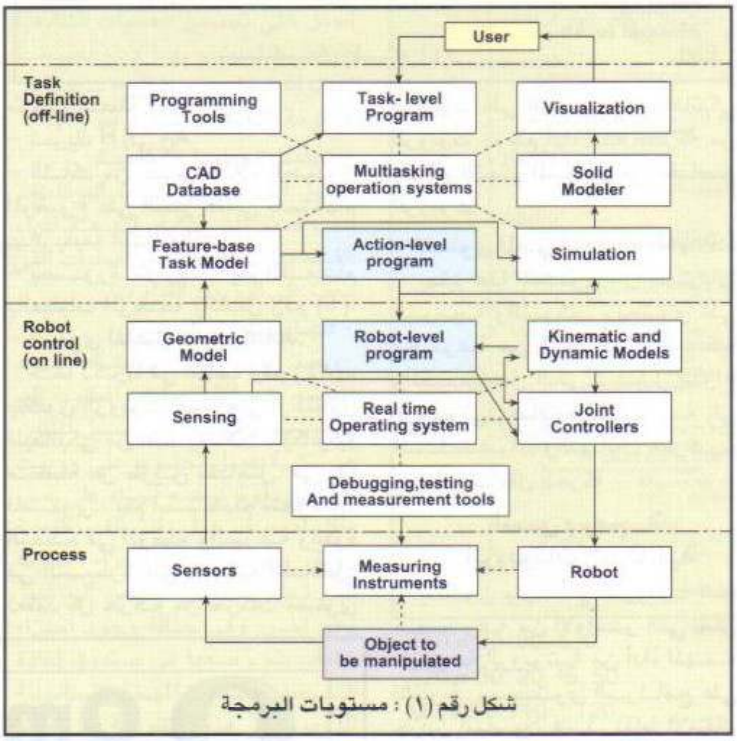
كيف تبنى "روبوت" حقيقى؟

١٩- لغة برمجة الروبوتات الصناعية "RAPID"

د. عـــلاء خميـس
الجامعة الألمانية بالقاهرة

تختلف برمجة الروبوت عن برمجة الحاسب الآلى.. فى أن الروبوت يتعامل مع أغراض مادية ثلاثية الأبعاد فى الفراغ يجب أن يكون لديه القدرة على استشعارها ومناولتها من مكان إلى آخر. وفى بعض الأحيان.. تعتمد الجدوى الاقتصادية لاستخدام الروبوتات الصناعية على الوقت اللازم لبرمجة الروبوت لأداء مهمة ما. ويعتبر وقت البرمجة دالة فى الخصائص التى توفرها لغة البرمجة ودرجة صعوبتها والدعم الذى توفره بيئة البرمجة لاختبار البرنامج واكتشاف الاخطاء. ويجب على لغة البرمجة.. توفير امكانية تعريف الروبوت بالمهمة المراد القيام بها.. بالاضافة الى التحكم فى الروبوت ذاته خلال أدائه تلك المهمة. يطرح هذا المطلب سؤالاً مهما هو.. هل يتم تصميم بيئة البرمجة لتسهيل عملية البرمجة أم لتحسين أداء البرنامج؟ فى كثير من الأحيان.. لا يمكن تحقيق هذين الغرضين معاً.. لذا يتم تصميم بيئة البرمجة حسب مستوى البرمجة المطلوب والموضح بالشكل رقم (١).

فى هذا الشكل.. نجد أنه بالانتقال إلى المستويات الدنيا.. يجب على لغة البرمجة أن تضع أداء الروبوت فى المقام الاول. أما فى المستويات العليا.. فتكون الأولوية لسهولة وسرعة البرمجة. لذا.. يتم فصل برامج التحكم – التى يلعب فيها الأداء او سرعة التنفيذ Time-Critical عاملاً مهماً – عن برامج التخطيط للقيام بالمهمة المطلوبة.



شكل رقم (١) : مستويات البرمجة

وتحتوى منظومات برمجة الروبوتات الصناعية على عدة مستويات هيكلية تعكس طبيعة المهمة المراد القيام بها وطبيعة الروبوت نفسه. يوضح الشكل رقم (٢) التركيب الهيكلى لمنظومات التخطيط والبرمجة والاستشعار والتحكم فى الروبوتات الصناعية.

