# Robòtica:
# Raiden Project

Bartomeu Miró Mateu *
Lluis Cortès Rullan †

22 de maig de 2011

Primera pràctica de Robòtica de l'apartat de robòtica industrial. Control d'un braç robot Mitsubishi RV-6S programat en MELFA Basic IV destinant a la paletització de peces colorides circulars.

*bartomeumiro a gmail punt com
†lluisbinet a gmail punt com

# Índex

# 1 Interpretació de l'enunciat i modelat de l'escenari

En aquest apartat s'expliquen les extensions i suposicions de l'enunciat original, així com el modelat de l'entorn del robot.

## 1.1 Interpretació de l'enunciat

Tal i com es demana el programa està parametritzat segons la posició de la càmera, el punt $P$ que aquesta detecta i l'angle $\alpha$. A més existeixen altres paràmetres com els punts on es troben les peces originalment i on es deixen al final.

Per altra banda s'ha introduït la possibilitat de fixar un nombre diferent de peces a cada pila, per tant es poden tenir piles amb diferent número de peces. També és té amb compte la possibilitat de tenir més d'una peça, o cap, d'algun tipus.

Tots aquests paràmetres es tornen a veure detallats en l'apartat d'explicació del codi on s'expliquen les *Macros pròpies* 4.1.

## 1.2 Modelat de l'entorn

L'enunciat deixa oberta la possibilitat de que fer amb les peces de tipus 4, en aquest punt s'ha optat per posar-les al pal 4 per aprofitar el codi ja escrit i així seguir la coherència i estructura dels tipus de peça anteriors. El fet de no optar per apilar-les en qualsevol punt de l'entorn es perquè en la paletització ja s'ha demostrat coneixement de com apilar peces i tractar el tipus 4 de manera diferent als anteriors minvava elegància al codi i l'execució.

En la figura 1.2 següent es poden veure tots els paràmetres i punts (marcats amb una estrella) que caracteritzen l'escenari.

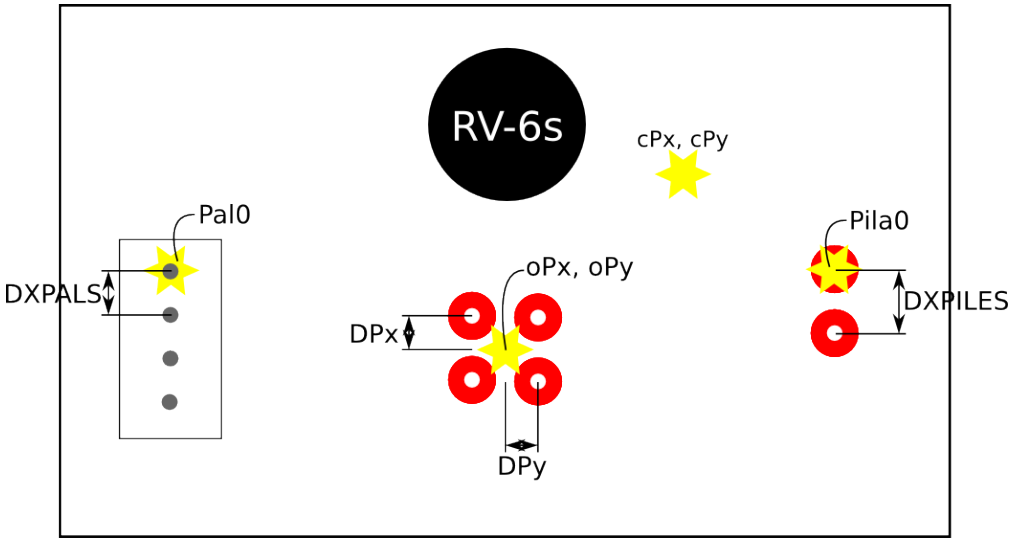


Figura 1: Escenari del robot

La figura es veu complementada amb el llistat de punts emprats a la pràctica.

```
DEF POS Paralisi = (337.16, -14.95, 270.30,-179.98, -0.36,-177.39)(7,0)
DEF POS Pal0     = (308.00,-550.00, 280.00, 171.00,-57.00, -79.00)(7,0)
DEF POS Pila0    = (340.18, 481.51, 280.00,-179.98, -0.36, -90.00)(7,0)
DEF POS Pale0    = (  0.00,   0.00, 280.00,-179.98, -0.36, -90.00)(7,0)
DEF POS PaleOut0 = (337.00,-450.00, 280.00, 171.00,-57.00, -79.00)(7,0)
```

Com es detalla en l'apartat de *calcul de punts* 3.2 s'ha intentat minimitzar el nombre de punts capturats per tal de calcular els demés en relació a aquests.

A continuació s'explica la utilitat de cada punt o posició.

**Paralisi** Guarda la posició del robot en repòs.

**Pal0** Posició del braç sobre el primer pal (amb la pinça tombada).

**Pila0** Posició del braç sobre la primera pila (amb pinça perpendicular)

**Pale0** Orientació del braç robot per posar peces del palé (amb la pinça perpendicular).

**PaleOut0** Orientació del braç robot per agafar les peces del palé (amb la pinça tombada)

### 1.3 Consideracions de l'entorn

A efectes pràctics a la imatge següent es veu el posicionat de les peces d'una forma humanament comprensible emprant les marques fetes per els alumnes sobre l'entorn i existents a data de 22 de maig de 2011.

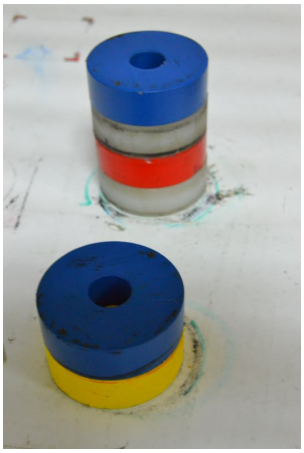

Figura 2: Posicionat de les piles a l'entorn

A la imatge es pot apreciar com la primera pila està al centre fet amb retolador negre i la segona seguint el cercle fet amb bolígraf *Bic* blau.

