó un cop el robot estigui situat a la posició desitjada. Per defecte es van guardant consecutivament (1,2,3,...) però si es desitja es pot guardar al número que es vulgui.
- On Soc: Aquest botó serveix per obtenir les coordenades dels eixos de les articulacions on es troba actualment el robot. És molt útil en el moviment manual.
- Informació: És un quadre de text que informa en tot moment de si hi ha connexió o no amb l'armari de control, si un moviment s'ha pogut fer correctament, si la pinça del robot està tancada, ...

Començant per dalt a l'esquerra hi ha un desplegable que ens permet escollir el mode de moviment que vulguem fer que segons la opció que s'esculli s'aniran modificant la resta de botons de la interfície:

3.5.1. Automàtic

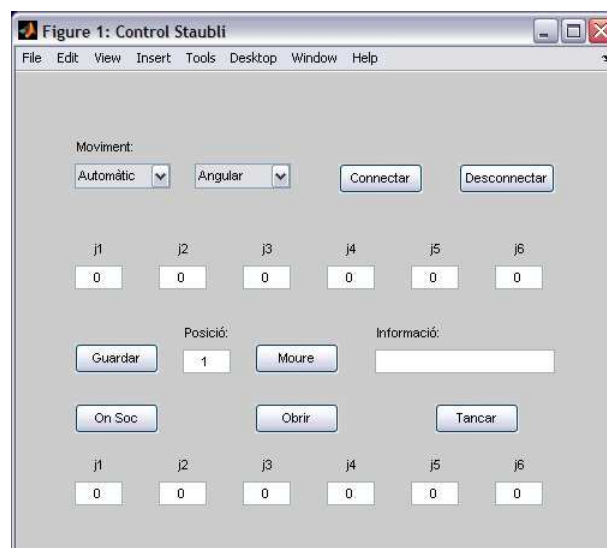


Figura 3.1. Interfície gràfica en el mode automàtic

En el mode automàtic (figura 3.1) si ens fixem en el segon desplegable veurem que es poden escollir 3 tipus més de moviments:

- Angular (els eixos del robot es mouran als angles on s'indiqui).
- Cartesià (la mà del robot es desplaçarà fins el punt indicat amb la orientació indicada).
- Lineal (farà el mateix que el cartesià però obligant la mà del robot a seguir una línia recta des del punt origen al destí).

Per moure el robot a la posició desitjada s'ha de prémer el botó "Moure" un cop s'hagi escrit aquesta posició en els quadres de text corresponents.

3.5.2. Manual

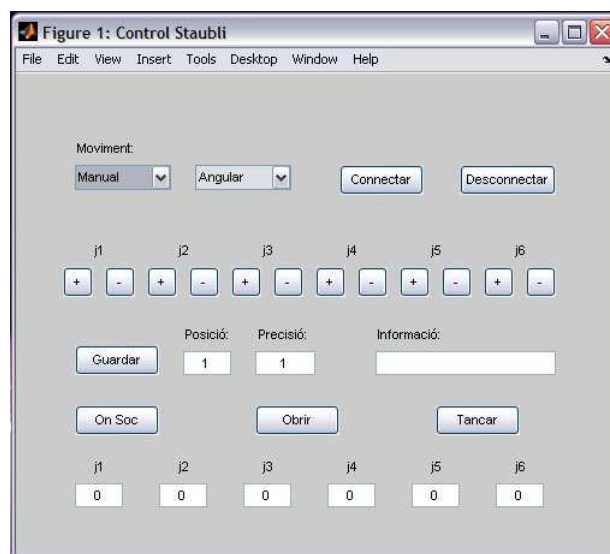


Figura 3.2. Interfície gràfica en el mode Manual

En el mode manual (figura 3.2) els quadres de text on s'indicava cap on havia de desplaçar-se el robot s'han substituït per botons d'increment i de decrement. D'aquesta manera es pot anar movent el robot de manera incremental amb la precisió que es desitgi. Per defecte aquesta és d'una unitat (1 grau en el cas del moviment angular i 1 mm en el cas del moviment cartesià o lineal) però es pot canviar modificant el quadre de text corresponent. Pel que fa als tipus de moviment del segon desplegable, els fa de la mateixa manera que en el mode automàtic però tenint en compte la precisió que s'hagi escollit.

3.5.3. Posició

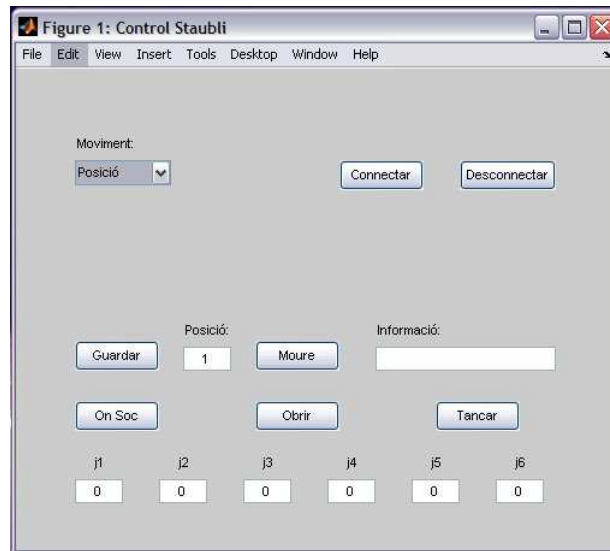


Figura 3.3. Interfície gràfica en el mode posició

En el mode posició (figura 3.3) s’ha de seleccionar alguna de les posicions que s’hagin memoritzat anteriorment en el quadre de text “Posició” i prémer el botó “Moure” per anar-hi directament. Si la posició escollida no existeix, el quadre de text “Informació” ens informarà de l’error. En aquest cas no es pot escollir el tipus de moviment perquè per anar d’una posició a l’altra el robot s’hi desplaçarà de la manera més ràpida possible.

3.6. Línia de Comandes

Mitjançant la línia de comandes del Matlab es poden anar introduint ordres per comunicar-se amb el robot directament sense haver d’utilitzar la interfície gràfica. D’aquesta manera fins i tot es poden programar scripts per tal que el robot segueixi una seqüència de moviments. Un script sempre haurà de començar amb la instrucció “t = connectar;” i haurà d’acabar amb “desconnectar(t);” per tal d’establir la comunicació amb l’armari de control del robot i tancar la comunicació respectivament.

Tot seguit hi ha un exemple en el qual se suposa que s’ha de recollir una peça a la posició (360,30,-360,0,0,0), deixar-la a la posició (500,130,-360,0,0,0) i que no hi ha cap obstacle pel mig:

```
t = connectar; % es connecta a l’armari de control
mourePunt(t,360,30,-300,0,0,0); % es mou fins a una posició de seguretat
modificarVelocitat(t,10,10,10); % es baixa la velocitat per seguretat
moureManual(t,2,0,0,-60,0,0,0); % arriba fins on hi ha la peça
tancarEina(t); % tanca la pinça per agafar la peça
moureManual(t,2,0,0,60,0,0,0); % torna amunt fins a la posició de seguretat
```

```
modificarVelocitat(t,50,50,50); % ja es pot augmentar la velocitat
mourePunt(t,500,130,-300,0,0,0); % es mou fins a una altra posició de
seguretat
modificarVelocitat(t,10,10,10); % es baixa la velocitat per seguretat
moureManual(t,2,0,0,-60,0,0,0); % s'apropa fins on ha de deixar la peça
obrirEina(t); % deixa la peça
moureManual(t,2,0,0,60,0,0,0); % torna amunt fins a la posició de segu-
retat
modificarVelocitat(t,50,50,50); % ja es pot augmentar la velocitat
inici(t); % es mou fins a la posició inicial del robot
desconnectar(t); % es desconnecta de l'armari de control i s'acaba l'aplicació
```

4. VISIÓ

En aquest altre bloc s'ha fet èmfasi a l'ús de la visió per computador fent servir una càmera de vídeo. S'ha creat un script en Matlab que permet saber quina és la localització de 3 objectes que vegi la càmera. També s'ha creat una funció en C per tal de fer més eficients alguns passos que en el Matlab eren molt lents.

4.1. Segmentació Utilitzant Color

4.1.1. Espai RGB

La descripció RGB (de l'anglès Red, Green, Blue; “vermell, verd, blau”) d'un color fa referència a la composició del color en termes de la intensitat dels colors primaris amb els que es forma: el vermell, el verd i el blau. És un model de color basat en la síntesi additiva, amb el que és possible representar un color mitjançant la barreja per addició dels tres colors llum primaris. El model de color RGB no defineix per ell mateix el que significa exactament vermell, verd o blau, motiu pel qual els mateixos valors RGB poden mostrar colors notablement diferents en diferents dispositius que utilitzin aquest model de color. Encara que utilitzin un mateix model de color, els seus espais de color poden variar considerablement.

Per indicar en quina proporció barregem cada color, s'assigna un valor a cadascun dels colors primaris, de manera, per exemple, que el valor 0 significa que no intervé a la barreja i, a mesura que aquest valor augmenta, s'entén que aporta més intensitat a la barreja. Tot i que l'interval de valors podria ser qualsevol (valors reals entre 0 i 1, valors sencers entre 0 i 37, etc.), és freqüent que cada color primari es codifiqui amb un byte (8 bits). Així, la intensitat de cadascuna de les components es mesura segons una escala que va del 0 al 255.

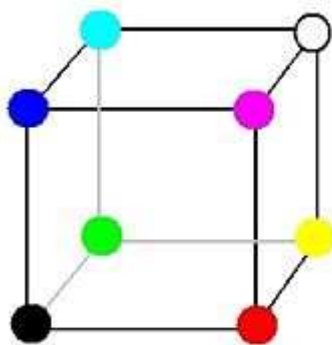


Figura 4.1. Cub RGB

Per tant, el vermell s'obté amb $(255,0,0)$, el verd amb $(0,255,0)$ i el blau amb $(0,0,255)$, obtenint, en cada cas un color resultant monocromàtic. L'absència de color – el que nosaltres coneixem com color negre – s'obté quan les tres components són 0, $(0,0,0)$ tal com es veu al cub de la figura 4.1.

La combinació de dos colors a nivell 255 amb un tercer a nivell 0 dóna a lloc a tres colors intermedis. D'aquesta forma el groc és (255,255,0), el cyan (0,255,255) i el magenta (255,0,255).

Òbviament, el color blanc es forma amb els tres colors primaris al seu màxim nivell (255,255,255).

El conjunt de tots els colors es pot representar en forma de cub. Cada color és un punt de la superfície o de l'interior d'aquest. L'escala de grisos estaria situada a la diagonal que uneix al color blanc amb el negre.

4.1.2. Espai HSV

El model HSV (de l'anglès Hue, Saturation, Value – Tonalitat, Saturació, Valor), també anomenat HSB (Hue, Saturation, Brightness – Tonalitat, Saturació, Brillantor), defineix un model de color en termes de les seves components constituents en coordenades cilíndriques:

Tonalitat, el tipus de color (com vermell, blau o groc). Es representa com un grau d'angle on els seus possibles valors van de 0 a 360° (tot i que per algunes aplicacions es normalitzen del 0 al 100%). Cada valor correspon a un color. Exemples: 0 és vermell, 60 és groc i 120 és verd.

Saturació. Es representa com la distància a l'eix de brillantor negre-blanc. Els valors possibles van del 0 al 100%. A aquest paràmetre també se li sol dir “puresa” per l'analogia amb la puresa d'excitació i la puresa colorimètrica de la colorimetria. Quant menor sigui la saturació d'un color, major tonalitat grisosa hi haurà i més descolorit estarà. Per això és útil definir la instauració com la inversa qualitativa de la saturació.

Valor del color, la brillantor del color. Representa l'alçada a l'eix blanc-negre. Els valors possibles van del 0 al 100%. 0 sempre es negre. Depenent de la saturació, 100 podria ser blanc o un color més o menys saturat.

El model HSV va ser creat al 1978 per Alvy Ray Smith. Es tracta d'una transformació no lineal de l'espai de color RGB, i es pot utilitzar en progressions de color. HSV és el mateix que HSB però no que HSL o HSI.

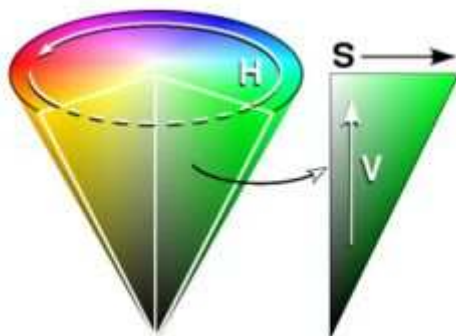


Figura 4.2. Representació del color HSV

És comú, que desitgem escollir un color adequat per alguna de les nostres aplicacions, quan es així resulta molt útil utilitzar la ruleta de color HSV. En ella la tonalitat es representa per una regió circular; una regió triangular separada, pot ser utilitzada per representar la saturació i el valor del color. Normalment, l'eix vertical del triangle denota la saturació, mentre que l'eix horitzontal correspon al valor del color. D'aquesta manera, un color pot ser escollit al prendre primer la tonalitat d'una regió circular, i després seleccionar la saturació i el valor del color desitjats de la regió triangular.