Level	Planning steps	Program	Model/sensing
System	Production process	Set of parallel tasks.	User's view of the world.
⇓			⇓
Task	What is to be done?	Specification of individual tasks	Process model.
⇓			⇓
Action	Decompose task into a sequence of actions.	Sequence of actions	Task model. Perception.
⇓			⇕
Robot	Decompose actions into a sequence of robot motions in Cartesian space.	Sequence of robot operation primitives.	Geometric model. Sensor fusion.
⇓			⇕
Joint	Decompose robot motions into parallel joint space motions.	Parallel joint motions.	Kinematic model of robot. Physical model of sensors.
⇓			⇑
Physical	⇒ Control of linkages.	⇒ Actuators.	⇒ Sensors.

شكل رقم (٢) : التخطيط والبرمجة والاستشعار والتحكم

– مستوى المنظومة System Level:

يقوم المستخدم بشراء الروبوت لأداء مهمةما أو عدة مهام فى بيئة الانتاج. لذا.. يمكن النظر إلى المنظومة الروبوتية على أنها أداة لتحقيق الأهداف الانتاجية.

– مستوى المهمة Task Level:

يحدد المستخدم بالتفصيل مايريد أن يفعله الروبوت على حسب تفهم المستخدم الكامل للعملية الانتاجية وكيفية توظيف الروبوت لخدمتها. على سبيل المثال.. يوضح الشكل رقم (٣) منظومة تعبئة يقوم فيها الروبوت بالتقاط جزء ما من على سير ناقل ووضعه داخل صندوق.

– مستوى التنفيذ Action Level:

يحتوى هذا المستوى على وصف للخطوات الواجب اتباعها للقيام بمهمة ما. فى المثال السابق.. يجب اتباع الخطوات التالية:

أ– وضع صندوق فارغ فى مكان التعبئة (السير الثانى).

ب– بشكل تكرارى يقوم الروبوت بالعمليات التالية:

– عندما يكون الجزء المراد تعبئته عند نهاية السير الناقل الأول.. يتم التقاطه ووضعه فى الصندوق الموجود على السير الثانى.

– عند امتلاء الصندوق.. يجب تغييره بآخر فارغ من على السير الثالث.

ويجب ربط هذه العمليات بنموذج هندسى لبيئة العمل عن طريق استخدام منظومات إسنادات كارتيزية عند المواضع المهمة واستخدام مصفوفات التحويل لايجاد العلاقة بين منظومات الاسناد المختلفة (انظر العدد رقم ٨٦). فى المثال السابق.. يتم استخدام منظومات إسناد فى المواضع التالية:

A_b : موضع فوق صندوق التعبئة.

A_c : موضع فوق الجزء المراد التقاطه.

B : موضع الصندوق خلال عملية التعبئة على السير الثانى.

C_n : نقطة المنتصف لنهاية السير الناقل الأول.

E : موضع الصندوق الفارغ على السير الثالث.

G_b : نقطة التقاط الجزء من على السير الأول.

H : موضع نقطة مركز الأداة TCP ليد الروبوت (انظر العدد رقم ٨٤).

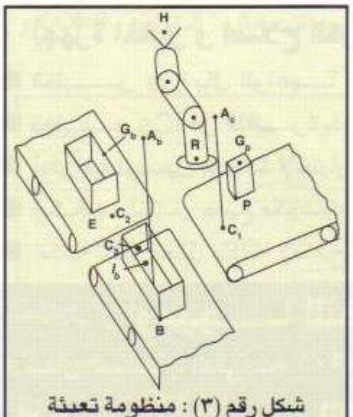
I_b : موضع الجزء داخل صندوق التعبئة على السير الثانى.

P : موضع الجزء على السير الناقل الأول.

R : مركز إسناد مرجعى (عادة ما يكون قاعدة الروبوت الثابتة).

– مستوى الروبوت Robot Level:

فى هذا المستوى.. يتم تجزئة كل عملية من العمليات التتابعية اللازمة



شكل رقم (٣) : منظومة تعبئة

لاتمام المهمة إلى عدة حركات أولية للروبوت مثل.. فتح الملقاط ـ الانتقال إلى G_p – غلق الملقاط ورفع الجزء لأعلى إلخ...