Degut a l'esgotament de les bateries dels codificadors angulars[1] l'han hagut de re-calular les orientacions del braç, això provoca un des-calibrat. El cas es que després del re-calibrat del braç sempre quedava uns mil·límetres curt en l'eix $Y$ per la introducció de les peces de tipus 4. Donat que l'eix $Y$ es suposadament constant als quatre pals s'ha optat per moure la base en lloc d'ofuscar el codi amb un cas especial. Concretament s'ha de rotar el suport dels pals sobre el Pal0 en sentit anti-horari quedant el peu del costat del pal 3 com indica la figura 1.3.

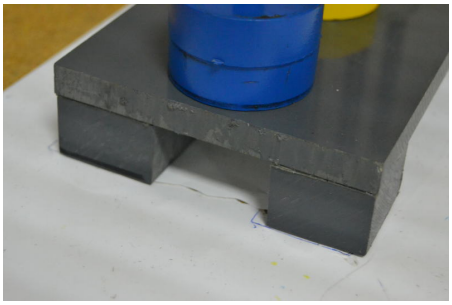

Figura 3: Pals moguts, correcció després de l'esgotament de les bateries

Aquest fet es deu molt possiblement a lleugeres modificacions de l'orientació de la pinça que s'han hagut de re-calcular. La inversió de temps que suposa el re-ajustament, que ja va fer-se al seu moment abans de l'incident, s'ha considerat escesiva i duplicada podent arreglar la situació amb un simple moviment de la base dels pals.

[1]Una setmana abans de l'entrega quan la pratica ja estava llesta uns alumnes reportarden el mal funcionament del braç, segons la versió oficial s'havien esgotat unes bateries encarregades d'alimentar els codificadors angulars optics que quarden les posicions de les articulacions del braç. Un com reestablert el robot ha variat lleugerament el seu calibrat, cosa que ha forçata als alumnes que ja tenen la practica llesta a reintroduir les punts i orientacions del braç que tenien guardats.

## 2 Moviment del robot

En aquest punt descrivim el moviment del braç robot i el perquè de l'ordre o orientacions del mateix.

Totes les aproximacions a les peces es fan des de dalt. El braç robot es posiciona a *XY* sobre la peça en qüestió i després efectua un descens en *Z*. Un cop agafada efectua el moviment invers, ascens a Z i després el desplaçament corresponent al pla *XY* on es consideren segurs els moviments. Aquest pla de seguretat en la pràctica ve donat per la variable `ZS` fixada per defecte a 280.0.

Els desplaçaments al pla de seguretat es realitzen a una major velocitat que els descensos que conformen les aproximacions. A l'apartat de *Macros pròpies* 4.1 estan descrites les variables que ho controlen.

En primer lloc el robot agafa les peces del les piles inicials. Aquestes són collocades a la zona de paletització. Ambdues accions és duen a terme amb la pinça perpendicular al pla *XY* (figura 2), ja que es la posició més segura i còmode de programar per l'agafada de peces.

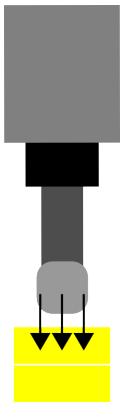

Figura 4: Aproximació amb la pinça perpendicular al pla *XY*

Cal remarcar que la pinça esta mirant cap al vidre protector i la paret, de tal manera que l'operari veu el metall. Si no s'agafés bé la peça i aquesta sortís disparada ho faria en direcció a la paret o el vidre protector, no cap a l'operari.

Un cop acabat el muntatge del palé és desmunta amb la pinça tombada, s'agafen les peces començant per les que es troben més a l'esquerra del braç robot, per collocar-les als pals.

El fet de tombar la pinça és per incrementar l'abast del braç robot. Amb la pinça perpendicular al pla *XY*, com s'havia efectuat el muntatge del palé, no és possible arribar als pals 3 i 4. Com que s'ha decidit tombar la pinça cap a l'esquerra[2], des

[2]Els pals estan a la dreta del robot, per tant convé tombar la pinça a la esquerra per simplificar els moviments i evitar haver de fer un gir del braç

de el punt de vista del robot, s'han de recollir les peces d'esquerra a dreta per tal de evitar que el braç col·lisioni (fig. 2) amb algun dels munts del palé encara existents.

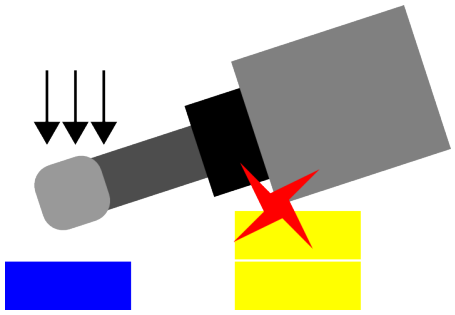

Figura 5: Col·lisió amb el munt de peces grogues intentant agafar l'última peça blava

Així doncs l'ordre de des-paletització depèn de l'angle i no del tipus de peça. Com és pot veure a la figura 2 existeixen vuit possibles casos que es veuen reflectits en l'explicació del codi font corresponent, secció 4.2.3.

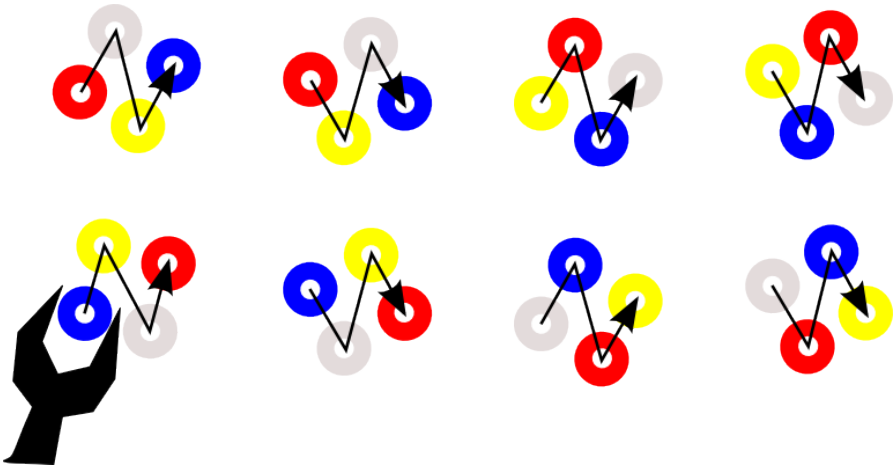

Figura 6: Ordre de recollida de les peces segons l'angle $\alpha$

## 3 Càlculs

L'apartat de càlculs inclou els requerits per l'enunciat a l'haver de transformar el punt P del sistema de coordenades de la càmera al del braç robot. En aquest punt es mostra el procés de obtenció de les equacions per fer tal transformació que seran implementades al programa.