4.1.3. Transformació entre Espais de Color

Transformació RGB a HSV

Sigui MAX el valor màxim de les components (R, G, B), i MIN el valor mínim d'aquests mateixos valors, les components de l'espai HSV es poden calcular com:

$$H = \begin{cases} \text{no definit,} & \text{si } MAX = MIN \\ 60^\circ \times \frac{G - B}{MAX - MIN} + 0^\circ, & \text{si } MAX = R \text{ i } G \geq B \\ 60^\circ \times \frac{G - B}{MAX - MIN} + 360^\circ, & \text{si } MAX = R \text{ i } G < B \\ 60^\circ \times \frac{B - R}{MAX - MIN} + 120^\circ, & \text{si } MAX = G \\ 60^\circ \times \frac{R - G}{MAX - MIN} + 240^\circ, & \text{si } MAX = B \end{cases}$$

$$S = \begin{cases} 0, & \text{si } MAX = 0 \\ 1 - \frac{MIN}{MAX}, & \text{altrament} \end{cases}$$

$$V = MAX$$

Transformació HSV a RGB

$$H_i = \left[\frac{H}{60} \right] \bmod 6, \quad f = \frac{H}{60} - H_i, \quad p = V(1 - S), \quad 1 = V(1 - fS), \quad t = V(1 - (1 - f)S)$$

$$\text{si } H_i = \left\{ \begin{array}{l}
0, \quad R = V \\
\quad G = t \\
\quad B = p \\
1, \quad R = q \\
\quad G = V \\
\quad B = p \\
2, \quad R = p \\
\quad G = V \\
\quad B = t \\
3, \quad R = p \\
\quad G = q \\
\quad B = V \\
4, \quad R = t \\
\quad G = p \\
\quad B = V \\
5, \quad R = V \\
\quad G = p \\
\quad B = q
\end{array} \right.$$

4.2. Programació en Matlab

El script que s'ha creat en Matlab per tal de localitzar la posició de 3 objectes utilitzant la segmentació per color es resumeix en 4 punts:

1. Seleccionar els colors que tindran els objectes que es volen localitzar
2. Transformar aquestes seleccions en HSV i calcular els rangs de valors dels colors.
3. Crear una LUT amb els valors que s'han trobat anteriorment.
4. Determinar a partir de la LUT on es troben aquests objectes.

En el primer punt l'usuari veurà la imatge que està filmant la càmera i amb el ratolí haurà de seleccionar 3 vegades (una per cada objecte) l'extrem superior esquerre i l'extrem inferior dret de l'objecte que vulgui trobar. Per cada objecte haurà de prémer "Enter" i quan s'hagin seleccionat tots 3 el Matlab seguirà amb l'execució. D'aquesta manera es retallarà un finestra de la mida que s'hagi seleccionat i es guardarà aquest tros d'imatge en una matriu.

En el segon punt es transformarà cada imatge en HSV per tal de poder segmentar millor el color perquè en RGB és més complicat de treballar-hi. En RGB et pots trobar amb 2 colors pràcticament iguals i amb unes components molt diferents, en canvi amb HSV si dos colors són semblants tindran unes components semblants. Pel que fa la component H (to) s'agafa com a valors màxim i mínim el màxim i mínim que s'hagin trobat dins de la finestra. En el cas de la component S (saturació) també s'agafa com a valors màxim i

mínim el màxim i mínim que s'hagin trobat dins de la finestra. La última component no es té en compte.

En el tercer punt és quan hi entra en joc la funció en C i les Mex-Files. Aquí es crea la LUT amb els valors calculats anteriorment cridant una funció en C. Aquesta funció i les Mex-Files s'explicaran en el següent apartat.

En el darrer punt s'aniran agafant imatges de la càmera a 10 fps (s'ha considerat que és una velocitat suficient per poder treballar i sense que es col·lapsi l'ordinador) i a cada imatge s'anirà buscant el centre de les peces.

4.3. Programació en C

4.3.1. LUT

Les sigles LUT són una abreviació del mot anglès "LookUp Table" i consisteix en aquest cas en una matriu on s'hi introdueixen valors per fer més ràpida la cerca d'uns determinats resultats. Normalment s'utilitzen les LUT per augmentar la velocitat a canvi d'espai de memòria. La LUT que s'utilitzarà en aquesta funció té unes dimensions de 255*255*255 que, inicialment es podria pensar que és molt, però amb la quantitat de memòria que tenen els ordinadors d'avui dia, es pot utilitzar perfectament.

4.3.2. Mex-Files

Les Mex-Files són subrutines que es poden cridar des del Matlab però que, a diferència de les habituals subrutines o funcions del Matlab, aquestes estan programades en C.

El codi font d'una Mex-File té dues parts diferenciades:

- Una rutina operativa que conté el codi per realitzar les operacions que es vulguin implementar a la Mex-File. Les instruccions poden ser únicament numèriques o també utilitzant entrades i sortides.
- Una rutina d'enllaç que enllaça la rutina operativa amb el Matlab a través de l'entrada "mexFunction" i els seus paràmetres "prhs", "nrhs", "plhs" i "nlhs". On "prhs" és un vector dels paràmetres d'entrada, "plhs" és el nombre de paràmetres d'entrada, "nrhs" és un vector amb els paràmetres de sortida i "nlhs" és el nombre de paràmetres de sortida. La rutina d'enllaç crida a la rutina operativa com una subrutina.

A la rutina d'enllaç, es pot accedir a les dades a l'estructura mxArray i llavors manipular-les a la subrutina operativa en C. Per exemple, l'expressió `mxGetPr(prhs[0])` retorna un punter de tipus `double *` a les dades reals del mxArray que està apuntat per `prhs[0]`. Es pot utilitzar aquest punter com qualsevol altre punter de tipus `double *` al C. Després de cridar la rutina operativa des de l'enllaç, es pot posar un punter de tipus mxArray a les dades que retorna. Llavors el Matlab és capaç de reconèixer la sortida de la teva rutina operativa com la sortida d'una Mex-File.

El següent esquema (figura 4.3) ensenya com entren les dades a una Mex-File, quines funcions utilitza la rutina d'enllaç i com es retorna la sortida al Matlab.

Es crida una Mex-File anomenada func de la forma $[C, D] = \text{func}(A,B)$, d'aquesta manera s'informa al Matlab que s'envien les variables A i B a la Mex-File. De moment C i D queden sense assignar.

La rutina d'enllaç de func.c utilitza la funció mxCreate per crear els vectors de Matlab pels paràmetres de sortida. Posa els paràmetres plhs[0], [1], ... als punters dels vectors de Matlab que s'acaben de crear. Utilitza les funcions mcGet per extreure les dades de prhs[0], [1], ... I llavors crida la subrutina en C, passant els punters d'entrada i de sortida de dades com a paràmetres de la funció.

Quan retorna al Matlab, plhs[0] s'assigna a C i plhs[1] s'assigna a D.

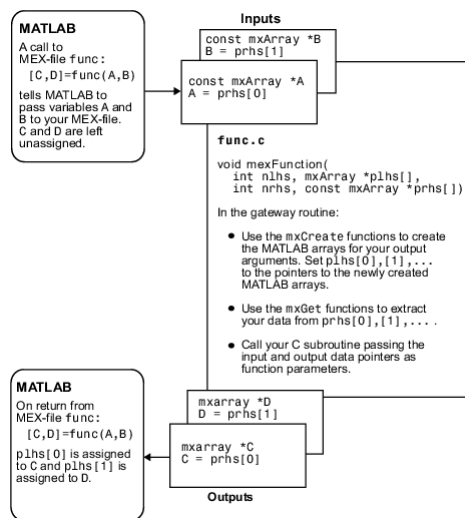


Figura 4.3. Exemple de crida d'una Mex-File des del Matlab

4.3.3. CalcLut

La funció programada en C s'ha creat per reduir de 30 minuts a pocs segons l'estona per crear una LUT. Aquesta funció consisteix en crear una LUT de $256*256*256$ on es trobaran totes les combinacions possibles RGB transformades ja a HSV.

Per fer la transformació de cada combinació RGB a HSV s'ha creat la funció "rgbahsv". És la mateixa funció que utilitza el Matlab que s'ha reduït a aquest cas concret i s'ha adaptat al C.

A mida que es va creant la LUT es va mirant quines combinacions pertanyen a una peça o una altra segons els rangs de valors del to i la saturació de cadascuna. D'aquesta manera cada peça queda marcada a la LUT amb un número concret que serà 0 si no concorda amb cap peça, 85 si és la peça 1, 127 si és la peça 2 i 255 si es tracta de la tercera peça.

4.4. Localitzar la Posició dels Objectes

En aquest apartat es pot comprovar l'aplicació del bloc de visió.

Sota la càmera s'hi posen 3 objectes, cadascun amb un color diferent (per tal que funcioni correctament hauria de ser d'un únic color i uniforme). Per tal d'iniciar l'aplicació s'ha d'escriure "visio" a la línia de comandos del Matlab.

Un cop iniciada l'aplicació es veuran els 3 objectes vistos des de la càmera en una finestra. S'haurà de seleccionar per cada objecte un punt de l'extrem de dalt esquerra i un altre de l'extrem de baix dret, tenint en compte que a partir d'aquests punts es crearà un rectangle i dins del rectangle només hi ha d'haver el color de l'objecte en sí, o sigui, que no s'ha de fer més gran que l'objecte. Per cada objecte s'ha de prémer "Enter" després de seleccionar els seus punts. A partir d'aquí el Matlab calcula els marges, crea la LUT i llavors l'aplica per segmentar la imatge per color.

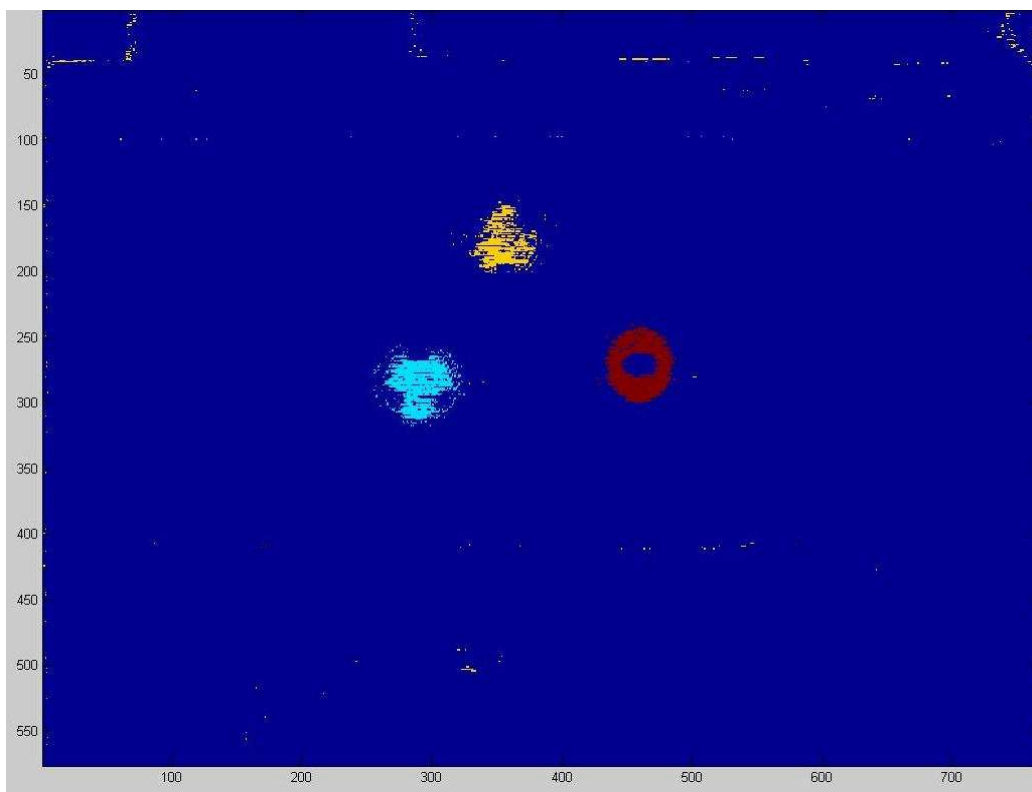


Figura 4.4. Segmentació amb soroll prop dels marges de la imatge

A la figura 4.4 es pot veure la segmentació de tres objectes en la qual hi ha molt de soroll.

Per tal de poder trobar el centre de cada objecte de manera més precisa s'ha reduït la zona de cerca retallant marges. Això es pot comprovar a la figura 4.5:

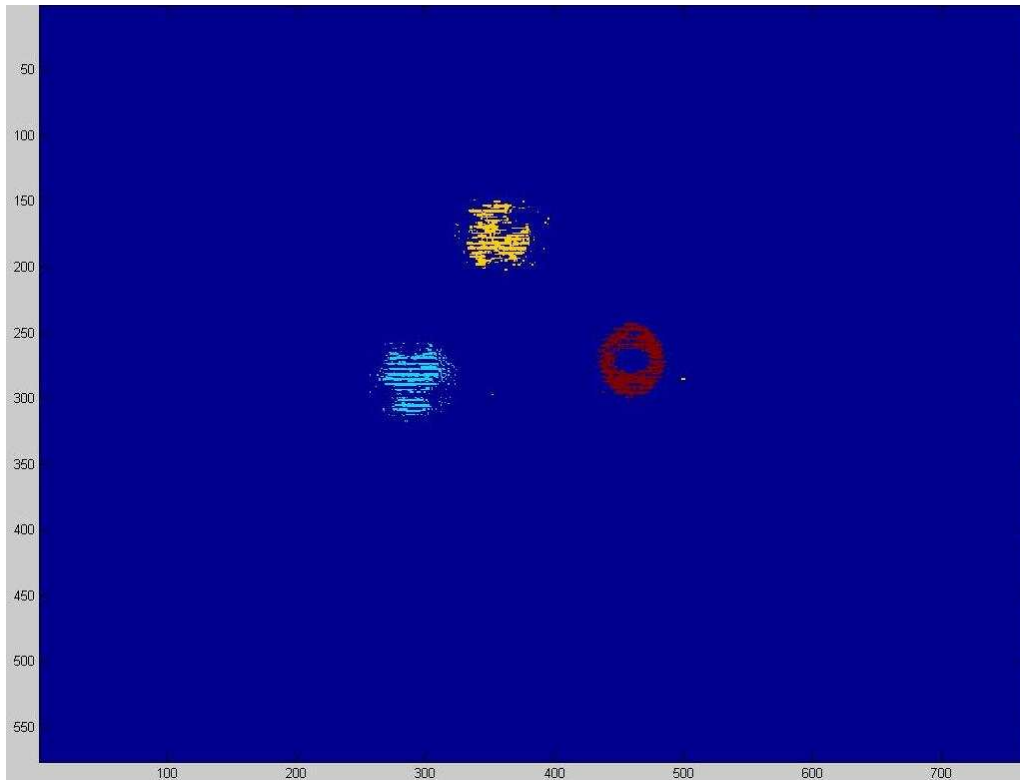


Figura 4.5. Segmentació sense soroll prop dels marges de la imatge

Ara ja es pot veure en una imatge que s'actualitza 10 vegades per segon on es troba cada objecte. A la línia de comandament del Matlab, també es va actualitzant 10 vegades per segon el centre de cadascun dels objectes.