يتم نمذجة الروبوت فى الفراغ الكارتيزى باستخدام كينماتيكا الروبوت (أنظر العدد رقم ٨٦) للحصول على النموذج الكينماتيكى.. وتستخدم البيانات المكتسبة عن طريق المستشعرات عن حالة الروبوت وحالة بيئة العمل فى تكوين نموذج هندسى لخلية العمل. فى المثال السابق.. يتم تجزئة عملية التقاط ووضع الأجزاء إلى الخطوات التالية:

– الانتقال من الموضع الابتدائى للنهاية الطرفية H إلى E_{Gb} (تحريك النهاية الطرفية إلى موضع الصندوق الفارغ).

– التقاط الجزء Grasp(t_p) حيث t_p هو سمك جدار الصندوق.

– تحريك H إلى G_b (تحريك الى موضع التعبئة).

– وضع الصندوق الفارغ فى موضع التعبئة.

– تحريك H إلى A_C.

– التأكد من عدم امتلاء الصندوق الموضوع على السير الثانى باستخدام عداد بقيمة ابتدائية C_p=0.

– بصورة تكرارية.. يتم القيام بالعمليات الموضحة بالشكل رقم (٤).

– مستوى المفصل Joint Level:

كما ذكرنا فى العدد رقم (٨٤).. يتكون الروبوت الصناعى أو المناول الميكانيكى من عدة وصلات ميكانيكية متصلة عن طريق مفاصل. فى هذا المستوى.. يتم استخدام منظومات التحكم فى الموضع والسرعة والقوة فى السيطرة على حركة المفاصل. تعتمد كل حركة من حركات مستوى الروبوت على النموذج الكينماتيكى للروبوت.. ويتم أداء هذه الحركة من خلال التحكم المتزامن فى مفاصل الروبوت.

– المستوى المادى Physical Level:

يعتبر هذا المستوى أقل مستويات البرمجة والتحكم.. ويحتوى على مجموعة من الوصلات الميكانيكية والمفاصل التى تشكل الذراع الروبوتى.. بالاضافة إلى المستشعرات والنهايات الطرفية ومنظومات نقل الحركة.

أسس برمجة الروبوتات الصناعية

يتكون برنامج الروبوت من مجموعة من الأوامر التى تمكن المنظومة الروبوتية من أداء المهمة المطلوبة. ويحتوى البرنامج على جزأين أساسيين هما.. نقاط الانتقال والأوامر البرمجية.

١– نقاط الانتقال:

وهي مجموعة من النقاط الفراغية فى نطاق العمل تسمى بنقاط الانتقال أو الموضع Translation or Position Points.. يجب على الروبوت المرور بها عند تنفيذ البرنامج. تحتوى كل من هذه النقاط على قيمة لكل محور من محاور الروبوت. يوضح الشكل رقم (٥) مثالاً لروبوت اسطوانى Seiko الذى يتمتع بأربع درجات حرية حركة.. مما يتطلب استخدام أربع قيم للتعبير عن نقاط الانتقال تمثل قيم الازاحة لمحاور الروبوت x و y و z (بالملليمتر) بالاضافة إلى زاوية الدوران A (بالدرجات) كما هو مبين بالشكل رقم (٦).

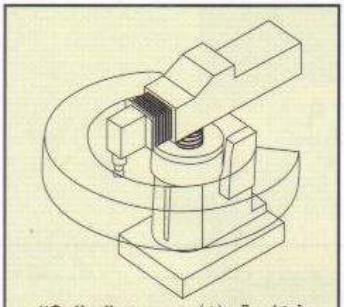
توجد طرق مختلفة لتحديد نقاط الانتقال وتعريفها للروبوت مثل.. استخدام طريقة التلقين أو تعلم نقاط الانتقال عن طريق إحداثيات المدخلات أو باستخدام العلاقات الجبرية أو بواسطة البرمجة خارج الخط.