Per altra banda es mostra el càlcul dels punts emprats de la situació dels objectes en l'entorn a partir dels seus homòlegs. Com s'ha mencionat la posició dels pals es relativa a un punt, concretament tots els pals on es deixen les peces venen en funció de la posició del primer. De la mateixa manera la posició de cada pila on inicialment es recullen les peces ve en funció de la posició de la primera pila, el nombre de piles i la distància entre elles.

Finalment també es descriu el càlcul de punts en Z on s'agafen les peces en funció del pla de terra, l'altura de les peces i la quantitat de les mateixes.

### 3.1 Transformació de sistemes de coordenades

Amb l'enunciat tenim la posició de la càmera {C} respecte el robot {R} i la posició del punt P, que es el centre de sistema de coordenades del palé {P}, respecte la càmera. El que volem es posar el punt P en el sistemes de coordenades del robot.

Així doncs definim:

${}^{R}T_{C}$ La matriu de transformació d'un punt del robot a la càmera

${}^{C}T_{P}$ La matriu de transformació d'un punt de la càmera al palé

${}^{P}P$ Punt en el sistema de coordenades del palé

${}^{R}P$ Punt en el sistema de coordenades del robot

Combinant aquestes matrius obtenim que:

$$ {}^{R}T_{C} \times {}^{C}T_{P} \times {}^{P}P = {}^{R}P $$

Per tant hem de cercar ${}^{R}T_{P}$:

$$ {}^{R}T_{P} = {}^{R}T_{C} \times {}^{C}T_{P} $$

$$ {}^{R}T_{P} = \left(\begin{array}{ccc|c} 0 & 1 & 0 & cPx \\ 1 & 0 & 0 & cPy \\ 0 & 0 & -1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \times \left(\begin{array}{ccc|c} cos\phi & -sin\phi & 0 & oPx \\ sin\phi & cos\phi & 0 & oPy \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right) $$

$$ {}^{R}T_{P} = \left(\begin{array}{ccc|c} sin\phi & cos\phi & 0 & oPy + cPx \\ cos\phi & -sin\phi & 0 & oPx + cPy \\ 0 & 0 & -1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right) $$

Així doncs per passar d'un punt del sistema de coordenades ${}^{P}P$ a ${}^{R}P$ ho feim així:

$$ {}^{R}T \times {}^{P}P = {}^{R}P $$

En el cas genèric agafam ${}^{P}P = \left(\begin{array}{c} x \\ y \\ z \\ \hline 1 \end{array}\right)$

$$ \left(\begin{array}{ccc|c} sin\phi & cos\phi & 0 & oPy + cPx \\ cos\phi & -sin\phi & 0 & oPx + cPy \\ 0 & 0 & -1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \times \left(\begin{array}{c} x \\ y \\ z \\ \hline 1 \end{array}\right) = \left(\begin{array}{c} x \times sin\phi + y \times cos\phi + cPy + cPx \\ x \times cos\phi - y \times sin\phi + oPx + cPy \\ -z \\ \hline 1 \end{array}\right) $$

De on extreim les següents equacions que seran implementades dins del nostre programa (secció 4.2.1).

$$ {}^{R}P_X = x \times sin\phi + y \times cos\phi + cPy + cPx $$
$$ {}^{R}P_Y = x \times cos\phi - y \times sin\phi + oPx + cPy $$
$$ {}^{R}P_Z = -z $$

### 3.2 Càlcul de piles, pals (X,Y) i altura de peces (Z)

A banda de la transformació de punts entre sistemes de coordenades hi ha altres punts a calcular. El primer són les piles d'on s'agafen les peces. Dins del llistat de punts tenim el punt de la primera pila, les següents es troben situades a un desplaçament en *X*. Així doncs en el nostre escenari senzillament s'ha de incrementar en *X* el valor de la distància entre piles. Succeeix exactament el mateix amb els pals on es deixen les peces.

$$ Pila_i.X = Pila_0.X + i \times \triangle X_{pila} $$
$$ Pila_i.Y = Pila_0.Y + i \times \triangle Y_{pila} $$
$$ Pal_i.X = Pal_0.X + i \times \triangle X_{pal} $$
$$ Pal_i.Y = Pal_0.Y + i \times \triangle Y_{pal} $$

Els detalls de implementació i valors de les constants de separació es poden veure en l'apartat especific 4.2.3 on s'explica el codi font. Cal remarcar que al codi també figura un desplaçament en Y, per si es volguessin posar les piles o pals fent una diagonal, però per defecte la *macro* està inicialitzada a 0.

## 4 Estructura del programa

El programa es divideix en cinc seccions.

**Capçalera de l'enunciat** Interfície de *macros* requerida per l'enunciat on hi ha paràmetres com l'angle $\alpha$.

**Macros pròpies** Re-assignació de valors per les funcionalitats extra que s'han implementat com ara emprar piles amb nombre de peces diferents, si es vol fer una execució amb l'enunciat original sols fa falta assignar els valors de les variables de la secció anterior.

**Declaracions pròpies** Aquí es declaren les variables globals pròpies del programa, en principi sols les ha de tocar el programador.

**Rutines** Codi de les rutines on es desenvolupa tota la pràctica.

**Main** Simple crida ordenada a les rutines per desencadenar l'execució.

### 4.1 Macros, paràmetres

Les *macros*[3] de la *capçalera de l'enunciat* ja estan explicades al propi enunciat, aquí s'expliquen les altres *macros* introduïdes per les funcionalitats pròpies, corresponents a l'apartat de *Macros pròpies*.

**PINCA** Valor de la pinça emprada, en el nostre cas el robot només en té una i és la 1.

**DPALE** Retard (*D*elay) de 4 segons requerit a l'enunciat un cop s'ha muntat el palé.

**D#PINCA%** Retards aplicats després de *O*#brir o *T*#ancar la pinça en funció de si es *I*%nicial o *F*%inal, abans o després d'agafar la peça.