5. DETECCIÓ I RECOL·LECCIÓ DE PECES

Ara que ja s'ha vist per separat com funcionen els dos blocs en els quals s'ha basat el projecte (comunicació i visió) toca ajuntar-los per tal de treure'n el màxim benefici. D'aquesta manera s'ha pogut crear l'aplicació que es presenta tot seguit.

S'ha agafat com a base l'aplicació de visió en la que es localitzava la posició de 3 objectes mitjançant la càmera i la segmentació per color. A partir d'aquí s'han afegit unes quantes instruccions del bloc de comunicació per fer moure el robot i s'ha aconseguit que tot sol agafi una peça i la deixi al seu lloc.

5.1. Preparació Prèvia

Abans de començar a programar l'aplicació ens trobàvem amb el problema que costava molt segmentar la pinça del robot degut a que era tota del mateix color platejat. S'havia de trobar un color diferent per cada costat de la pinça per tal de poder-ho segmentar millor. Havia de ser amb colors primaris per poder-los diferenciar millor. El vermell no podia ser perquè era el color de la peça que es volia agafar. Negre tampoc perquè era el color del terra on s'hi col·locava la peça. Blau tampoc perquè entre la pinça del robot hi passa un cable d'aquest color. El blanc tampoc podia servir perquè la càmera el confonia amb el color platejat de la resta de la pinça. Quedaven el groc i el verd, i així s'ha fet. S'ha folrat cada costat de la pinça amb un d'aquests colors tal com es veu a la figura 5.1. A partir d'aquí ja quedaven els colors ben diferenciats (vermell, verd i groc) i ja es podia començar a programar.



Figura 5.1. Detall de la pinça

5.2. Programació de l'Aplicació

A la figura 5.2 es pot veure un diagrama de les etapes per les quals passa el procés de recol·lecció d'una peça. Les etapes es van explicant tot seguit.

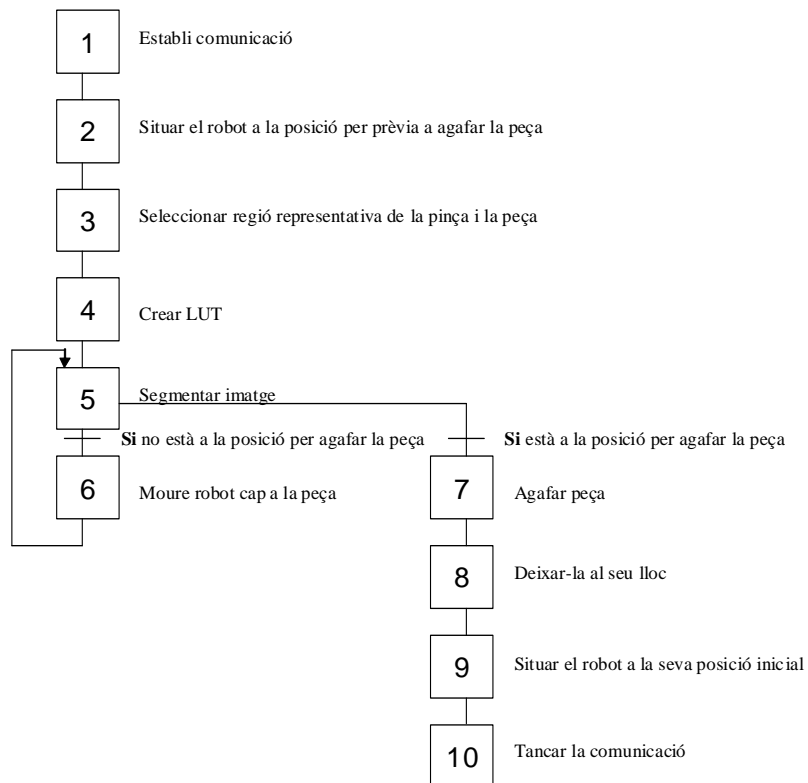


Figura 5.2. Diagrama d'etapes per les que passa el procés

Primer de tot s'ha d'establir comunicació amb el robot (etapa 1) i situar-lo en una posició on la càmera pugui veure alhora la seva pinça i la peça que ha d'agafar (etapa 2). Això es fa amb les instruccions següents:

```

t = connectar;
moureJoint(t,0,0,-100,0,0,0);
moureJoint(t,0,-30,-100,0,-50,0);
moureManual(t,2,0,0,-50,0,0,0);
  
```

Ara que ja està situat es segueix amb l'aplicació del bloc de visió amb la diferència que aquesta vegada no es seleccionaran 3 peces o objectes diferents sinó que es seleccionarà primer la part dreta de la pinça del robot (verda), llavors la part esquerra de la pinça del robot (grog) i finalment la peça que es vol que agafi (vermella) (etapa 3). A partir d'aquí el Matlab calcula els marges a partir dels quals crea la LUT (etapa 4) i la va aplicant a les imatges que rep de la càmera a 10 imatges per segon (etapa 5).

Les imatges que tracta el Matlab s'han retallat per tal d'eliminar una part del soroll que hi havia a les vores i feia que la detecció fos poc precisa (figura 5.3). Tot i això encara queda soroll però no impedeix que es pugui complir amb l'objectiu de recollir la peça.

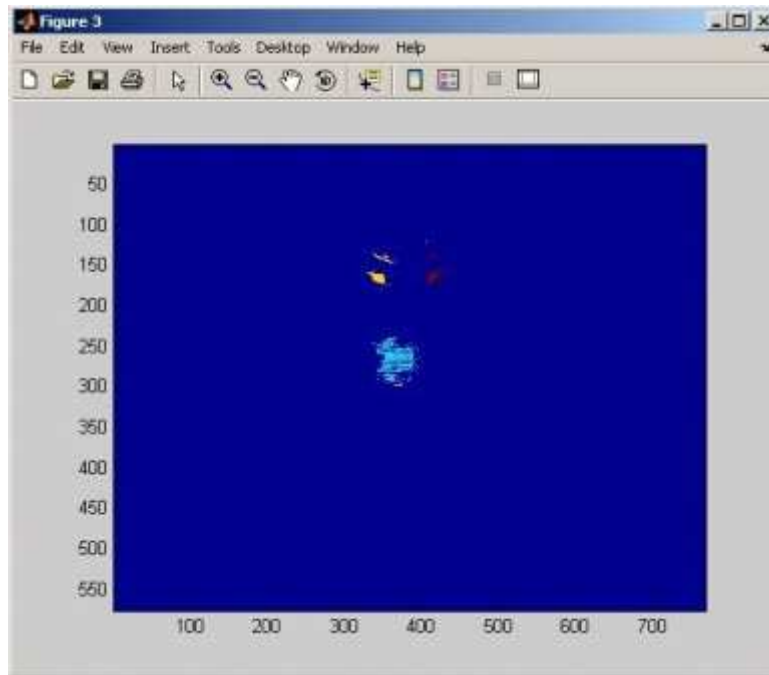


Figura 5.3. Segmentació de la pinça i la peça

El que ha de fer el robot és aconseguir que la peça quedi entre les dues parts de la pinça. Per aquest motiu, el Matlab va calculant contínuament el centre de les parts de la pinça i de la peça.

La metodologia que segueix és senzilla, si el centre de la peça (component y) està més a la dreta que la part de la dreta de la peça, el Matlab enviarà l'ordre al robot que es desplaci cap a la dreta (figura 5.4). En el cas que el centre de la peça (component y) estigui més a l'esquerra que la part esquerra de la pinça del robot, el Matlab i enviarà l'ordre que es desplaci cap a l'esquerra. Quan la peça es troba entre les dues parts de la pinça llavors el Matlab li envia l'ordre que vagi avançant cap a la peça fins que coincideixi el centre (component x) de la peça amb el de les dues parts de la pinça. Tot això es fa amb moviments molt petits per tal de no fer moviments bruscos, es fa amb moviments de 2 mm en el sentit que toca (etapa 6) . D'aquesta manera s'aconsegueix també més precisió. Ho fa amb les instruccions:

```

moureManual(t,2,0,2,0,0,0,0); % en el cas del moviment a l'esquerra
moureManual(t,2,0,-2,0,0,0,0); % en el cas del moviment a la dreta
moureManual(t,2,0,0,2,0,0,0); % en el cas del moviment endavant

```



Figura 5.4. El braç apropant-se a la peça que ha d'agafar

Un cop la pinça s'ha situat al lloc correcte, es tanca (etapa 7) i comença la maniobra per deixar-la al seu lloc (etapa 8). Això ho fa amb les instruccions:

```
tancarEina(t);  
moureJoint(t,0,-30,-100,0,-50,0);  
moureJoint(t,150,0,-100,0,-100,0);  
moureJoint(t,154,-75,-20,0,-100,0);
```

Quan la deixa al seu lloc (figura 5.5) obra la pinça amb la següent instrucció:

```
obrirEina(t);
```



Figura 5.5. El braç deixant la peça al seu lloc

Torna al seu lloc inicial (etapa 9) amb les següents instruccions:

```
moureJoint(t,154,0,0,0,0,0);  
moureJoint(t,0,0,0,0,0,0);
```

I es desconnecta de l'interpret de comandes (etapa 10) amb la següent instrucció:

```
desconnectar(t);
```

6. CONCLUSIONS

6.1. Resum

En aquest projecte he descobert la quantitat d'eines que pot arribar a proporcionar el Matlab per treballar. Gràcies a això s'ha aconseguit que amb un sol programa es pugui fer moure un robot alhora que s'adquireixen i es processen imatges d'una càmera connectada a través de cable de vídeo compost.

Per tal de dur a terme aquest projecte he hagut d'aprendre el funcionament del robot, del llenguatge de programació VAL3 i de les eines que proporciona el Matlab per poder adquirir i processar imatges.

He creat un conjunt de funcions en Matlab, un intèrpret de comandes en VAL3 i també un protocol de comunicació per tal de poder establir la comunicació entre el Matlab i l'intèrpret de comandes que s'executarà a l'armari de control del robot.

A part d'això he dissenyat una interfície gràfica des del Matlab que permetrà comunicar-se amb el robot d'una manera més senzilla i intuïtiva que des de la línia de comandes.

En una aplicació he utilitzat la segmentació per color per tal de saber en temps real quin és el centre de tres peces situades sota la càmera. Per tal de fer que l'execució pogués ser en temps real s'ha utilitzat una LUT. Aquesta LUT s'ha programat en C per tal que la seva creació sigui més eficient.

Per tal de comprovar el correcte funcionament de les eines creades anteriorment s'ha dissenyat i implementat una aplicació que consisteix en que el robot ha d'aconseguir agafar una peça utilitzant la segmentació per color i deixar-la al seu lloc. Tot això des del propi Matlab i en temps real.

Aquest ha estat un projecte molt complet en el qual he treballat en diversos camps de l'àmbit informàtic relacionant-los entre ells. S'ha treballat en:

- Robòtica perquè s'ha fet anar un manipulador industrial.
- Visió per computador perquè s'ha treballat amb una càmera de vídeo i s'han aplicat algorismes de segmentació per color.
- Comunicacions perquè per tal de poder enviar dades del Matlab a l'armari de control s'ha fet a través d'Ethernet i creant un protocol de comunicació.
- Llenguatges de programació perquè per tal que tot això funcionés s'ha hagut d'escriure codi i, a més, s'ha programat amb 3 llenguatges diferents els quals s'han anat comunicant entre ells:
 - o VAL3
 - o Matlab
 - o C

6.2. Assoliment dels Objectius

Tot seguit es veurà com s'han anat assolint els objectius que s'han plantejat al inici del projecte.

Primer objectiu: realitzar un intèrpret de comandes en VAL3 que rebi les ordres a través d'una connexió TCP/IP.

S'ha dissenyat i implementat un intèrpret de comandes en VAL3 amb un conjunt de funcions per tal de rebre ordres a través de sockets Ethernet. La principal dificultat ha estat familiaritzar-me amb el llenguatge de programació VAL3, sobretot amb la comunicació a través de sockets Ethernet.

Segon objectiu: realitzar una toolbox de Matlab per enviar diferents ordres mitjançant una connexió TCP/IP.

S'han creat un conjunt de funcions en Matlab equivalents a instruccions del llenguatge VAL3 per tal de poder comandar el robot des del propi Matlab. S'ha dissenyat un protocol de comunicació que permet enviar les ordres del Matlab a l'armari de control del robot mitjançant sockets Ethernet. El llenguatge del Matlab ja el coneixia però he hagut d'aprofundir-hi molt més per tal de dur a terme aquest projecte.

Tercer objectiu: adquirir i processar mitjançant Matlab imatges de la càmera en temps real i detectar la posició d'objectes artificials mitjançant la segmentació per color.

S'han estudiat els "toolbox" de Matlab que permeten adquirir i processar imatges. S'ha dissenyat i implementat una aplicació en Matlab en la qual s'adquireixen imatges de la càmera i es processen utilitzant la segmentació per color amb la finalitat de detectar la posició d'objectes en temps real.

Quart objectiu: dissenyar i realitzar una aplicació amb Matlab que reculli peces detectades amb la càmera.

S'ha dissenyat i implementat una aplicació en Matlab que consisteix en que el robot pot recollir una peça que detecta la càmera utilitzant la segmentació per color i deixar-la al seu lloc corresponent. Tot aquest procés es fa en temps real, des del moment en què la càmera detecta la posició de la peça fins que el robot la deixa al seu lloc.