– فى الطريقة الأولى.. يتم تحريك الروبوت بواسطة أداة تلقين Teach Pendant (انظر العدد رقم ٨٧) حتى الوصول إلى نقطة الانتقال المراد تخزين إحداثياتها.. ثم يتم الضغط على زر التخزين فى أداة التلقين او كتابة الأمر التالى (HERE T<0- 399>) كما هو الحال فى الروبوت Seiko. يسمح هذا الروبوت بتخزين ٤٠٠ نقطة انتقال.

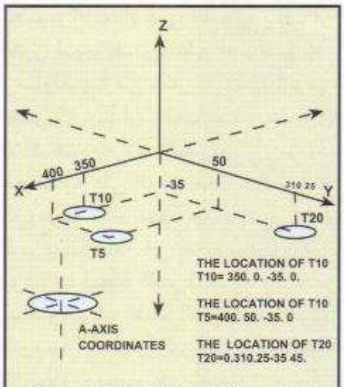
– فى الطريقة الثانية.. يتم إدخال إحداثيات نقطة الانتقال كجزء من البرنامج. على سبيل المثال.. فى حالة الروبوت Seiko يمكن تعريف نقطة الانتقال كالتالى:

```
Loop {forever}
 If Sp Then    {Part present signal}
   Move H to PGp
   Grasp(wp)   {wp is thickness of part}
   If grasp error Exit (Error Flag)
   Move H to Ac    {part conveyor starts to move}
   Move H to Ab
   PlacePart(Ib,full) {calculate where to put part and set flag when box is full}
   Move H to Ib
   If collision Exit(Error Flag)
   Ungrasp
   Move H to Ab
 End
```

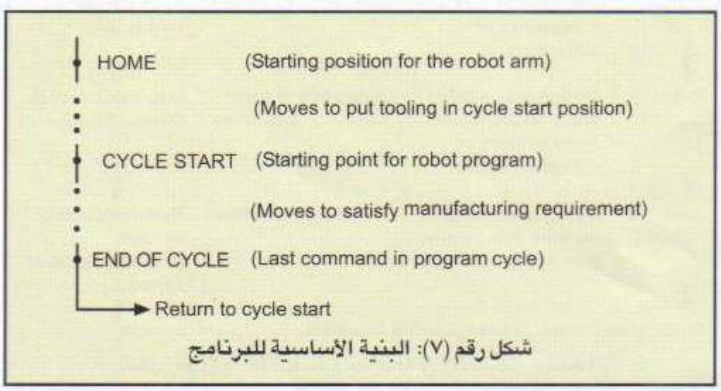
شكل رقم (٤)



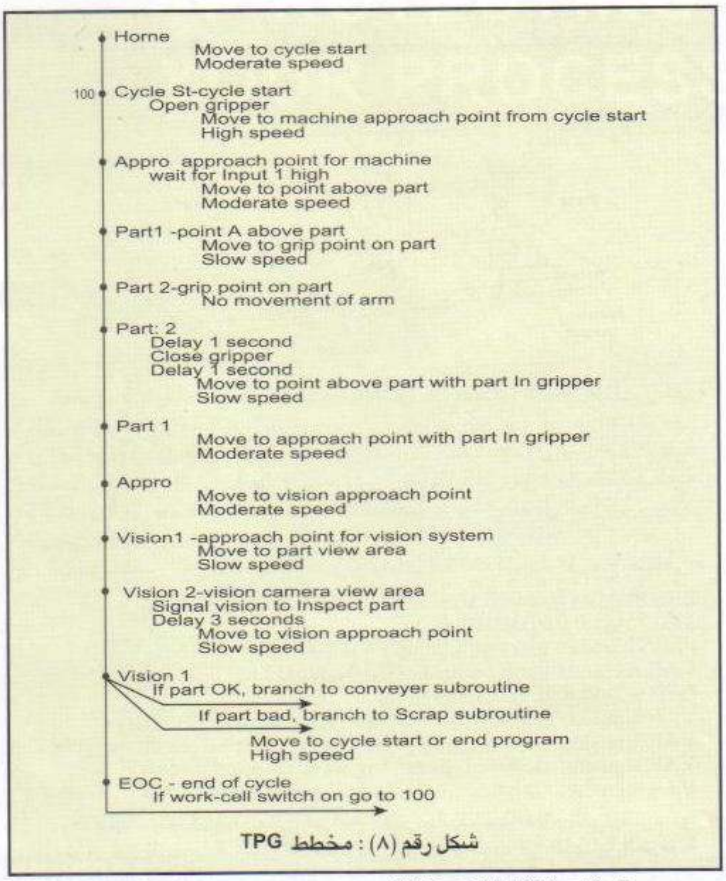
شكل رقم (٥) : روبوت "Seiko"