**VLENT** *V*elocitat a la qual es duen a terme les aproximacions delicades, com l'agafada de peces.

**VNORMAL** *V*elocitat a la que es desplaça normalment el braç en els trajectes segurs.

**ZS** Pla *Z* on es consideren segurs els moviments del braç robot i on es duen a terme els desplaçaments llargs.

**ZTPT** Pla de *T*erra ($Z = 0$) amb la *P*inça *T*ombada

**ZTPR** Pla de terra amb la pinça perpendicular al pla *XY*

[3] Cal mencionar que aquests valors no són realment *macros* sinó variables, una *macro* es resol en temps de pre-processador no en temps de compilació o execució, el fet de que el llenguatge no disposi de *macros* porta alguns problemes mencionats a l'apartat de incidents 6.

**TP** *T*ipus de *P*eça diferents que tenim, es igual el número de munts del palé i de pals de destí.

**PILES** Número de piles d'on es fa la recollida inicial de peces.

**PALS** Número de pals de destí on es deixen les peces.

**DT** Retard per l'obertura i tancament de la pinça.

**HDISC** Altura de les peces (discs)

**D#PILES** Distància en $X$ i $Y$ entre les piles

**D#PALS** Distància en $X$ i $Y$ entre els pals

**DZPAL** Distància $Z$ a baixar per introduir la peça dins el pal.

**RELPALE(T,E)** Distància relativa de la peça tipus $T$ al punt $P$, en funció de l'eix $E$ (1 per $X$, 2 per $Y$)

### 4.2 Estructura del codi

En el codi s'ha fet una funció per cada acció i alguna d'auxiliar transversal. La intenció es dedicar tota la feina a les rutines i que la lectura del programa sigui el més natural possible. En els nivells més alts entrant en detalls a mesura que es va aprofundint en les funcions.

Així doncs tenim aquesta jerarquia:

| Munta palé | | Desmunta palé | |
|---|---|---|---|
| Agafa pila | Posa palé | Agafa Palé | Posa pal |
| Obre/tanca pinça | | | |

Cal fer especial menció a dos vectors de control de l'estat del programa, aquests són `npcspila` i `alloc`.

`npcspila(p)` conté el nombre de peces que resten a la pila inicial d' on s'agafen.

`alloc(m)` indica quantes peces ja estan *a lloc*, situades al munt `p` del palé, recordem que cada munt correspon a un tipus de peça.

Ambdues variables serveixen per controlar l'altura en $Z$ d'on agafar les peces, i per sabre quan ja no en queden més.

Amb aquest principi ens queda un cos principal tan lleuger com aquest.

```
970 gosub *INIT
980 gosub *CALCPTS
990 gosub *MNTPALE
1000 dly DPALE
1010 gosub *DESPALE
1020 gosub *ACABA
1030 end
```

### 4.2.1 Auxiliars

**Canvi sistemes de coordenades** En la funció `CAM2ROB` es troba implementat el canvi de sistemes de coordenades del la càmera al robot. Com a interfície llegeix els punts `ppx` i `ppy` que són els que es desitgen transformar, així com les *macros* del programa `oPx`, `oPy`, `cPx`, `cPy` i l'`angle`. Deixa els valors de `ppx` i `ppy` en funció del sistema de coordenades del braç robot en les variables `prx` i `pry`.

```
2020 *CAM2ROB
2030    angleR =(angle*M_PI) / 180.0
2040    prx = ppx*sin(angleR) + ppy*cos(angleR) + oPy + cPx
2050    pry = ppx*cos(angleR) - ppy*sin(angleR) + oPx + cPy
2060    return
```

**Obertura-Tancament de pinça** Per tal de facilitar la llegibilitat del codi i no afegir constantment els retards per la obertura i tancament de pinça s'han definit les funcions `OPINCA` i `TPINCA` que encapsulen aquest comportament.

```
2080 *OPINCA
2090    dly DOPINCAI
2100    hopen PINCA
2110    dly DOPINCAF
2120    return
```

```
2130 *TPINCA
2140    dly DTPINCAI
2150    hclose PINCA
2160    dly DTPINCAF
2170    return
```

### 4.2.2 Inicialitzacions

En aquesta rutina es duu a terme l'operació de *homing* i demés inicialitzacions físiques del braç robot com l'obertura de la pinça o de les variables que controlen l'entorn del robot, com el comptador de peces que s'han posat a cada palé.

```
1050 *INIT
1060    servo ON
1070    ovrd VNORMAL
1080    mov Paralisi
1090    gosub *OPINCA
1100    for i = 1 to TP
1110        alloc(i) = 0
1120    next
1130    return
```

### 4.2.3 Càlcul de punts

Com ja s'ha comentat en la pràctica s'ha fet el màxim esforç per posar tots els punts en funció d'uns pocs, per això en la funció de càlcul de punts s'omplen els vectors que contenen els punts homòlegs calculats a partir de les *macros* que descriuen l'entorn físic. En aquest punt també es calcula l'ordre de recollida de les peces en funció de l'angle (figura 2). Per limitacions del llenguatge comentades a l'apartat 6 d'incidents el codi no has sortir gaire elegant.

Un cop acabada la funció *càlcul de punts* tots els vectors de punts estan apunt per ser recorreguts per part del braç robot en els eixos (X,Y), el Z és calculat en temps d'execució ja que depèn de quina peça s'estigui tractant, ja que son apilades a l'eix Z. Aquests vectors emprats de interfície són:

**`Pila`** Cada component indica la posició (X,Y) de cada una de les piles d'on s'agafen inicialment les peces, en l'execució per defecte seran dues.