D'aquesta manera es pot dir que s'han assolit la totalitat dels objectius proposats de manera satisfactòria.

6.3. Treball Futur

Aquest ha estat un primer pas per moure el robot a través del Matlab però sense implementar totes les funcions que permet el VAL3. Un dels treballs futurs podria ser **ampliar les instruccions traduïdes del VAL3 al Matlab**.

S'ha creat una interfície gràfica per comunicar-se amb el robot de manera més senzilla i intuïtiva que per la línia de comandes. Es podrien **ampliar les funcions de la interfície** de tal manera que no fes falta programar scripts des de la línia de comandes del Matlab i es poguessin programar directament des de la interfície podent fer servir les posicions a les quals s'hi hagi accedit utilitzant el moviment manual.

Pel que fa a la part de visió també es podria millorar, per exemple, **utilitzant tècniques de lògica difusa** a l'hora de recollir les peces per tal que el robot es desplaci ràpidament quan està lluny de la peça i a mida que es vagi apropant vagi reduint la seva velocitat.

Per tal de detectar les peces s'ha utilitzat la tècnica de la segmentació per color. Com a treball futur es podrien **provar altres tècniques de visió per computador** i fer-ne estudis per saber quina dóna més bons resultats tant a nivell de velocitat com a nivell de precisió.

7. BIBLIOGRAFIA

Per tal de dur a terme aquest projecte s'han consultat les fonts següents:

Manuais Stäubli:

- Armario de Control CS8C
- Brazo – Familia TX Serie 60
- Manual de Referencia VAL3

Manual Matlab: www.mathworks.com

Wikipedia: www.wikipedia.es

ANNEX 1. MANIPULADOR STÄUBLI TX60 I ARMARI DE CONTROL CS8C

A1.1. Robot

El braç manipulador Stäubli TX60 consta d'una cadena cinemàtica de sis sòlids rígids o elements units entre sí mitjançant sis articulacions de revolució (q_1, \dots, q_6) que ens aporten sis graus de llibertat. Tal com es pot veure a la figura A1.1 els diferents sòlids rígids del braç són: la base (A), l'espatlla (B), el braç (C), el colze (D), l'avantbraç (E) i el puny (F).

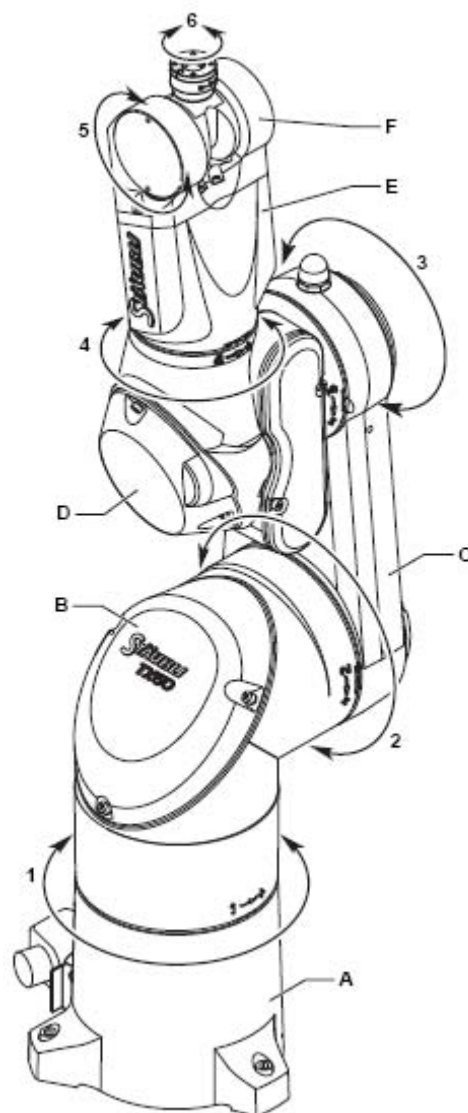


Figura A1.1. Esquema del manipulador on s'indiquen els seus diferents sòlids rígids

A1.1.1. Espai de Treball

L'espai o volum de treball (figura A1.2) és el conjunt de punts accessibles per l'extrem del robot. Aquest volum queda determinat per la grandària, forma i tipus dels elements que l'integren, juntament amb les limitacions imposades pel sistema de control. El fet que un punt de l'espai sigui accessible pel robot no implica que ho pugui fer en qualsevol orientació. Existeix un conjunt de punts que solament podem accedir en una determinada orientació, mentre que altres punts admetran qualsevol orientació. En el braç Stäubli TX60 l'espai o volum de treball queda definit amb els següents paràmetres:

R.M (Radi de treball màxim entre eixos 1 i 5) 600 mm

R.m1 (Radi de treball mínim entre eixos 1 i 5) 190 mm

R.m2 (Radi de treball mínim entre eixos 2 i 5) 189 mm

R.b (Radi de treball entre eixos 3 i 5) 310 mm

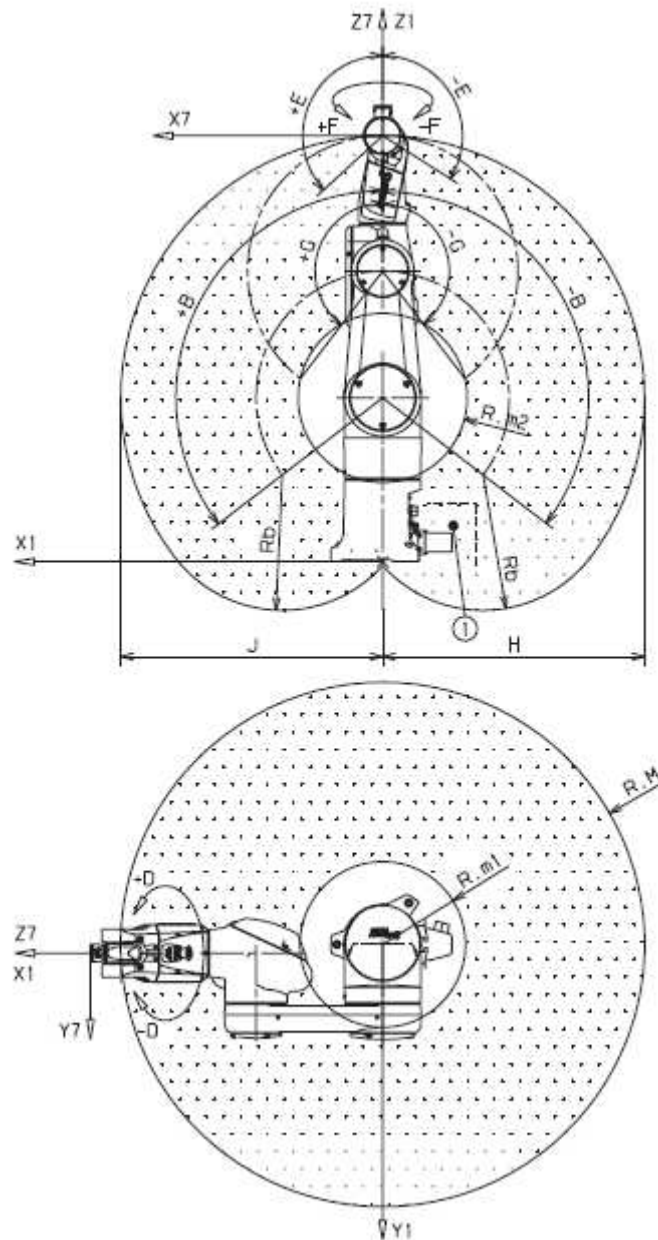


Figura A1.2. Espai de treball del braç Stäubli TX60

A1.1.2. Càrrega Nominal Transportable i Càrrega Màxima

La càrrega nominal és la càrrega que el braç pot transportar tot garantint el correcte funcionament del robot i sense limitar les prestacions del braç en velocitat i acceleració. La càrrega nominal transportable per al Stäubli TX60 és de 3.5 Kg. Per altra banda la càrrega màxima pot arribar a ser de 9 kg només sota condicions especials.

A1.1.3. Velocitat Màxima

La velocitat a la que es pot moure l'extrem del braç d'un robot depèn fortament de la càrrega transportada. Estan inversament relacionats. Per tant, la màxima velocitat del robot serà amb càrrega nul·la. La velocitat màxima per al Stäubli TX60 en el centre de gravetat de la càrrega és de 8 m/s.

A1.1.4. Dimensions

Les principals dimensions dels sis elements que integren el braç manipulador són les que es poden veure a la figura A1.3:

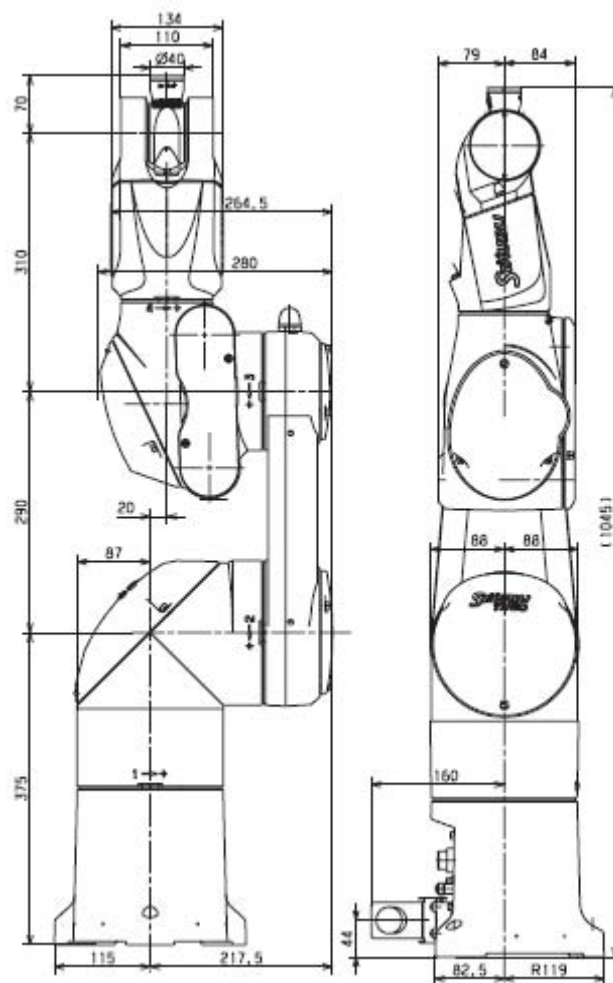


Figura A1.3. Mides dels elements del braç manipulador

A1.1.5. Repetibilitat

La repetibilitat és de ± 0.02 mm.

A1.1.6. Amplitud, Velocitat i Resolució

El braç Stäubli TX60 consta de sis articulacions i sis eixos de rotació els quals tenen els següents paràmetres:

EIX	1	2	3	4	5	6
Amplitud (°)	360	255	285	540	255	540
Distribució d'amplitud (°)	A ± 180	B ± 127.5	C ± 142.5	D ± 270	E + 133.5 - 122.5	F ± 270
Velocitat nominal (°/s)	287	287	431	410	320	700
Velocitat màxima (°/s)	373	373	500	968	800	1125
Resolució angular (°. 10^{-3})	0.057	0.057	0.057	0.114	0.122	0.172

A1.1.7. Ambient de Treball

Temperatura de funcionament: +5°C a +40°C
Humitat de 30% a 95% màx. sense condensació
Altitud 2000 m màx.

A1.1.8. Pes

El pes total del robot és de 51.4 kg.

A1.2. Armari de Control

A1.2.1. Localització de les Entrades – Sortides

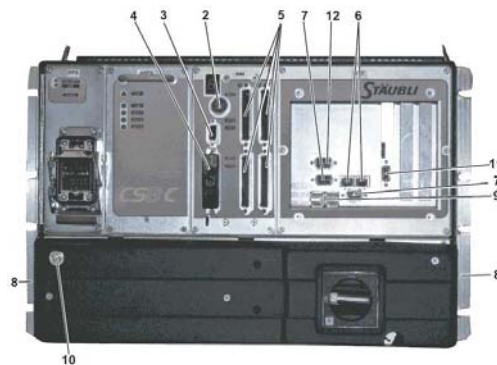


Figura 2.6. Entrades – Sortides de l'armari de control CS8C

Les entrades – sortides de l’armari de control són les següents:

- Connector per la MCP (2).
- Entrades – Sortides ràpides (3).
- Connexió amb la cèl·lula (aturada d’emergència, porta, ...) (4).
- Opcions E/S digitals (BIO) (5).
- Connexions Ethernet (6).
- Connexions sèrie (7).
- Connexions USB (9).
- Braçalet antiestàtic (10).
- Entrada codificador (11).
- Sortida CAN pels robots Scara (12).

A1.2.2. Enllaç Ethernet

Configuració

El CS8C disposa de 2 ports Ethernet J204 i J205. La direcció IP de cadascun es pot modificar des del panell de control. La modificació és efectiva immediatament. De fàbrica, el primer port està configurat amb la direcció 192.168.0.254 (màscara 255.255.255.0), el segon amb la direcció 172.31.0.1 (màscara 255.255.0.0).

També es pot obtenir automàticament una direcció IP a partir de la xarxa (mitjançant el protocol DHCP).

El controlador CS8C pot accedir a altres subxarxes Ethernet, mitjançant passarel·les configurables a partir del tauler d’instruments.

Cada passarel·la està definida per:

- La direcció IP del perifèric utilitzat com a passarel·la. Aquesta direcció ha de pertànyer a la mateixa subxarxa que el controlador CS8C.
- La direcció IP de la subxarxa que s’ha d’abastar. Es pot utilitzar una direcció nul·la “0.0.0.0” per definir una passarel·la per defecte, que permeti accedir a totes les subxarxes que no siguin administrades per una passarel·la específica.

Protocol FTP

L’armari de control CS8C és un servidor FTP que permet intercanviar fitxers per Ethernet. N’hi ha prou amb definir la direcció IP de l’armari de control del CS8C per poder accedir-hi a través de FTP, i utilitzar un login i contrasenya de xarxa corresponent a un perfil d’usuari definit al CS8C. El dret d’accés de lectura i escriptura depèn del perfil d’usuari seleccionat.