شكل رقم (٦) : احداثيات نقاط الانتقال



شكل رقم (٧): البنية الأساسية للبرنامج



شكل رقم (٨) : مخطط TPG

T25 = 300. 350. -10. 5.
MOVE T25

في السطر الاول.. يتم تعريف إحداثيات نقطة الانتقال T25.. وفى السطر الثانى يتم توجيه الروبوت للحركة حتى الوصول إلى هذه النقطة. يمكن أيضاً دمج سطرى البرنامج فى سطر واحد كالتالى:

DO T25 = 300. 350. -10. 5

– فى الطريقة الثالثة.. يتم تعريف نقاط الانتقال باستخدام علاقات جبرية. على سبيل المثال.. يمكن إضافة نقاط انتقال باستخدام العلاقة التالية:

T10 = 20. 30. 5. 0.
T20 = 30. 20. -5. 50.
T30 = T10 + T20

يمكن أيضاً إجراء عمليات طرح وضرب للحصول على نقاط انتقال كالتالى:

T10 = 20. 30. 5. 0.
T20 = 30. 20. -5. 50.
MOVE = T10 _ T20
Y = 2
MOVE = T10 x (-1) + T20 x Y

– فى الطريقة الاخيرة.. يتم الحصول على نقاط الانتقال من برنامج محاكاة كما هو الحال فى البرمجة خارج الخط (انظر العدد رقم ٨٧) أو من منظومة رؤية روبوتية أو من وحدة التحكم فى خلية العمل. تستخدم الثلاث طرق الأولى فى البرمجة الفورية.. حيث يجب وقف العلمية الانتاجية حتى الانتهاء من عملية التلقين او البرمجة.

٢– الاوامر البرمجية:

هى قائمة بالأوامر التى تتضمن تتابع حركة الروبوت بالاضافة إلى آليات اتخاذ القرار واكتساب البيانات والاتصال بالوحدات الانتاجية الأخرى فى بيئة العمل. يمكن تلخيص برنامج التحكم فى الروبوت الصناعى فى الخطوات التالية:

– البنية الأساسية للبرنامج:

كما هو مبين بالشكل رقم (٧).. يبدأ الروبوت من موضع ابتدائى يسمى HOME.. ثم يتم تنفيذ دورة العمل التى تتضمن العمليات التتابعية الواجب القيام بها لأداء المهمة المطلوبة كما هو مشروح سلفاً فى منظومة التعبئة – شكل رقم (٣).

– تحليل العملية الانتاجية:

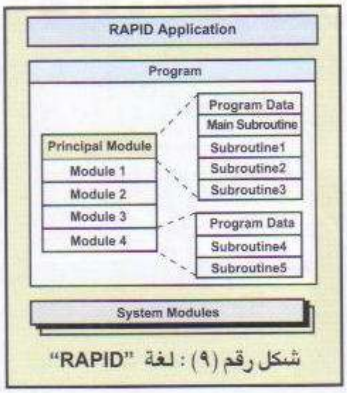
قبل الشروع فى كتابة البرنامج.. يجب فهم المشكلة او المهمة المراد القيام بها لتحديد تتابع العمليات التى يجب على الروبوت القيام بها.

– المهام والمهام الفرعية:

بعد الفهم الكامل للمشكلة يقوم المبرمج بتجزئة العمليات المطلوبة إلى مهام ومهام فرعية. على سبيل المثال.. فى خلية عمل تحتوى على روبوت صناعى يقوم بوضع أجزاء على سير ناقل لفحصها بواسطة منظومة رؤية.. ينقل الروبوت الأجزاء السليمة التى تجتاز اختبار الفحص إلى سير ناقل آخر. وفى حالة وجود أجزاء تالفة.. يتم وضعها فى صندوق إعادة التشغيل. تتلخص المهمة الأساسية للروبوت فى هذه المنظومة فى وضع الأجزاء على السير الناقل لفحصها بواسطة منظومة الرؤية. يمكن اعتبار نقل الأجزاء الصالحة إلى السير الناقل الثانى والتخلص من الأجزاء التالفة.. كمهام فرعية للروبوت يمكن برمجتها فى صورة برامج فرعية أو Subroutines.