**`Pale`** Cada component correspon a la posició on anirà col·locat cada tipus de peça al palé.

**`PalI`** Posició inicial on es col·lcoca el braç per iniciar el descens sobre el pal.

**`PalF`** Punt final de descens del braç on es considera que la peça ja hi ha entrat i pot ser amollada.

```
1140 *CALCPTS
1150    for pil = 1 to PILES
1160        Pila(pil) = Pila0
1170        Pila(pil).x = Pila(1).x + DXPILES * (pil - 1)
1180        Pila(pil).y = Pila(1).y + DYPILES * (pil - 1) 'es 0
1190    next

1210    for tipus = 1 to TP
1220        ppx = RELPALE(tipus, 1) 'paramentres de entrada de CAM2ROB
1230        ppy = RELPALE(tipus, 2)
1240        gosub *CAM2ROB           'prx i pry son parametres de sortida
1250        Pale(tipus) = Pale0    'Escriu altres components i orientacions
1260        Pale(tipus).x = prx
1270        Pale(tipus).y = pry
1280        PaleOut(tipus) = PaleOut0
1290        PaleOut(tipus).x = prx '- 7.0
1300        PaleOut(tipus).y = pry - 15.0
1310    next

1330    for pal = 1 to PALS
1340        PalI(pal) = Pal0
1350        PalI(pal).x = PalI(pal).x + DXPALS * (pal - 1)
1360        PalI(pal).y = PalI(pal).y + DYPALS * (pal - 1)
```

```
1370        PalF(pal) = PalI(pal)
1380        PalF(pal).z = PalF(pal).z - DZPAL
1390    next
1400
1410    angle = angle mod 360
1420    if ((0.0 < angle) and (angle < 45.0)) then
1430        ordre(1) = 3
1440        ordre(2) = 2
1450        ordre(3) = 4
1460        ordre(4) = 1
1470        return
1480     endif
1490    if ((45.0 < angle) and (angle < 90.0)) then
1500        ordre(1) = 3
1510        ordre(2) = 4
1520        ordre(3) = 2
1530        ordre(4) = 1
1540        return
1550    endif
1560    if ((90.0 < angle) and (angle < 135.0)) then
1570        ordre(1) = 4
1580        ordre(2) = 3
1590        ordre(3) = 1
1600        ordre(4) = 2
1610        return
1620    endif
1630    if ((135.0 <= angle) and (angle < 180.0)) then
1640        ordre(1) = 4
1650        ordre(2) = 1
1660        ordre(3) = 3
1670        ordre(4) = 2
1680        return
1690    endif
1700    if ((180.0 <= angle) and (angle < 225.0)) then
1710        ordre(1) = 1
1720        ordre(2) = 4
1730        ordre(3) = 2
1740        ordre(4) = 3
1750        return
1760    endif
1770    if ((225.0 <= angle) and (angle < 270.0)) then
1780        ordre(1) = 1
1790        ordre(2) = 2
1800        ordre(3) = 4
1810        ordre(4) = 3
1820        return
1830    endif
1840    if ((270.0 <= angle) and (angle < 315.0)) then
1850        ordre(1) = 2
```

```
1860        ordre(2) = 1
1870        ordre(3) = 4
1880        ordre(4) = 3
1890        return
1900    endif
1910    ' 315.0 > angle < 360.0
1920    ordre(1) = 2
1930    ordre(2) = 3
1940    ordre(3) = 1
1950    ordre(4) = 4
1960    return
```

### 4.2.4 Munta palé

Tal i com indica el títol en aquesta funció es munta el palé, aquesta tasca consisteix en iterar per cada una de les peces existents en cada una de les piles per agafar-les i dipositar-les al lloc corresponent del palé.

```
2180 *MNTPALE
2190    peca = 1
2200    for pil = 1 to PILES
2210        for pecapila = 1 to npcspila(pil)
2220            gosub *AGAFPILA
2230            gosub *POSAPALE
2240            peca = peca + 1
2250        next
2260    next
2270    return
```

#### Agafa de la pila

```
2380 *AGAFPILA
2390    mov Pila(pil)
2400    Prvsnl = Pila(pil)
2410    Prvsnl.z = Prvsnl.z - ZTPR + (npcspila!(pil) - pecapila) * HDISC
2420    ovrd VLENT
2430    mvs Prvsnl
2440    gosub *TPINCA
2450    mvs Pila(pil)
2460    ovrd VNORMAL6
2470    return
```

### Posa al Palé

```
2480 *POSAPALE
2490    mov Pale(tipusP(peca))
2500    Prvsnl = Pale(tipusP(peca))
2510    Prvsnl.z = Prvsnl.z - ZTPR + alloc(tipusP(peca)) * HDISC
2520    ovrd VLENT
2530    mvs Prvsnl
2540    gosub *OPINCA
2550    mvs Pale(tipusP(peca))
2560    ovrd VNORMAL
2570    alloc(tipusP(peca)) = alloc(tipusP(peca)) + 1
2580    return
```

### 4.2.5 Desmunta palé

En aquest punt del programa el braç va munt a munt del palé agafant les peces i deixant-les als pals, no passa al següent munt mentre a l'actual quedin peces. L'ordre d'agafada dels munts ha estat calculat en *càlcul de punts* com ja s'ha esmentat (secció 4.2.3).

```
2290 *DESPALE
2300    for tipus = 1 to TP
2310        munt = ordre(tipus)
2320        while alloc(munt) > 0
2330            gosub *AGAFPALE
2340            gosub *POSADEST
2350        wend
2360    next
2370    return
```

### Agafa del palé

```
2590 *AGAFPALE
2600    mov PaleOut(munt)
2610    Prvsnl = PaleOut(munt)
2620    Prvsnl.z = Prvsnl.z - ZTPT + (alloc(munt) - 1) * HDISC
2630    ovrd VLENT
2640    mvs Prvsnl
2650    gosub *TPINCA
2660    mvs PaleOut(munt)
2670    ovrd VNORMAL
2680    alloc(munt) = alloc(munt) - 1
2690    return
```

#### Posa pal

```
2700 *POSADEST
2710    mov PalI(munt)
2720    ovrd VLENT
2730    mvs PalF(munt)
2740    gosub *OPINCA
2750    mvs PalI(munt)
2760    ovrd VNORMAL
2770    return
```

### 4.2.6 Acaba

De manera complementària a l'inicialitza, *acaba* s'encarrega del posicionament físic del robot en una posició de repòs i la desconnexió dels servomotors.

```
1970 *ACABA
1980    mov Paralisi
1990    hopen PINCA
2000    servo OFF
2010    return
```

## 5 Joc de proves

Per tal de provar la pràctica s'han realitzat una sèrie de jocs de proves descrits a continuació.