Ethernet Sockets (TCP)

L’armari de control CS8C pot ser configurat per comunicar-se a través d’Ethernet mitjançant sockets. L’armari de control CS8C suporta fins a 40 sockets simultanis en mode

client i/o en mode servidor. La configuració dels sockets Ethernet es fa des de l'aplicació "Panel de Control" (Control Panel → I/O → Socket). No es suporten els sockets UDP.

Els paràmetres d'un socket servidor són:

- El port de connexió entre 0 i 65535.
- El nombre màxim de clients simultanis.
- El temps previ a l'activació d'error (temps màxim d'espera en lectura o en connexió). Un valor nul elimina el control del temps d'espera.
- El caràcter de final de cadena.

Els paràmetres d'un socket client són els mateixos amb, a més a més, la direcció IP del socket servidor. Un menú "Prueba" permet provar la connexió amb el servidor.

Un socket servidor s'activa ("obert") en el CS8C quan l'utilitza un programa VAL3, i es desactiva ("tancat") quan es desconnecta l'últim client. Quan s'arriba a la quantitat màxima de clients per un socket servidor, els altres clients que s'intenten connectar són acceptats però la comunicació és interrompuda immediatament pel servidor.

A1.2.3. Port Sèrie

Hi ha dues connexions sèrie disponibles a l'armari de control CS8C (J203, COM1 i J201, COM2) per intercanviar dades entre una aplicació VAL3 i un equip de la cèl·lula.

La configuració de les connexions sèrie s'efectua des de la visualització de les entrades – sortides del "Panel de Control".

Els paràmetres configurables de l'enllaç sèrie són:

- La velocitat de transmissió (de 110 a 115200 bauds).
- La quantitat de bits de dades (de 5 a 8).
- La quantitat de bits de stop (de 1 a 2).
- La paritat (parell, senar o sense paritat).
- El temps previ a l'activació d'error (temps màxim d'espera en lectura). Un valor nul elimina el control de temps d'espera.
- El caràcter de final de cadena.
- Per J201 (COM2), la configuració RS232/RS422.

A1.2.4. Calculador

El calculador executa el programa del qual se'n deriven l'accionament del braç, la gestió d'entrades – sortides, etc.

Els seus components principals són:

- Una targeta CPU sobre la qual hi ha muntat un Flash Disk per l'emmagatzemament del sistema VAL3 i dels programes de l'aplicació. Aquesta targeta està equipada en el panell frontal de les connexions Ethernet, USB i Sèrie.

- Els ports USB “User” (J202 i J209) es poden utilitzar per claus que suportin els següents protocols:
 - o #1: Reduce Block Commands (RBC) T10 Project 1240-D.
 - o #4: UFI.
 - o #6: SCSI transparent command set.
- Una targeta STARC que assegura la interconnexió amb les transmissions en el CS8C i amb les targetes DSI al peu del braç mitjançant una comunicació numèrica. Es disposa igualment d’una entrada codificador (J305).
- Emplaçaments lliures per opcions de format PCI subministrades per Stäubli.

A1.2.5. Alimentació RPS 235

L’alimentació RPS235 genera la tensió contínua necessària pels amplificadors a partir d’una tensió alterna trifàsica. Quan la velocitat dels motors ha de disminuir, l’alimentació és capaç d’absorbir la potència sobrant dels motors en una resistència de regeneració.

L’alimentació també genera un senyal de control per la targeta RSI: Estat de l’alimentació (PWR-OK).

A1.2.6. Alimentació ARPS

L’alimentació ARPS genera les tensions contínues per la part lògica de l’armari de control:

- Una tensió de 24 VDC pel calculador, la targeta RSI i els amplificadors,
- Una tensió de 24 VDC per la ventilació de l’armari del robot,
- Una tensió de 24 VDC pels frens. Aquesta tensió està present quan la targeta RSI subministra un senyal de comandament (BRK_REL_EN).
- Una tensió de 13 VDC per les targetes DSI situades al peu del braç.

L’alimentació ARPS també genera senyals de control per la targeta RSI: Estat de la alimentació (senyal “ALIM_OK”), presència de la tensió de xarxa (senyal “SEC-TEUR_OK”).

A1.2.7. Amplificadors

Cada amplificador està dedicat a 2 eixos i depèn del tipus de motors accionats i de les característiques desitjades. Així, és possible que dos amplificadors siguin mecànicament idèntics però no intercanviables.

La personalització d’una transmissió per un determinat eix està indicat per la referència escrita a la transmissió. Aquesta està alhora composta per una configuració de hardware i una configuració de software.

Els amplificadors estan alimentats per l’alimentació RPS325 per la part de potència i per l’alimentació ARPS per la part lògica.

Els amplificadors s'accionen a partir de la targeta STARC.

A1.2.8. Mòdul de Potència (PSM)

Aquest conjunt, constituït per un transformador i proteccions associades està situat al fons de l'armari de control.

Segons el tipus d'armari de control, la connexió s'efectua en monofàsic o en trifàsic, amb o sense transformador. Els fusibles de l'entrada tenen unes dimensions de 10,3 x 38 mm / 500 V.

En el cas d'un transformador trifàsic multitensió, l'elecció de la tensió es fa a la regleta del transformador.

Les sortides 230VAC del transformador estan protegides per disjuntors monofàsics i trifàsics.

El disjuntor trifàsic alimenta la part de potència de l'armari de comandament a través dels contactors i de l'alimentació RPS325. El disjuntor monofàsic alimenta les parts lògiques de l'armari de comandament a través de l'alimentació ARPS.

A1.2.9. Targeta RSI

La targeta RSI reuneix tots els elements per assegurar, en bones condicions de seguretat elèctrica, la posada en tensió del braç o l'aturada d'emergència del mateix. Està equipada amb relés de seguretat redundants. Aquesta pot tenir fins a 32 entrades i 32 sortides digitals (opcions "BIO").

La targeta RSI està enllaçada:

- a la seva alimentació 24V (ARPS) per mitjà de J104. Aquest enllaç també permet governar el comandament de l'alimentació dels frens subministrada per ARPS,
- a la targeta STARC per mitjà de J100 per la gestió de la seguretat i de les Entrades/Sortides,
- als amplificadors a J112 amb la finalitat de proporcionar la seva tensió de 24V i recuperar el seu senyal d'error,
- al comandament dels contactors per potència a través de J105,
- al braç a J101 (electrovàlvules, Limit Switch, comandament manual dels frens).
Les sortides d'electrovàlvules estan protegides per microfusibles de 0,5 A.

A la cara davantera d'aquesta targeta es troben disponibles:

- La connexió amb la MCP a J110,
- La connexió amb la cèl·lula per la connexió dels senyals d'aturada d'emergència a J109,
- Entrades/Sortides ràpides a J111.

La targeta RSI assegura la continuïtat dels circuits d'aturada d'emergència, enllaçant els diferents contactes d'aturada d'emergència l'estat dels quals està senyalitzat pel visor.

Hi ha dues cadenes d'aturada d'emergència redundants alimentades “en oposició” (una cadena alimentada entre +24V i 0V i l'altra entre 0V i +24V), per estar en condicions de detectar un curtcircuit entre aquestes 2 cadenes: si es produeix un curtcircuit, el crema el fusible F2 (microfusible 250 mA) i s'activa un procediment d'aturada d'emergència.

Un muntatge de relés redundants assegura la coherència dels senyals d'aturada d'emergència, controlada també per la targeta CPU. Quan s'acciona una aturada d'emergència, si tan sols s'obre un dels seus contactes, aquest defecte és detectat per la targeta CPU. En aquest cas, s'ha de trobar l'origen del defecte i suprimir-se el defecte. Hi ha 3 tipus d'ajudes per trobar l'origen del defecte:

- Els missatges mostrats a la MCP,
- El visor de la targeta RSI,
- Les Entrades / Sortides mostrades al panell de control.

Quan es sol·licita la posada en tensió del braç, si es queda obert algun o tots dos contactes d'una aturada d'emergència, no serà possible efectuar el procediment de posada en tensió del braç.

ANNEX 2. LENGUATGE DE PROGRAMACIÓ VAL3

A2.1. Introducció

El llenguatge VAL3 és un llenguatge de programació d'alt nivell creat pel comandament dels robots Stäubli a l'àmbit d'aplicacions industrials de manipulació i d'assemblatge.

El llenguatge VAL3 ha conservat els aspectes fonamentals de les possibilitats d'un llenguatge informàtic en temps real estàndard, integrant-se les funcionalitats específiques pel comandament d'un robot en una cèl·lula industrial:

- Eines de control del robot
- Eines de modelització geomètrica
- Eines de control d'entrades i sortides

A2.2. Elements del Llenguatge VAL3

Els elements constitutius del llenguatge VAL3 són:

- Les aplicacions
- Els programes
- Les biblioteques
- Els tipus de dades
- Les constants
- Les variables (globals, constants, paràmetres)
- Les tasques

A2.2.1. Aplicacions

Definició

Una aplicació VAL3 és un software autònom de programació del robot i de les entrades-sortides vinculades a un CS8.

Una aplicació VAL3 està constituïda pels elements següents:

- un conjunt de programes: les instruccions VAL3 que s'han d'executar
- un conjunt de variables globals: les dades de l'aplicació
- un conjunt de biblioteques: les instruccions i les dades externes utilitzades per l'aplicació

Quan una aplicació s'està executant, també conté:

- un conjunt de tasques: els programes en curs d'execució

Contingut Predeterminat

Una aplicació VAL3 sempre conté els programes **start()** i **stop()**, un sistema de referència (tipus *frame*) “world” i una eina (tipus *tool*) “flange”. En el moment de crear-se també conté les instruccions i les dades del model escollit.

Posada en Marxa i Aturada

Les instruccions VAL3 no permeten manipular aplicacions: La càrrega, descàrrega, posada en marxa i aturada de les aplicacions es fa únicament a través de la interfície d’usuari del CS8.

Quan es posa en marxa una aplicació VAL3, s’executa el seu programa **start()**.

Una aplicació VAL3 s’atura tota sola quan acaba les seves tasques: llavors, s’executa el programa **stop()**. Totes les tasques creades per biblioteques, si queden, són destruïdes en l’ordre invers de la seva creació.

Si l’aturada d’una aplicació VAL3 es provoca des de la interfície d’usuari del CS8, la tasca d’inici, si encara existeix, es destrueix immediatament. S’executa després del programa **stop()** i, tot seguit, totes les tasques de l’aplicació, si queden per fer, es destrueixen en l’ordre invers de l’ordre en que van ser creades.

Paràmetres d’Aplicació

Una aplicació VAL3 pot ser configurada pels següents paràmetres:

- la unitat de longitud
- la mida de memòria d’execució

Aquests paràmetres no són accessibles per una instrucció VAL3, i només poden ser modificats a través de la interfície d’usuari del CS8.

Unitats de Longitud

La unitat de longitud d’una aplicació VAL3 pot ser el mil·límetre o la polsada. Aquesta és utilitzada pels tipus de dades geomètrics del VAL3: sistema de referència, punt, transformació, eina, allisat de trajectòria.

La unitat de longitud d’una aplicació es defineix en el moment de la seva creació per la unitat de longitud en curs del sistema i ja no es pot modificar.

Mida de Memòria d’Execució

La mida de la memòria d’execució d’una aplicació VAL3 és la mida de memòria disponible per cadascuna de les tasques per emmagatzemar allà, entre altres coses, les variables locals dels programes. La seva mida predeterminada és de 5000 bytes.

Aquest valor pot ser insuficient per aplicacions que contenen vectors de grans dimensions a variables locals, o algorismes recursius. En aquest cas s'ha d'augmentar des de la interfície d'usuari del CS8.

A2.2.2. Programes

Definició

Un programa és una seqüència d'instruccions VAL3 per executar.

Un programa està constituït pels següents elements:

- una seqüència d'instruccions: les instruccions VAL3 que s'han d'executar
- un conjunt de variables locals: les dades internes del programa
- un conjunt de paràmetres: les dades facilitades al programa quan es crida

Els programes permeten agrupar seqüències d'instruccions susceptibles de ser utilitzades en diversos llocs en una aplicació. A més de l'estalvi de programació que aporten, permeten fer que es destaquï l'estructura de les aplicacions, facilitar la seva programació i manteniment i millorar la seva lectura.

El nombre d'instruccions d'un programa està únicament limitat per l'espai disponible en la memòria del sistema.

El nombre de variables locals i de paràmetres està únicament limitat per l'espai disponible a la memòria d'execució del programa.

Reentrada

Els programes són reentrants, que vol dir que un programa es pot cridar a ell mateix de manera recursiva (instrucció **call**), o ser cridat simultàniament per diverses tasques. Cada cop que es crida el programa té variables locals i paràmetres propis.

Programa **Start()**

El programa **start()** és el programa que es crida quan es posa en marxa una aplicació VAL3. No pot tenir paràmetres.

En aquest programa es trobaran especialment totes les operacions necessàries per la posada en marxa de l'aplicació: inicialització de les variables globals, de les entrades-sortides, posada en marxa de les tasques de l'aplicació, etc.

L'aplicació no s'acaba necessàriament al final del programa **start()**, si altres tasques de l'aplicació estan encara en execució.

És possible cridar el programa **start()** dins d'un programa (instrucció **call**) com qualsevol altre programa.

Programa Stop()

El programa **stop()** és el programa que es crida quan s'atura una aplicació VAL3. No pot tenir paràmetres.

Dins d'aquest programa es trobaran especialment totes les operacions necessàries per acabar pròpiament l'aplicació: reinicialització de les entrades-sortides, aturada de les tasques de l'aplicació segons una seqüència apropiada, etc.

És possible cridar el programa **stop()** dins d'un programa (instrucció **call**) com qualsevol altre programa: la crida del programa **stop()** no activa l'aturada de l'aplicació.

A2.2.3. Les Biblioteques

Definició

Una biblioteca VAL3 és un software reutilitzable per una aplicació o altres biblioteques VAL3.