– مخطط المهام والنقاط Task Point Graph (TPG):

يمكن اعتبار هذا المخطط كخريطة سريان توضح الحركات التتابعية المطلوبة لإنجاز المهمة المكلف بها الروبوت. يحتوى المخطط على بيانات كل نقاط الانتقال ومتغيرات الحركة عن كل نقطة والعمليات المنطقية الواجب إجراؤها خلال البرنامج. يبين الشكل رقم (٨) مخطط TPG لمنظومة الفحص السابقة.



شكل رقم (٩) : لغة "RAPID"

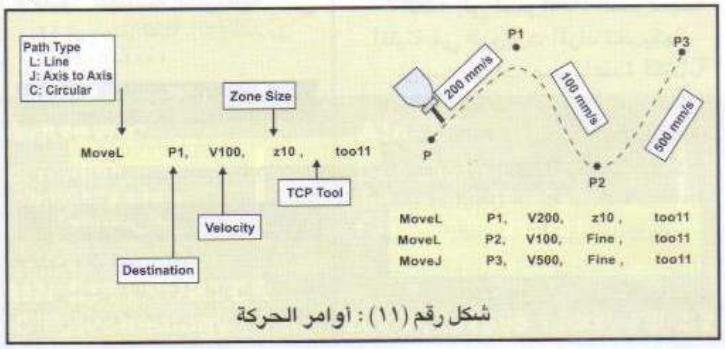
لغة البرمجة "ربيد"

تعتبر لغة «ربيد» Robotics Application Programming Interactive Dialogue (RAPID) التى ابتكرتها شركة ABB.. من لغات المستوى الثالث النصية متعددة الأغراض للروبوتات والتى توفر إمكانية القيام بالمهام الأربع المذكورة فى العدد السابق (المناولة والاستشعار والذكاء ومعالجة البيانات). ولا يقتصر استخدام هذه اللغة على برمجة روبوتات ABB.. ولكن يمكن استخدامها لبرمجة أنواع أخرى من الروبوتات. تتكون لغة «ربيد» من جزأين اساسيين هما وحدات برمجية خاصة بالبرنامج بالاضافة إلى وحدات أخرى خاصة بالمنظومة.. كما هو مبين بالشكل رقم (٩).

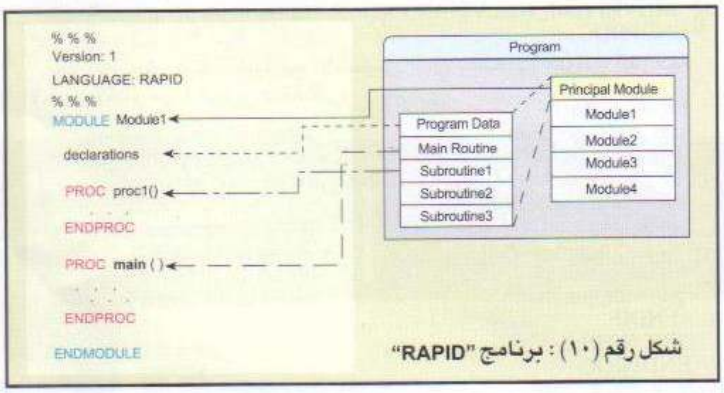
تعتبر كل وحدة برمجية في حد ذاتها كبرنامج يحتوى على عدة برامج فرعية بالإضافة إلى البيانات اللازمة لتنفيذ البرنامج. يوجد برنامج أساسى main وحيد كما هو الحال فى لغة C.. بالاضافة الى عدة برامج فرعية كما هو موضح بالشكل رقم (١٠).

تستخدم وحدات المنظومة لتعريف البيانات المشتركة والبرامج الفرعية التى توصف وحدات المنظومة.. مثل الأداة الطرفية أو وحدة التلقين أو خلية العمل.