Cal esmentar que al no poder oferir una traça de l'execució aquest apartat sols serveix per mencionar proves interessants a fer per validar qualsevol canvi el programa.

La primera prova es realitza amb l'enunciat per defecte, sols dos munts i quatre peces, una de cada tipus. A continuació es duen a terme rotacions del palé.

Cada una de les rotacions es fa per comprovar que el braç agafi les peces amb l'ordre que toca sense col·lisonar amb els demés munts. Així doncs es realitzen vuit execucions corresponents als vuit casos (figura 2) de recollida que s'han esmentat a l'apartat d'explicació de l'enunciat.

A continuació es realitza la prova de posar les quatre peces del mateix tipus, aquest test serveix per comprovar si funciona correctament el desplaçament en $Z$. Per altra banda amb aquest test es comprova que passa quan no existeix algun tipus de peça en la seqüència.

Finalment es prova l'execució amb piles de diferent dimensió (4 i 2) amb peces repetides del mateix color.

Cal esmentar que tots els tests foren passats abans de l'escotament de les bateries que guardaven els valors dels codificadors angulars òptics. Després del reestabliment del robot sols s'han fet proves amb les piles de diferent dimensió amb $\alpha = 133$ i $\alpha = 240$ [4] ja que dur a terme els tests es una inversió de temps considerable.

[4] Aquestes dues exuecions es poden visualitzar a:
`http://www.youtube.com/watch?v=fPLusSjFok4`
`http://www.youtube.com/watch?v=ENp9qKfcj20`

# 6 Incidents, problemàtica al desenvolupament i valoració

Durant el desenvolupament com es habitual hom es troba amb problemes a resoldre, alguns d'ells fruit de l'aprenentatge però d'altres deguts a limitacions de l'entorn ja sigui físic com virtual.

## 6.1 Problemàtica física

El primer problema a afrontar era fer arribar les peces als pas 3 i 4. S'entén que aquest fet es intencionat per complicar una mica més la pràctica. En el nostre cas això ha estat resolt tombant la pinça com ja s'ha exposat en les seccions anteriors. El fet de tombar la pinça porta una sèrie complicacions addicionals. L'una és que el sistema de coordenades de la pinça no està ubicat perfectament a la zona de contacte on s'agafen les peces, això fa que s'hagi de fer una petita correcció en $XY$ respecte a la posició quan està perpendicular. Aquesta correcció és veu al codi com una resta de 17.0 a $Y$, no definit com a *macro* per resignació.

La sol·lució no es gaire elegant però si efectiva, tal vegada si el treball a realitzar fos més complexe consideraríem intentar moure el sistema de coordenades al punt correcte, ja sigui amb la funció `tool` com fent un càlcul propi a mà i tenint una funció similar a la `CAM2ROB` (secció 4.2.1).

En l'entorn físic també ens hem trobat amb el problema que les peces tenen restes de cola i vegades s'aferren entre elles movent dues peces a la vegada, cosa que desbarata totalment l'execució i requereix un reseteig manual de l'entorn.

S'entén que tots aquests problemes físics són fruit de viure en un món material amb certes complicacions que, al cap i a la fi, fan la vida un poc més interessant. Per altra banda si entram en l'apartat de problemes o incidents en el programari ens trobam amb coses menys justificables.

## 6.2 Problemàtica virtual

El llenguatge de programació del robot es un autèntic desastre, incòmode i poc pràctic. A classe s'ha comentat que una de les problemàtiques de la robòtica industrial és el temps que és té el robot aturat per programar-lo i d'aquí que s'inverteixi molt de temps en simuladors, que no poden, entre d'altres, simular els problemes físics esmentats anteriorment. Així doncs seria molt més eficaç tenir millors llenguatges que no indueixin a l'error i facilitin un desenvolupament més ràpid que no pas dedicar-se a simuladors.

Concretament al llenguatge li sobren coses com el numerat de línia, sobretot que l'entorn no ho faci de manera completament automàtica. Requerir numerat només fa que s'hagi de pitjar el botó de re-enumeració cada cop i esperar que es faci el procés, que incomprensiblement tarda uns 2 o 3 segons.

Al llenguatge també són necessaris més elements, el primordial són variables locals i crides de funcions amb un fàcil pas de paràmetres, les variables globals i el `gosub` indueixen freqüentment a l'error.

Falta de *macros*, en tot programa, i més els que tracten entorns físics, són necessaris números màgics, aquests poden ser resolts amb una variable però si són necessaris per la declaració d'un vector tenim un problema, i més si el llenguatge no permet que aquests siguin dinàmics. És trist haver de posar números màgics per tot amb un comentari el costat on aparegui algun identificador per poder-los localitzar al llarg del codi quan s'ha de realitzar algun canvi.

Tampoc estaria de més el suport de la funció de `case` amb `floats` o com a mínim l'entorn podria aixecar un advertiment en ser usat, dient que no funciona.

Finalment comentar que el sistema de mostrat d'errors pobre a més de sovint pràcticament inexistent.

## 6.3 Valoració

Aquests comentaries venen perquè l'algorisme i modelat de la pràctica s'elaboraren en unes hores. El gruix de temps invertit ha estat ajustant paràmetres del robot i barallant-se amb el llenguatge i entorn. Així i tot ha estat d'agrair el fet de poder estar amb contact amb un braç real. Cal dir que tot i així la pràctica ha estat amena i entretinguda, però en aquests temps convé més ser critic per millorar encara més el que és té, intentant que cada cop la Universitat sigui un millor lloc per el creixement de l'estudiant.



El document en versió digital i el codi font el trobareu a
`https://github.com/bmiro/raiden`

Aquest document i tota la part de la practica que s'ha pogut ha estat desenvolupat emprant programari lliure:
LaTeX i Kile per el text, Inkscape pels diagrames, Kate i Vim per l'edició del codi font. Git com a sistema de control de versions.