Com una aplicació, una biblioteca VAL3 està constituïda pels elements següents:

- un conjunt programes: les instruccions VAL3 que s'han d'executar
- un conjunt de variables globals: les dades de la biblioteca
- un conjunt de biblioteques: les instruccions i dades externes utilitzades per la biblioteca

Quan la biblioteca s'està executant, també pot contenir:

- un conjunt de tasques: els programes propis de la biblioteca que s'està executant

El format de salvaguarda d'una biblioteca és el mateix que el d'una aplicació VAL3. Qualsevol aplicació es pot utilitzar com una biblioteca, i qualsevol biblioteca es pot utilitzar com una aplicació, si els programes **start()** i **stop()** hi estan definits.

Càrrega i Descàrrega

Quan una aplicació VAL3 està oberta, totes les biblioteques declarades són analitzades per construir les interfícies corresponents. Aquesta etapa no carrega les biblioteques a la memòria.

Quan es carrega una biblioteca, s'inicialitzen les seves variables globals i es comproven els seus programes per detectar eventuais errors de sintaxis.

No és necessària la descàrrega d'una biblioteca; aquesta es fa automàticament quan s'acaba l'aplicació o quan es carrega una nova biblioteca enlloc d'una altra.

Quan s'atura una aplicació VAL3 des de la interfície d'usuari del CS8, primer s'executa el programa **stop()** i, tot seguit, es destrueixen totes les tasques de l'aplicació i de les seves biblioteques, si en queden.

A2.2.4. Tipus de Dades

Definició

El tipus d'una constant o d'una variable VAL3 és una característica que permet que el sistema controli les instruccions i els programes a la seva disposició.

Totes les constants i variables VAL3 tenen un tipus. Això permet un control inicial pel sistema en el moment d'editar un programa i, per tant, la detecció immediata de certs errors de programació.

El llenguatge VAL3 suporta els següents tipus senzills:

- el tipus *bool*: pels valors booleans (cert / fals)
- el tipus *num*: pels valors numèrics
- el tipus *string*: per les cadenes de caràcters
- el tipus *dio*: per les entrades-sortides tot o res
- el tipus *aio*: per les entrades-sortides numèriques (analògiques o digitals)
- el tipus *sio*: per les entrades-sortides en connexió sèrie i socket Ethernet

Un tipus estructurat és un tipus que reuneix diverses dades separades en camps. Els camps de tipus estructurats són accessibles individualment pel seu nom.

El llenguatge VAL3 suporta els següents tipus estructurats:

- el tipus *trsf*: per les transformacions geomètriques cartesianes
- el tipus *frame*: pels plans geomètrics cartesianes
- el tipus *tool*: per les eines ajustades en un robot
- el tipus *point*: per les posicions cartesianes d'una eina
- el tipus *joint*: per les posicions articulars del robot
- el tipus *config*: per les configuracions del robot
- el tipus *mdesc*: pels paràmetres de desplaçament del robot

Tipus Senzills

num **size**(*<variable>*): retorna la mida de la variable.

Bool

Els valors possibles de les variables o constants de tipus *bool* són:

- “true”: valor cert
- “false”: valor fals

De manera predeterminada, una variable de tipus *bool* s'inicialitza en el valor “false”.

Num

El tipus *num* modelitza un valor numèric, amb aproximadament 14 xifres significatives.

Cada càlcul numèric es fa per tant amb una precisió limitada per aquestes 14 xifres significatives.

S'ha de tenir en compte quan es vol provar la igualtat de dos valors numèrics: la majoria de les vegades és necessari provar en un interval.

Les variables de tipus *num* es reinicialitzen de manera predeterminada a 0.

Instruccions:

num **sin**(<*num* angle>): retorna el sinus de l'angle.

num **asin**(<*num* valor>): retorna el sinus invers de valor, en graus. El resultat estarà comprès entre -90 i +90 graus. Si valor és superior a 1 o inferior a -1 es genera un error.

num **cos**(<*num* angle>): retorna el cosinus de l'angle.

num **acos**(<*num* valor>): retorna el cosinus invers de valor, en graus. el resultat estarà comprès entre 0 i 180 graus. Si valor és superior a 1 o inferior a -1 es genera un error.

num **tan**(<*num* angle>): retorna la tangent de l'angle.

num **atan**(<*num* valor>): retorna la tangent inversa de valor, en graus. El resultat estarà comprès entre -90 i +90 graus.

num **abs**(<*num* valor>): retorna el valor absolut de valor.

num **sqrt**(<*num* valor>): retorna l'arrel quadrada de valor. Es genera un error d'execució si valor és negatiu.

num **exp**(<*num* valor>): retorna l'exponencial de valor. Es genera un error d'execució si valor és massa gran.

num **ln**(<*num* valor>): retorna el logaritme neperià de valor. Es genera un error d'execució si valor és negatiu o nul.

num **log**(<*num* valor>): retorna el logaritme decimal de valor. Es genera un error d'execució si valor és negatiu o nul.

num **roundUp**(<*num* valor>): retorna valor arrodonit a l'enter immediatament superior.

num **roundDown**(<*num* valor>): retorna valor arrodonit a l'enter immediatament inferior.

num **round**(<*num* valor>): retorna valor arrodonit a l'enter més proper.

num **min**(<*num* x>, <*num* y>): retorna el valor mínim de x i y.

num **max**(<*num* x>, <*num* y>): retorna el valor màxim de x i y.

num **limit**(<*num* valor>, <*num* min.>, <*num* max.>): retorna valor limitat per min. i max..

num **sel**(<*bool* condició>, <*num* valor1>, <*num* valor2>): retorna valor1 si condició és “true”, en cas contrari retorna valor2.

String

Les variables de tipus cadena permeten emmagatzemar texts.

La longitud màxima d'una cadena de caràcters és de 128 caràcters.

Els caràcters suportats pel tipus *string* són els caràcters editables no accentuats (codi ASCII entre 32 i 126).

Les variables de tipus *string* s'inicialitzen de manera predeterminada amb el valor “” de longitud nul·la.

Instruccions:

string **toString**(<*string* format>, <*num* valor>): retorna una cadena de caràcters que representa valor segons el format de visualització format. El format és “mida.precisió”, on mida és la mida mínima del resultat (s'afegeixen espais a la mida de la cadena si fos necessari), i precisió és la quantitat de xifres significatives després de la coma (els 0 al final de la cadena són substituïts per espais). De manera predeterminada mida i precisió valen 0. La part entera del valor mai es trunca, encara que la seva longitud de visualització sigui més gran que mida.

string **toNum**(<*string* cadena>, <*num*& valor>, <*bool*& relació>): calcula el valor numèric representat al principi de la cadena especificada, i retorna una cadena en la qual tots els caràcters han estat suprimits fins a la representació següent d'un valor numèric. Si el principi de cadena no representa cap valor numèric, relació es posa “false” i no es modifica valor.

string **chr**(<*num* codi ASCII>): retorna la cadena constituïda pel valor del codi ASCII introduït, si és suportat pel tipus *string*. En cas contrari retorna una cadena buida.

num **asc**(<*string* cadena>, <*num* posició>): retorna el codi ASCII d'índex posició. Retorna -1 si posició és negatiu o superior a la mida de cadena.

string **left**(<*string* cadena>, <*num* mida>): retorna mida dels primers caràcters de cadena. Si mida és més gran que la longitud de cadena, la instrucció retorna cadena. Es genera un error d'execució si mida és negatiu.

string **right**(<*string* cadena>, <*num* mida>): retorna mida dels últims caràcters de cadena. Si el nombre especificat és més gran que la longitud de cadena, la instrucció retorna cadena.

string **mid**(<*string* cadena>, <*num* mida>, <*num* posició>): retorna mida dels caràcters de cadena a partir del caràcter d'índex posició, parant-se al final de cadena. Es genera un error d'execució si mida o posició són negatius.

string **insert**(<*string* cadena>, <*string* inserció>, <*num* posició>): retorna cadena en la qual inserció està inserit després del caràcter d'índex posició. Si posició és més gran que la mida de cadena, s'afegeix inserció al final de cadena. El resultat es trunca en 128 caràcters. Es genera un error d'execució si posició és negatiu.

string **delete**(<*string* cadena>, <*num* mida>, <*num* posició>): retorna cadena en la qual mida de caràcters han estat suprimits a partir del caràcter d'índex posició. Si posició és superior a la longitud de cadena, la instrucció retorna cadena. Es genera un error d'execució si mida o posició són negatius.

string **replace**(<*string* cadena>, <*string* substitució>, <*num* mida>, <*num* posició>): retorna cadena en la qual mida dels caràcters han estat substituïts a partir del caràcter de l'índex de posició per substitució. Si posició és superior a la longitud de cadena, la instrucció retorna cadena. Es genera un error d'execució si mida o posició són negatius.

num **find**(<*string* cadena1>, <*string* cadena2>): retorna l'índex (entre 0 i 127) del primer caràcter de la primera ocurrència de cadena2 a cadena1. Si cadena2 no consta a cadena1, la instrucció retorna -1.

num **len**(<*string* cadena>): retorna la mida de la cadena.

Dio

El tipus *dio* permet vincular una variable VAL3 a una entrada-sortida tot o res del sistema.

Les entrades-sortides declarades al sistema són directament utilitzables en una aplicació VAL3, sense haver-les de declarar com a variables globals o locals en una aplicació. Per tant, el tipus *dio* servirà sobretot per entrar els paràmetres d'un programa que utilitzi una entrada-sortida.

Qualsevol instrucció que utilitzi una variable de tipus *dio* no vinculada en una entrada-sortida declara en el sistema genera un error d'execució.

De manera predeterminada, una variables de tipus *dio* no està vinculada a cap entrada-sortida i si és utilitzada tal qual en un programa genera un error d'execució.

Instruccions:

void **dioLink**(<*dio*& variable>, <*dio* origen>): connecta variable a l'entrada-sortida del sistema a l'origen de la qual està enllaçada. Es genera un error d'execució si origen és una entrada-sortida protegida del sistema.

num **dioGet**(<*dio* dvector>): retorna el valor numèric de dvector de *dio* llegit com un nombre enter escrit en binari, és a dir: dvector[0] + 2 * dvector[1] + 4 * dvector[2] + ...

+ $2^k * \underline{dvector}$, on $\underline{dvector}[i] = 1$ si $\underline{dvector}[i]$ és “true”, 0 en el cas contrari. Es genera un error d’execució si un membre de dvector no està connectat a una entrada-sortida del sistema.

num dioSet(<*dio dvector*>, <*num valor*>): assigna la part sencera de valor escrita en binari a les sortides tot o res de dvector, i retorna el valor efectivament assignat, és a dir: $\underline{dvector}[0] + 2 * \underline{dvector}[1] + 4 * \underline{dvector}[2] + \dots + 2^k * \underline{dvector}[k]$, on $\underline{dvector}[i] = 1$ si dvector es “true”, 0 en cas contrari. Es genera un error d’execució si un membre de dvector no està connectat a una entrada-sortida del sistema.

Aio

El tipus *aio* permet enllaçar una variable VAL3 a una entrada-sortida numèrica del sistema (digital o analògic).

Les entrades-sortides declarades en el sistema són directament utilitzables en una aplicació VAL3, sense haver-la de declarar com a variable global o local en l’aplicació. Per tant, el tipus *aio* servirà sobretot per entrar els paràmetres d’un programa que utilitzi una entrada-sortida.

Qualsevol instrucció que utilitzi una variable de tipus *aio* no vinculada a una entrada-sortida declarada en el sistema genera un error d’execució.

De manera predeterminada, una variable de tipus *aio* no està vinculada a cap entrada-sortida i si és utilitzada tal qual en un programa genera un error d’execució.

Instruccions:

void aioLink(<*aio& variable*>, <*aio origen*>): connecta variable a l’entrada-sortida del sistema a l’origen de la qual està enllaçada. Es genera un error d’execució si origen és una entrada-sortida protegida en el sistema.

num aioGet(<*aio entrada*>): retorna el valor numèric d’entrada. Es genera un error d’execució si entrada no està enllaçada a una entrada-sortida del sistema.

num aioSet(<*aio sortida*>, <*num valor*>): assigna valor a sortida, i retorna valor. Es genera un error d’execució si sortida no està enllaçada a una sortida del sistema.

Sio

El tipus *sio* permet vincular una variable VAL3 a una entrada-sortida sèrie del sistema o una connexió mitjançant socket Ethernet. Una entrada-sortida *sio* es caracteritza per:

- Paràmetres propis al tipus de comunicació, definits en el sistema
- Un caràcter de fi de cadena, per fer possible la utilització del tipus *string*
- Un temps d’espera de comunicació

Les entrades-sortides sèrie del sistema estan sempre actives. Les connexions mitjançant socket Ethernet s’activen en el primer accés en lectura o escriptura en per mitjà d’un

programa VAL3. Les connexions mitjançant socket Ethernet es desactiven automàticament quan s'acaba una aplicació VAL3.

Les entrades-sortides declarades en el sistema són directament utilitzables en una aplicació VAL3, sense haver-les de declarar com a variable global o local en l'aplicació. Per tant, el tipus *sio* servirà sobretot per entrar els paràmetres d'un programa que utilitzi una entrada-sortida.

Qualsevol instrucció que utilitzi una variable de tipus *sio* no vinculada a una entrada-sortida declarada en el sistema genera un error d'execució.

De manera predeterminada, una variable de tipus *sio* no està vinculada a cap entrada-sortida i si és utilitzada tal qual en un programa genera un error d'execució.