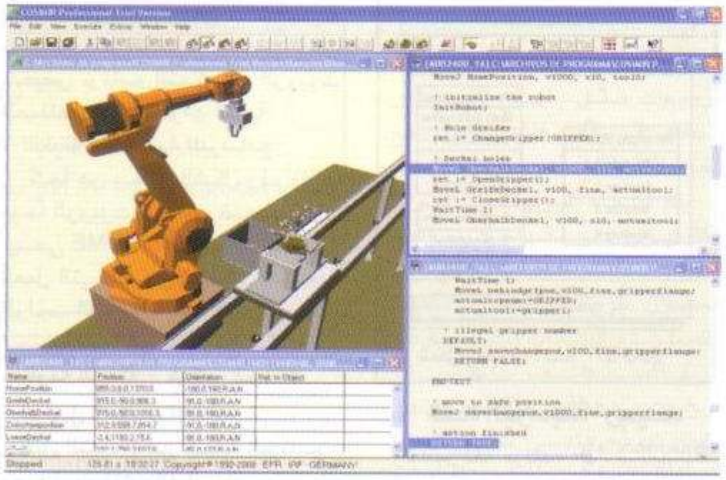
توفر لغة "ربيد" عدة أنواع من البيانات مثل.. الثوابت CONS و



شكل رقم (١١) : أوامر الحركة



شكل رقم (١٠) : برنامج "RAPID"



شكل رقم (١٢) : بيئة البرمجة "Festo COSIMIR"

المتغيرات VAR والمتغيرات التى يجب إعطاؤها قيمًا ابتدائية PERS. يمكن تعريف متغيرات أو ثوابت عديدة أو نصية باستخدام أنواع مختلفة مثل Num للقيم العددية و bool للقيم المنطقية (True أو False) و String للقيم النصية. يمكن أيضاً استخدام المسجلات Registers لتسجيل قيم الموضع أو الموضع وزاوية الدوران كالتالى:

```
VAR pos position1;
position1 := [500, 0, 940];
position1.x := position1.x + 50;
VAR pose pos1;
pos1 := [[500, 100, 800],
[1,0,0,0]];
pos1.trans := [650, -230,
1230];
pos1.trans y := -23.54;.
```

توفر لغة «ربيد» أيضاً إمكانية وصف الأداة الطرفية باستخدام المسجل tooldata كالتالى:

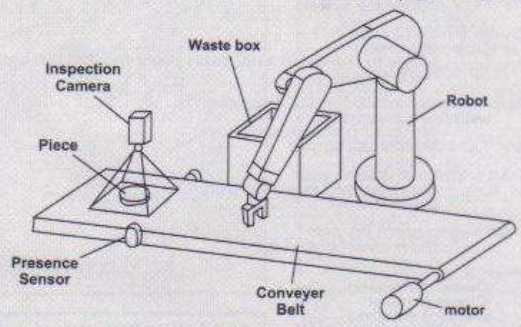
```
PERS tooldata gripper:=
[TRUE,[[97,0,220],
[0.924,0,0.383,0]],5,[-23,0,75],
[1,0,0,0],0,0,0]]
```

تعنى TRUE.. أن الماسك مثبت فى النهاية الطرفية للذراع الروبوتى وتتضمن المعاملات الأخرى وصفًا لموضع الماسك وكتلته ومركز الجاذبية وزوايا دوران محاور العزم حول مركز الجاذبية وعزم القصور الذاتى. توجد أيضاً عدة أوامر للتحكم فى حركة النهاية الطرفية للروبوت كما هو مبين بالشكل رقم (١١).. مثل MoveL للانتقال من نقطة إلى أخرى فى خط مستقيم.. و MoveC للانتقال فى مسار دائى.. و MoveJ للتحرك من نقطة إلى أخرى دون التقيد بمسار معين.. حيث يتم التحكم فى حركة المفاصل لاختيار أقصر مسار أو لتوفير الطاقة.

يحتوى أمر التحرك على أربعة معاملات يتحتم تعريفها. يمثل المعامل الأول P1 النقطة المراد الوصول إليها.. ويشير المعامل الثانى v100 إلى سرعة الحركة (ملليمتر\ثانية).. بينما يعبر المعامل الثالث z10 عن درجة الدقة أو السماحية (تمثل Fine أعلى دقة ممكنة).. ويشير المعامل الرابع Tool 1 إلى اسم الأداة الطرفية المثبتة فى الروبوت المراد تحريكها.