# 7 Annex: Codi font complet

```
10 ''''''''''''''''''''''''Capçalera requerida a l'enunciat''''''''''''''''''''''''
20 tool(0,0,190,0,0,0)
30 def INTE GROC, BLANC, BLAU, VERMELL
40 VERMELL = 1
50 BLANC = 2
60 GROC = 3
70 BLAU = 4

80 def INTE N 'veure variable pcsPila de la seccio de declaracions propies
90 N = 3
100 dim tipusP!(6)
110 tipusP(1) = BLAU
120 tipusP(2) = BLANC
130 tipusP(3) = VERMELL
140 tipusP(4) = BLANC
150 tipusP(5) = BLAU
160 tipusP(6) = GROC

170 'coordenades x i y del punt P respecte la camera
180 def FLOAT oPx, oPy
190 oPx = 0.0
200 oPy = 0.0
210 'angle de rotació del palé respecte la camera
220 def FLOAT angle
230 angle = 133.0
240 'coordenades x i y de l'origen del sistema de la camera
250 def FLOAT cPx, cPy
260 cPx = 550.0
270 cPy = 0.0

280 ''''''''''''''''''''''''     Macros pròpies     ''''''''''''''''''''''''
290 def INTE PINCA
300 PINCA = 1

310 def FLOAT DPALE 'Delay despres de haver munta tel pale
320 DPALE = 4.0

330 def FLOAT DOPINCAI, DOPINCAF, DTPINCAI, DTPINCAF 'emps abans i despres per obrir/tancar pi
340 DOPINCAI = 1.0 'Es important estar ben estatic el deixar
350 DOPINCAF = 0.5
360 DTPINCAI = 0.5
370 DTPINCAF = 0.5

380 def INTE VLENT, VNORMAL
390 VLENT = 10
400 VNORMAL = 30

410 def FLOAT ZS, ZTPT, ZTPR 'Pla Z on es desplaça el robot per sobre de la terra _real_
420 ZS = 280.0
430 ZTPT = ZS - 10.0 ' Pla de terra amb pinça tombada
440 ZTPR = ZS + 6.5  ' Pla de terra amb pinça recta
```

```
52
53
54  450 def INTE TP 'tipus de pença diferents
55  460 TP = 4
56
57  470 def INTE PILES 'Numero de pilaes inicials on hi ha les peces
58  480 PILES = 2
59
60  490 def INTE PALS 'Numero de pals
61  500 PALS = 4
62
63  510 def FLOAT DT 'Diferencial de temps per la pinça
64  520 DT = 0.5
65
66  530 def FLOAT HDISC 'Altura dels discs
67  540 HDISC = 17.5
68
69  550 'TODO documentar cap on intrementen les x i tot l'entorn, premises de colocacio etc.
70  560 def FLOAT DXPILES 'Distancia X entre les pilaes
71  570 DXPILES = 100.0
72  580 def FLOAT DYPILES 'Distancia Y entre les pilaes
73  590 DYPILES = 0.0
74  600 def FLOAT DXPALE  'Distancia X entre els punts del pale
75  610 DXPALE  = 110.0
76  620 def FLOAT DYPALE  'Distancia Y entre els punts del pale
77  630 DYPALE = 110.0
78  640 def FLOAT DXPALS  'Distancia X entre els pals
79  650 DXPALS  = 79.0
80  660 def FLOAT DYPALS  'Distancia Y entre els pals
81  670 DYPALS  = 0.0
82  680 def FLOAT DZPAL   'Distancia Z per entrar dins el pal
83  690 DZPAL  = 45.0
84
85  700 dim RELPALE# (4,2) '4 = TP, posicions relatives al centre per fer el palé (x, y)
86  710 RELPALE (1,1) = -55.0
87  720 RELPALE (1,2) =  55.0
88  730 RELPALE (2,1) =  55.0
89  740 RELPALE (2,2) =  55.0
90  750 RELPALE (3,1) =  55.0
91  760 RELPALE (3,2) = -55.0
92  770 RELPALE (4,1) = -55.0
93  780 RELPALE (4,2) = -55.0
94
95  790 ''''''''''''''''''''''''     Declaracions pròpies     ''''''''''''''''''''''''
96  800 dim npcsPila!(2) '2 = PILES
97  810 npcsPila(1) = N+1
98  820 npcsPila(2) = N-1
99
100 830 dim ordre!(4) '4 = TP Ordre de recollida de les peces
101
102 840 dim alloc!(4) '4 = TP Tantes posicions com tipus de peça, les peces que ja estan a lloc
103
104 850 def POS Prvsnl 'Punt Provisional per calcul
105
```

```
106  860 dim Pale(4) '4 = TP, posicio segura sobra la columna d'on s'ha de posar la peça
107  870 dim PaleOut(4)
108  880 dim Pila(2) '2 = PILES posicio segura sobre la pila
109  890 dim PalI(4) ' 4 = PALS
110  900 dim PalF(4) ' 4 = PALS
111
112  910 def INTE i, pil, pal, munt, peca, pecapila, tipus 'iteradors
113
114  920 def FLOAT angleR
115  930
116  940 def FLOAT ppx, ppy 'punt pale  ' emprats com a paramentres per el pals de
117  950 def FLOAT prx, pry 'punt robot ' coorenades del pale al robot
118
119  960 '''''''''''''''''''''''''''''''''   MAIN   '''''''''''''''''''''''''''''''''''
120  970 gosub *INIT
121  980 gosub *CALCPTS
122  990 gosub *MNTPALE
123 1000 dly DPALE
124 1010 gosub *DESPALE
125 1020 gosub *ACABA
126 1030 end
127
128 1040 ''''''''''''''''''''''''''''''''    RUTINES   '''''''''''''''''''''''''''''''''
129 1050 *INIT
130 1060    servo ON
131 1070    ovrd VNORMAL
132 1080    mov Paralisi
133 1090    gosub *OPINCA
134 1100    for i = 1 to TP
135 1110        alloc(i) = 0
136 1120    next
137 1130    return
138
139 1140 *CALCPTS
140 1150    for pil = 1 to PILES
141 1160        Pila(pil) = Pila0
142 1170        Pila(pil).x = Pila(1).x + DXPILES * (pil - 1)
143 1180        Pila(pil).y = Pila(1).y + DYPILES * (pil - 1) 'es 0
144 1190    next
145
146 1210    for tipus = 1 to TP
147 1220        ppx = RELPALE(tipus, 1) 'paramentres de entrada de CAM2ROB
148 1230        ppy = RELPALE(tipus, 2)
149 1240        gosub *CAM2ROB           'prx i pry son parametres de sortida