Instruccions:

void **sioLink**(<*sio*& variable>, <*sio* origen>): enllaça variable a l'entrada-sortida sèrie del sistema sobre el qual hi està enllaçat origen. Es genera un error d'execució si origen és una entrada-sortida protegida pel sistema.

num **clearBuffer**(<*sio* entrada>): buida el buffer de lectura d'entrada i retorna la quantitat de caràcters eliminats. Per una connexió socket Ethernet, **clearBuffer** desactiva (tanca) el socket, **clearBuffer** retorna -1 si el socket ja estava desactivat. Es genera un error d'execució si entrada no està enllaçat a una connexió sèrie o socket Ethernet del sistema.

num **sioGet**(<*sio* entrada>, <*num*& dades>): Llegeix un vector de caràcters a entrada i retorna la quantitat de caràcters llegits. La lectura s'atura quan el vector dada està ple o quan el buffer de lectura d'entrada està buit. Per una connexió socket Ethernet, **sioGet** primer intenta establir connexió si és que aquesta ja no estava activa. Quan s'arriba al temps d'espera de comunicació d'entrada, **sioGet** retorna -1. Si la connexió està activa però no hi ha dades al buffer de lectura d'entrada, **sioGet** espera fins que es rebin les dades o que el temps d'espera es compleixi. Es genera un error d'execució si entrada no està enllaçat a una connexió sèrie o socket Ethernet del sistema o si dades no és una variable VAL3.

num **sioSet**(<*sio* sortida>, <*num*& dades>): escriu un vector de caràcters a sortida i retorna la quantitat de caràcters escrits. Els valors numèrics són convertits abans de la transmissió en enters entre 0 i 255, prenent l'enter més proper a mòdul 256. Per una connexió socket Ethernet, **sioSet** primer intenta establir connexió si és que aquesta ja no estava activa. Quan s'arriba al temps d'espera de comunicació d'entrada, **sioSet** retorna -1. La quantitat de caràcters escrits pot ser inferior a la mida de dades si es detecta un error de comunicació. Es genera un error d'execució si sortida no està enllaçat a una connexió sèrie o socket Ethernet del sistema o si dades no és una variable VAL3.

Tipus Estructurats

Joint

Un punt articular (tipus *joint*) defineix la posició angular de cada eix del robot.

El tipus *joint* és un tipus estructurat els camps del qual, presentats en ordre, són:

- *num j1*: posició articular de l'eix 1
- *num j2*: posició articular de l'eix 2
- *num j3*: posició articular de l'eix 3
- *num j...*: posició articular de l'eix ... (un camp per eix)

Aquests camps s'expressen en graus pels eixos de rotació i en mil·límetres pels eixos lineals. L'origen de cada eix es defineix segons el tipus de braç utilitzat.

De manera predeterminada, cada camp d'una variable de tipus *joint* s'inicialitza en el valor 0.

Instruccions:

joint abs(*<joint posició>*): retorna el valor absolut d'un camp *posició*, camp per camp.

joint herej(*<>*): retorna la posició actual del camp del braç.

bool isInRange(*<joint jposició>*): prova si la posició d'un camp s'ajusta als límits de software de camp del braç.

void setLatch(*<dio entrada>*): activa el bloqueig del robot sobre el proper flanc ascendent del senyal d'*entrada*.

bool getLatch(*<joint& jposició>*): llegeix l'última posició bloquejada del robot.

Trsf

Una transformació (tipus *trsf*) descriu la posició i la orientació d'un sistema de referència cartesià respecte a un altre sistema de referència.

El tipus *trsf* és un tipus estructurat els camps del qual, presentats en ordre, són:

- *num x*: component en translació segons l'eix x
- *num y*: component en translació segons l'eix y
- *num z*: component en translació segons l'eix z
- *num rx*: component en rotació al voltant de l'eix x
- *num ry*: component en rotació al voltant de l'eix y
- *num rz*: component en rotació al voltant de l'eix z

Els camps *x*, *y* i *z* s'expressen en la unitat de longitud de l'aplicació (mil·límetre o pol·sada). Els camps *rx*, *ry* i *rz* s'expressen en graus.

Les coordenades \underline{x} , \underline{y} i \underline{z} són les coordenades cartesianes de l'origen del pla, respecte al sistema que serveix de referència. Quan \underline{rx} , \underline{ry} , i \underline{rz} són nuls, ambdós plans tenen la mateixa orientació.

De manera predeterminada, una variable *trsf* s'inicialitza en el valor {0,0,0,0,0,0}.

Instruccions:

num distance(*<trsf posició1>*, *<trsf posició2>*): retorna la distància entre posició1 i posició2.

Frame

El tipus *frame* permet definir la posició de sistemes de referència a la cèl·lula.

El tipus *frame* és un tipus estructurat amb un únic camp accessible:

- *trsf* trsf: posició del pla en el seu sistema de referència

El sistema de referència d'una variable de tipus *frame* es defineix en el moment de la seva inicialització (des de la interfície d'usuari o per l'operador =). El sistema de referència "world", de tipus *frame*, està sempre definit en una aplicació VAL3: tot pla està, directament o a través d'altres sistemes de referència, vinculats al sistema "world".

Es genera un error d'execució quan s'efectua qualsevol càlcul geomètric si les coordenades del pla "world" han estat modificades.

De manera predeterminada, una variable de tipus *frame* utilitza "world" com a sistema de referència.

Instruccions:

num setFrame(*<point origen>*, *<point eixOx>*, *<point plaOxy>*, *<frame& marca de referència>*): calcula les coordenades de marca de referència a partir del seu origen, d'un punt eixOx a l'eix (Ox), i d'un punt plaOxy al pla (Oxy). El punt eixOx ha d'estar del costat de les x positives. El punt plaOxy ha d'estar del costat de les y positives. La funció retorna: 0, sense error; -1, el punt eixOx està massa a prop de l'origen; -2, el punt plaOxy està massa a prop de l'eix (Ox). Es genera un error d'execució si un dels punts no té un sistema de referència.

Tool

El tipus *tool* permet definir la geometria i l'acció d'una eina.

El tipus *tool* és un tipus estructurat en camps i en l'ordre següent:

- *trsf* trsf: posició del punt del centre d'eina (TCP) en la seva eina de base
- *dio* gripper: sortida tot o res que serveix per accionar l'eina
- *num* otime: temps d'obertura de l'eina (segons)
- *num* ctime: temps de tancament de l'eina (segons)

L'eina bàsica d'una variable de tipus *tool* queda definida al inicialitzar-la (des de la interfície d'usuari o amb l'operador =). L'eina bàsica "flange", de tipus *tool*, està sempre definida en una aplicació VAL3: tota eina està, directament o a través d'altres eines, vinculada a l'eina "flange".

Es genera un error d'execució quan s'efectua qualsevol càlcul geomètric si les coordenades de l'eina "flange" han estat modificades.

De manera predeterminada, la sortida d'una eina és la sortida io: vàlvula1 del sistema, els temps d'obertura i tancament estan a 0, i l'eina bàsica és "flange".

Instruccions:

void open(<*tool eina*>): acciona *eina* (obertura) posant la sortida digital de l'eina a "true". Abans d'accionar l'eina, *open()* espera que el robot estigui en el punt al fer l'equivalent d'un *waitEndMove()*. Un cop activat, el sistema espera *otime* segons abans d'executar la instrucció següent. Amb aquesta instrucció, no és segur que el robot estigui estabilitzat en la seva posició final abans que l'eina s'activi. Quan l'espera d'estabilització completa del moviment és necessària s'ha d'utilitzar la instrucció *is-Settled()*. Es genera un error d'execució si la *dio* de l'eina no està definida o no és una sortida, o si una instrucció de moviment anteriorment gravada no pot ser executada (destí fora de l'abast).

void close(<*tool eina*>): acciona *eina* (tancament) posant la sortida digital de l'eina a "false". Abans d'accionar l'eina, *close()* espera que el robot estigui en el punt al fer l'equivalent d'un *waitEndMove()*. Un cop activat, el sistema espera *ctime* segons abans d'executar la instrucció següent. Amb aquesta instrucció, no és segur que el robot estigui estabilitzat en la seva posició final abans que l'eina s'activi. Quan l'espera d'estabilització completa del moviment és necessària s'ha d'utilitzar la instrucció *is-Settled()*. Es genera un error d'execució si la *dio* de l'eina no està definida o no és una sortida, o si una instrucció de moviment anteriorment gravada no pot ser executada (destí fora de l'abast).

Point

El tipus *point* permet definir la posició i la orientació de l'eina del robot a la cèl·lula.

El tipus *point* és un tipus estructurat en camps i en l'ordre següent:

- *trsf trsf*: posició del punt en el seu sistema de referència
- *config config*: configuració del braç per arribar a la posició

El sistema de referència d'un *point* és una variable de tipus *frame*, que es defineix al iniciar-la (des de la interfície d'usuari, per l'operador = i les instruccions *here()*, *appro()* i *compose()*).

Es genera un error d'execució si s'utilitza una variable de tipus *point* el sistema del qual no estigui definit.

Instruccions:

num **distance**(*<point* posició1, *point* posició2): retorna la distància entre posició1 i posició2. Es genera un error d'execució si posició1 o posició2 no tenen pla definit.

point **compose**(*<point* posició, *<frame* marca de referència, *<trsf* transformació): retorna posició a la qual està aplicada la transformació geomètrica transformació definida respecte a marca de referència. El sistema de referència i la configuració del punt retornat són els de posició. Es genera un error d'execució si posició no té sistema de referència definit.

point **apro**(*<point* posició, *<trsf* transformació): retorna un punt calculat per transformació geomètrica d'una posició d'entrada. Les coordenades de transformació es proporcionen en la mateixa base que la posició d'entrada (la base del marc de referència de la posició d'entrada). El marc de referència i la configuració del punt retornat corresponen als de la posició d'entrada. Es genera un error d'execució si posició no té sistema de referència definit.

point **here**(*<tool* eina, *<frame* marca de referència): retorna la posició actual d'eina a marca de referència (posició ordenada i no la posició mesurada). El sistema de referència del punt retornat és marca de referència. La configuració del punt retornat és la configuració en curs del braç.

point **joinToPoint**(*<tool* eina, *<frame* marca de referència, *<joint* posició): retorna la posició d'eina a marca de referència quan el braç està a la posició articular posició. El sistema de referència del punt retornat és marca de referència. La configuració del punt retornat és la configuració del braç a la posició articular posició.

bool **pointToJoint**(*<tool* eina, *<joint* inicial, *<point* posició, *<joint&* coordenades): calcula les coordenades articulares que corresponen a la posició especificada. Retorna "true" si s'han trobat unes coordenades articulares, "false" si no existeix una solució. La posició articular cercada respecta la configuració de posició. Els camps de valor "free" no imposen la configuració. Els camps de valor "same" imposen la mateixa configuració que inicial. Pels eixos que poden fer més d'una volta, hi ha diverses solucions articulares que tenen exactament la mateixa configuració: en aquest cas s'adopta la solució més propera a inicial. No hi pot haver una solució si posició està fora de l'abast (braç massa curt) o fora dels marges dels softwares. Si posició especifica una configuració, pot estar fora dels marges per aquesta configuració, però dins dels marges per una altra configuració. Es genera un error d'execució si posició no té sistema de referència definit.

trsf **position**(*<joint* posició, *<frame* marca de referència): retorna les coordenades de posició a marca de referència. Es genera un error d'execució si posició no té sistema de referència.

Config

En general, un robot té diverses possibilitats per arribar a una determinada posició cartesiana.

Aquestes diferents possibilitats s'anomenen "configuracions". A la figura A2.1 es representen 2 configuracions diferents:

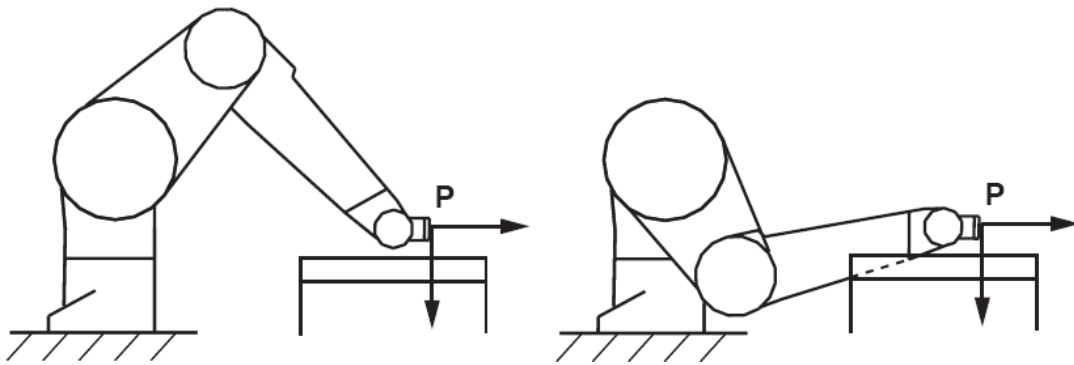


Figura A2.1. Dues configuracions diferents per arribar a un mateix punt

En alguns casos és important especificar, entre totes les configuracions possibles, les que són vàlides i les que es vol prohibir. Per resoldre aquest problema, el tipus *point* permet especificar les configuracions admeses pel robot gràcies al seu camp de tipus *config* que es defineix tot seguit.

El tipus *config* permet definir les configuracions autoritzades per una determinada posició cartesiana.

Depèn del tipus de braç utilitzat.

Per un braç RX/TX el tipus *config* és un tipus estructurat, els camps del qual són, en aquest ordre:

- shoulder: configuració de l'espatlla
- elbow: configuració del colze
- wrist: configuració del canell

Els camps shoulder, elbow i wrist poden prendre els valors següents:

- shoulder:
 - o righty: configuració de l'espatlla righty imposada
 - o lefty: configuració de l'espatlla lefty imposada
 - o ssame: canvi de configuració de l'espatlla prohibit
 - o sfree: configuració de l'espatlla lliure
- elbow:
 - o epositive: configuració del colze epositive imposada
 - o enegative: configuració del colze enegative imposada
 - o esame: canvi de configuració del colze prohibit
 - o efree: configuració del colze lliure
- wrist:
 - o wpositive: configuració del canell wpositive imposada
 - o wnegative: configuració del canell wnegative imposada
 - o wsame: canvi de configuració del canell prohibit
 - o wfree: configuració del canell lliure

Instruccions:

config **config**(<*joint* posició>): retorna la configuració del robot per la posició articular posició.