توجد أدوات مساعدة لكتابة البرامج بلغة "ربيد" من إنتاج شركة ABB مثل RAPID Syntax-Checker و Quick Teach. يمكن أيضاً استخدام بيئة برمجة متكاملة مثل.. COSIMIR لشركة Festo – شكل رقم (١٢)– أو RobotStudio لشركة ABB لاتمام عملية البرمجة والمحاكاة.

مثال على برمجة منظومة فحص



تحتوى منظومة الفحص الموضحة بالشكل على كاميرا فحص عينات وسير ناقل ومستشعر موضع.. بالاضافة إلى روبوت صناعى. فى هذه المنظومة.. يوقف الروبوت السير الناقل عن الحركة عند اكتشاف أى عيب فى العينة بفضل الإشارة القادمة من المستشعر.. ثم يقوم الروبوت بالتقاط العينة ووضعها فى صندوق التالف. بعد ذلك.. يعاود السير الناقل الحركة ويعود الروبوت إلى الوضع الابتدائى.

يبدأ البرنامج بتعريف المتغيرات والبيانات اللازمة لتنفيذ البرامج كالتالى:

```
ERS tooldata tool:=[FALSE,[[97,0,223],[0.924,0,0,0.383,0]],[5,[-
23,0,75],[1,0,0,0],0,0,0]]
PERS loaddata piece:=[5,[50,0,50],[1,0,0,0],0,0,0];
VAR robtarget conf_wait:=[[600,500,225],[1,0,0,0],[1,0,0,0],
[9E9,9E9,9E9,9E9,9E9,9E9]];
VAR signaldo gripper          !activation signal of the gripper
VAR signaldo activate_belt    !activation signal of the conveyer belt
VAR signaldi defected_piece   !signal of defected piece
VAR signaldi finish           !signal to end the program
```

بعد ذلك.. يتم كتابة بعض البرامج الفرعية لتنفيذ المهام التى يقوم بها الروبوت فى المنظومة.. مثل التحكم فى الماسك المثبت فى النهاية الطرفية للروبوت والتقاط العينة ووضعها فى صندوق التالف كالتالى:

```
PROC pick()
  Set gripper          !close gripper
    WaitTime 0.3       !Wait 0.3 seconds
    GripLoad piece     !Indicating that the piece is picked
ENDPROC

PROC place()
    Reset gripper      !Open gripper
    WaitTime 0.3       !Wait 0.3 seconds
    GripLoad LOAD0     !Indicating that there is no piece
ENDPROC

PROC pick_piece()
  MOVEJ *,VMAX,z60,tool    ! move the robot quickly to a certain
point
  MOVEL *,V500,z20,tool    !Move the robot in straight line
  MOVEL *,V150,FINE,tool   !go down with maximum resolution
  pick                     !pick the piece
  MOVEL *,V200,z20,tool    !go up with piece taken
ENDPROC

PROC place_piece()
  MOVEJ *,VMAX,z30,tool    !Move to the waste box
  MOVEJ *,V300,z30,tool
  place                    !place the piece
ENDPROC
```

يتم أيضاً كتابة برنامج فرعى لجعل الروبوت يعود إلى الموضع الابتدائى بعد انتهاء المهمة كالتالى:

```
PROC go_wait_position()
  MOVEJ conf_wait,VMAX,z30,tool  !Move to the initial position
ENDPROC
```

بعد ذلك.. يتم كتابة البرنامج الأساسى الذى يستخدم البرامج الفرعية الأخرى لاتمام مهمة الروبوت كالتالى:

```
PROC main()
  go_wait_position;                  !Move to initial position
  WHILE Dinput(finish)=0 Do          !wait end program signal
  IF Dinput(defected_piece)=1 THEN   !Wait defected piece signal
  SetDO activate_belt,0;             !Stop belt
  pick_piece                         !Pick the defected piece
  SetDO activate_belt,1;             !Activate the belt
    place_piece                      !Place the defected piece
  go_wait_position;                  !Move to the initial position
  ENDIF
 ENDWHILE
ENDPROC
```