150 1250        Pale(tipus) = Pale0    'Escriu altres components i orientacions
151 1260        Pale(tipus).x = prx
152 1270        Pale(tipus).y = pry
153 1280        PaleOut(tipus) = PaleOut0
154 1290        PaleOut(tipus).x = prx '- 7.0
155 1300        PaleOut(tipus).y = pry - 15.0
156 1310    next
157
158 1330    for pal = 1 to PALS
159 1340        PalI(pal) = Pal0
```

```
160  1350        PalI(pal).x = PalI(pal).x + DXPALS * (pal - 1)
161  1360        PalI(pal).y = PalI(pal).y + DYPALS * (pal - 1)
162  1370        PalF(pal) = PalI(pal)
163  1380        PalF(pal).z = PalF(pal).z - DZPAL
164  1390      next
165  1400
166  1410      angle = angle mod 360
167  1420           if ((0.0 < angle) and (angle < 45.0)) then
168  1430          ordre(1) = 3
169  1440          ordre(2) = 2
170  1450          ordre(3) = 4
171  1460          ordre(4) = 1
172  1470                  return
173  1480           endif
174  1490     if ((45.0 < angle) and (angle < 90.0)) then
175  1500          ordre(1) = 3
176  1510          ordre(2) = 4
177  1520          ordre(3) = 2
178  1530          ordre(4) = 1
179  1540                  return
180  1550           endif
181  1560     if ((90.0 < angle) and (angle < 135.0)) then
182  1570          ordre(1) = 4
183  1580          ordre(2) = 3
184  1590          ordre(3) = 1
185  1600          ordre(4) = 2
186  1610                  return
187  1620           endif
188  1630     if ((135.0 <= angle) and (angle < 180.0)) then
189  1640          ordre(1) = 4
190  1650          ordre(2) = 1
191  1660          ordre(3) = 3
192  1670          ordre(4) = 2
193  1680                  return
194  1690           endif
195  1700     if ((180.0 <= angle) and (angle < 225.0)) then
196  1710          ordre(1) = 1
197  1720          ordre(2) = 4
198  1730          ordre(3) = 2
199  1740          ordre(4) = 3
200  1750                  return
201  1760           endif
202  1770     if ((225.0 <= angle) and (angle < 270.0)) then
203  1780          ordre(1) = 1
204  1790          ordre(2) = 2
205  1800          ordre(3) = 4
206  1810          ordre(4) = 3
207  1820                  return
208  1830           endif
209  1840     if ((270.0 <= angle) and (angle < 315.0)) then
210  1850          ordre(1) = 2
211  1860          ordre(2) = 1
212  1870          ordre(3) = 4
213  1880          ordre(4) = 3
```

```
214  1890                  return
215  1900          endif
216  1910   ' 315.0 > angle < 360.0
217  1920   ordre(1) = 2
218  1930   ordre(2) = 3
219  1940   ordre(3) = 1
220  1950   ordre(4) = 4
221  1960   return
222
223  1970 *ACABA
224  1980     mov Paralisi
225  1990     hopen PINCA
226  2000     servo OFF
227  2010     return
228
229  2020 *CAM2ROB
230  2030          angleR =(angle*M_PI) / 180.0
231  2040     prx = ppx*sin(angleR) + ppy*cos(angleR) + oPy + cPx
232  2050     pry = ppx*cos(angleR) - ppy*sin(angleR) + oPx + cPy
233  2060     return
234
235  2080 *OPINCA
236  2090     dly DOPINCAI
237  2100     hopen PINCA
238  2110     dly DOPINCAF
239  2120     return
240
241  2130 *TPINCA
242  2140     dly DTPINCAI
243  2150     hclose PINCA
244  2160     dly DTPINCAF
245  2170     return
246
247  2180 *MNTPALE
248  2190     peca = 1
249  2200     for pil = 1 to PILES
250  2210         for pecapila = 1 to npcspila(pil)
251  2220             gosub *AGAFPILA
252  2230             gosub *POSAPALE
253  2240             peca = peca + 1
254  2250         next
255  2260     next
256  2270     return
257
258  2290 *DESPALE
259  2300     for tipus = 1 to TP
260  2310         munt = ordre(tipus)
261  2320         while alloc(munt) > 0
262  2330             gosub *AGAFPALE
263  2340             gosub *POSADEST
264  2350         wend
265  2360     next
266  2370     return
267
```

```
2380 *AGAFPILA
2390    mov Pila(pil)
2400    Prvsnl = Pila(pil)
2410    Prvsnl.z = Prvsnl.z - ZTPR + (npcspila!(pil) - pecapila) * HDISC
2420    ovrd VLENT
2430    mvs Prvsnl
2440    gosub *TPINCA
2450    mvs Pila(pil)
2460    ovrd VNORMAL
2470    return

2480 *POSAPALE
2490    mov Pale(tipusP(peca))
2500    Prvsnl = Pale(tipusP(peca))
2510    Prvsnl.z = Prvsnl.z - ZTPR + alloc(tipusP(peca)) * HDISC
2520    ovrd VLENT
2530    mvs Prvsnl
2540    gosub *OPINCA
2550    mvs Pale(tipusP(peca))
2560    ovrd VNORMAL
2570    alloc(tipusP(peca)) = alloc(tipusP(peca)) + 1
2580    return

2590 *AGAFPALE
2600    mov PaleOut(munt)
2610    Prvsnl = PaleOut(munt)
2620    Prvsnl.z = Prvsnl.z - ZTPT + (alloc(munt) - 1) * HDISC
2630    ovrd VLENT
2640    mvs Prvsnl
2650    gosub *TPINCA
2660    mvs PaleOut(munt)
2670    ovrd VNORMAL
2680    alloc(munt) = alloc(munt) - 1
2690    return

2700 *POSADEST
2710    mov PalI(munt)
2720    ovrd VLENT
2730    mvs PalF(munt)
2740    gosub *OPINCA
2750    mvs PalI(munt)
2760    ovrd VNORMAL
2770        return
```