Mdesc

El tipus *mdesc* permet definir els paràmetres d'un moviment (velocitat, acceleració, allisat).

El tipus *mdesc* és un tipus estructurat els camps del qual, presentat en ordre, són:

- *num accel*: acceleració articular màxima autoritzada, en % de l'acceleració nominal del robot.
- *num vel*: velocitat articular màxima autoritzada, en % de la velocitat nominal del robot.
- *num decel*: desacceleració articular màxima autoritzada, en % de la desacceleració nominal del robot.
- *num rvel*: velocitat màxima autoritzada de translació del centre de l'eina, en mm/s o polsades/s, segons la unitat de longitud de l'aplicació.
- *blend blend*: mode d'allisat: off (sense allisat), o joint (allisat).
- *num leave*: en el mode d'allisat joint, la distància al punt objectiu on comença l'allisat fins el punt següent, en mm o polsades segons la unitat de longitud de l'aplicació.
- *num reach*: en el mode d'allisat joint, la distància al punt objectiu on s'acaba l'allisat fins el punt següent, en mm o polsades segons la unitat de longitud de l'aplicació.

Per defecte, una variable de tipus *mdesc* s'inicialitza a {100,100,100,9999,9999,off,50,50}.

A2.2.5. Constants

Una constant és una dada definida directament en un programa VAL3, sense declaració prèvia. Una constant té un tipus que ve determinat implícitament pel sistema.

A2.2.6. Variables

Definició

Una variable és una dada a la qual es fa referència en un programa pel seu nom.

Una variable es caracteritza per:

- el seu nom: una cadena de caràcters
- el seu tipus: un dels tipus VAL3 descrits anteriorment
- la seva mida: per un vector, el seu nombre d'elements
- el seu abast: el o els programes que poden utilitzar la variable

El nom d'una variable és el d'una cadena de 1 a 15 caràcters entre “a..zA..Z0..9_”.

Totes les variables es poden utilitzar com a vectors. Les variables senzilles tenen una mida de 1. La instrucció **size()** permet saber la mida d'una variable.

Abast d'una Variable

L'abast d'una variable pot ser:

- global: tots els programes de l'aplicació poden utilitzar la variable, o
- local: la variable és accessible únicament des del programa en el que està declarada

Quan una variable global i una local tenen el mateix nom, el programa on està declarada la variable local utilitzarà la variable local, i no tindrà accés a la variable global.

Els paràmetres d'un programa són variables locals, accessibles únicament en el programa en el que estan declarats.

Accés al Valor d'una Variable

Els elements d'un vector són accessibles amb un índex entre els claudàtors '[' i ']'. L'índex ha d'estar comprès entre 0 i (mida - 1); en cas contrari es genera un error al realitzar l'execució.

Si no s'especifica cap índex, s'utilitza l'índex 0: `var[0]` és equivalent a `var`.

Els camps de les variables de tipus estructurat són accessibles per un '.' seguit del nom del camp.

Paràmetre Passat “per Valor”

Quan es passa un paràmetre “per valor”, el sistema crea una variable local i la inicialitza amb el valor de la variable o de l'expressió subministrada pel programa sol·licitant.

Les variables del programa sol·licitant utilitzades com paràmetres “per valor” no canvien, fins i tot si el programa cridat modifica el valor del paràmetre.

Un vector de dades no pot ser passat per valor.

Paràmetre Passat “per Referència”

Quan es passa un paràmetre “per referència”, el programa deixa de treballar sobre una còpia de la dada passada pel sol·licitant, fen-t'ho sobre la pròpia dada, a la que senzillament es canvia de nom localment.

Les variables del programa sol·licitant utilitzades com a paràmetres “per referència” canvien de valor quan el programa cridat modifica el valor del paràmetre.

Tots els elements d’un vector passat per referència poden ser utilitzats i modificats. Si es passa un element d’un vector per referència, aquest element i tots els elements següents poden ser utilitzats i modificats. El paràmetre és llavors vist com un vector que comença per l’element passat al fer la crida. La instrucció `size()` permet conèixer la mida efectiva d’un paràmetre.

Quan es passa “per referència” una constant o una expressió, una assignació del paràmetre corresponent no tindrà cap efecte: el paràmetre conservarà el valor de la constant o de l’expressió.

A2.2.7. Tasques

Definició

Una tasca és un programa que s’està executant.

En una aplicació, es trobarà especialment una tasca pels desplaçaments del braç, una tasca automàtica, una tasca per la interfície d’usuari, una tasca pel seguiment dels senyals de seguretat, tasques de comunicació...

Una tasca es caracteritza pels elements següents:

- un nom: un identificador únic de tasca dins la biblioteca o l’aplicació
- una prioritat o un període: paràmetre per la seqüenciació de les tasques
- un programa: punt d’entrada (i sortida) de la tasca
- un estat: actiu o aturat
- la propera instrucció a executar (i el seu context)

A2.3. Instruccions de Control de Seqüència

Comentari: //

Crida a un subprograma: **call**

Retorn de subprograma: **return**

Instruccions de control:

```
if <bool condició>  
<instruccions>  
[else  
<instruccions>]  
endIf
```

```
while <bool condició>  
<instruccions>
```

endWhile

do <instruccions>
until <bool condició>

for <num comptador> = <num inici> to <num final> [step <num pas>]
<instruccions>
endFor

switch <num selecció>
case <num cas1> [, <num cas2>]
<instruccions1-2>
break
[**case** <num cas3> [, <num cas4>]
<instruccions3-4>
break]
[**default**
<instruccions predeterminades>
break]
endSwitch

A2.4. Instruccions de Moviment

void **movej**(<joint joint>, <tool eina>, <mdesc desc>), *void* **movej**(<point punt>, <tool eina>, <mdesc desc>): registra una ordre de moviment articular fins a la posició joint o punt, amb eina i els paràmetres de moviment desc.

void **movel**(<point punt>, <tool eina>, <mdesc desc>): registra una ordre de moviment lineal fins la posició punt, amb eina i els paràmetres de moviment desc.

void **movec**(<point punt intermedi>, <point punt d'arribada>, <tool eina>, <mdesc desc>): grava una instrucció de moviment circular partint del punt de destí del moviment anterior fins el punt d'arribada passant pel punt intermedi. L'orientació de l'eina s'interpolava de manera que és possible programar una orientació constant en absolut o respecte a la trajectòria.

void **stopMove**(): atura el braç a la trajectòria i suspèn l'autorització de moviment programada. Els paràmetres cinemàtics utilitzats per efectuar l'aturada són els del moviment en curs. Els moviments només es poden reprendre després d'executar una instrucció **restartMove**() o **resetMotion**(). Els moviments no programats (desplaçaments manuals) són possibles.

void **resetMotion**(), *void* **resetMotion**(*joint* sortida): atura el braç a la trajectòria i anul·la totes les ordres de moviments registrats. L'autorització de moviment programat es pot restaurar si havia estat suspesa per la instrucció **stopMove**(). Si s'especifica la posició articular sortida, la propera ordre de moviment només es podrà executar a partir d'aquesta posició: prèviament s'haurà d'efectuar un moviment d'enllaç per arribar a sortida. Si no s'especifica cap posició articular, la propera ordre de moviment s'efectuarà a partir de la posició en curs del braç, fos la que fos.

void restartMove(): restaura l'autorització de moviment programat, i reprèn la trajectòria interrompuda per la instrucció **stopMove()**. Si l'autorització de moviment programat no ha estat interrompuda per la instrucció **stopMove()**, aquesta instrucció no té cap efecte.

void waitEndMove(): anul·la l'allisat de l'última ordre de moviment registrada i espera que aquesta ordre s'hagi executat. Aquesta instrucció no espera que el robot s'estabilitzi a la seva posició final, únicament que la consigna de la posició enviada als variadors correspongui amb la posició final desitjada. Quan l'espera d'estabilització completa del moviment és necessària, s'ha d'utilitzar la instrucció **isSettled()**. Es genera un error d'execució si una ordre de moviment anteriorment registrada no pot ser executada (destí fora de l'abast).

bool isEmpty(): retorna "true" si totes les ordres de moviment s'han executat, "false" si queda com a mínim una ordre en curs.

bool isSettled(): retorna "true" si el robot està aturat, "false" si la seva posició no està encara estabilitzada. La posició es considera estabilitzada quan l'error de posició a cada eix roman, durant 50 ms, inferior a l'1% de l'error de posició màxima autoritzada.

void autoConnectMove(<bool actiu>), *bool autoConnectMove()*: en mode desalineat, el moviment d'enllaç és automàtic si el braç està molt a prop de la seva trajectòria (distància inferior a l'error d'arrossegament màxim autoritzat). Si el braç està massa allunyat de la seva trajectòria, el moviment d'enllaç serà automàtic o amb control manual segons el mode definit per la instrucció **autoConnectMove**: automàtic si actiu val "true", sota control manual si actiu val "fals". Si es crida sense paràmetre, **autoConnectMove** segueix amb el mode de moviment d'enllaç en curs. De manera predeterminada, el moviment de connexió en mode desplaçat està sota control manual.

A2.5. Instruccions de Control del Robot

void disablePower(): talla la potència del braç i espera que la potència s'hagi tallat. Si el braç està en moviment, s'efectua una aturada ràpida en trajectòria abans del tall de potència.

void enablePower(): en el mode desplaçat aplica potència al braç. No té cap efecte en els modes local, manual o prova, o quan la potència s'està tallant.

bool isPowered(): retorna l'estat de la potència del braç: "true", el braç està amb potència; "false", el braç està exempt de potència, en curs d'aplicació de potència o de tall.

bool isCalibrated(): retorna l'estat de recuperació del robot: "true", tots els eixos del robot estan calibrats; "false", almenys un eix del robot no està calibrat.

num workingMode(), *num workingMode(num& estat)*: retorna el mode de funcionament en curs del robot. Al manual hi ha tots els modes explicats.

num speedScale(): retorna la velocitat de monitor en curs.

num esStatus(): retorna l'estat del circuit de senyals de seguretat. Al manual hi ha els codis explicats.

A2.6. Instruccions de la Pàgina d'Usuari

Les instruccions de la interfície d'usuari del llenguatge VAL3 permeten:

- la visualització de missatges en una pàgina del MCP (comandament manual) reservada a l'aplicació
- l'adquisició de les presions-tecles al teclat del MCP

La pàgina d'usuari té 14 línies de 40 columnes. L'última línia pot ser utilitzada per realitzar menús amb la tecla associada. Es troba disponible una línia addicional per la visualització d'un títol.

void userPage(), *void userPage(<bool fix>)*: fa que es presenti la pàgina de l'usuari a la pantalla del MCP. Si el paràmetre *fix* és "true", només la pàgina d'usuari serà accessible a l'operador, a excepció de la pàgina de canvi de perfil accessible per la tecla d'accés ràpid "Shift User". Quan aquesta pàgina es mostra, és possible aturar l'aplicació amb la tecla "Stop" si el perfil d'usuari en curs ho autoritza. Si el paràmetre és "false", les altres pàgines del CS8 tornen a ser accessibles.

void gotoxy(<num x>, <num y>): col·loca el cursor de la pàgina d'usuari a les coordenades (*x*,*y*). La cantonada de dalt a l'esquerra té les coordenades (0,0) i la cantonada de baix a la dreta (39,13). L'abscissa *x* es pren mòdul 40. L'ordenada *y* es pren mòdul 14.

void cls(): esborra la pàgina d'usuari i col·loca el cursor a (0,0).

void put(<string cadena>), *void put(<num valor>)*, *void putln(<string cadena>)*, *void putln(<num valor>)*: mostra, a la posició del cursor de la pàgina d'usuari, la *cadena* o *valor* especificat (amb 3 xifres després de la coma). Tot seguit, el cursor es col·loca damunt el primer caràcter que segueix el darrer caràcter del missatge visualitzat (instrucció *put*), o sobre el primer caràcter de la línia següent (instrucció *putln*). Al final de la línia, la visualització segueix a la línia següent. Al final de la pàgina, la visualització de la pàgina d'usuari es desplaça una línia cap amunt.

void title(<string cadena>): canvia el títol de la pàgina d'usuari. La posició actual del cursor no es modifica per la instrucció *title()*.

num get(<string& cadena>), *num get(<num& valor>)*, *num get()*: fa l'adquisició d'una *cadena*, un número o una tecla al teclat de la consola. *Cadena* o *valor* es mostren a la posició en curs del cursor i pot ser modificat per l'usuari. L'entrada s'acaba prement una tecla de menú, de "Return" o de "Esc". La instrucció retorna el codi de la tecla que ha acabat l'entrada de dades. Quan es prem "Return" o un menú, la variable *cadena* o *valor* s'actualitza. Quan es prem "Esc", no canvia. Si no ha passat cap paràmetre, la instrucció *get()* espera que es premi qualsevol tecla i retorni el seu codi. La tecla premuda no apareix en pantalla. En tots els casos, la posició en curs del cursor no es modifica per la instrucció *get()*.

num **getKey()**: fa l'adquisició d'una tecla al teclat de la consola. Retorna el codi de la darrera tecla premuda després de la darrera crida a **getKey()**, retorna -1 si no s'ha premut cap tecla. A diferència de la instrucció **get()**, **getKey()** torna de seguida. La tecla premuda no es mostra i la posició en curs del cursor no es modifica.

bool **isKeyPressed(<num codi>)**: retorna l'estat de la tecla especificada pel seu codi, "true" si es prem la tecla, "false" en cas contrari.

void **popUpMsg(<string cadena>)**: presenta cadena en una finestra "popup" per sobre de la finestra en curs del MCP. Aquesta finestra es manté visualitzada fins que sigui validada pel menú "OK" o la tecla "Esc".

void **logMsg(<string cadena>)**: escriu cadena a l'historial del sistema. El missatge serà gravat amb la data i l'hora en curs.

string **getProfile()**: retorna el nom del perfil d'usuari en curs.