

# DPL

## (P2: T7)

Yeray Méndez Romero

[yeray.mendez@udc.es](mailto:yeray.mendez@udc.es)

Daniel Rivera López

[d.rivera1@udc.es](mailto:d.rivera1@udc.es)

|  |    |
|--|----|
| 1-Introducción.....  | 3  |
| 2-Programación de robots .....   | 3  |
| 3-Relacion de los lenguajes de programación de robots y los habituales ..... | 5  |
| 4-Lenguajes en empresas.....   | 5  |
| 5-Lenguajes en educación .....   | 9  |
| 6-Relación de lenguajes en empresa y educación .....                         | 11 |
| 7-Bibliografía .....   | 12 |

## 1-Introducción

Robots son aquellos sistemas electromecánicos conducidos por un cierto programa que les permite realizar una determinada tarea de forma autónoma.

## 2-Programación de robots

Para ser capaces de definir el comportamiento de un robot existen dos niveles de programación, gestual y textual:

-Gestual: En la programación gestual el robot interviene en el proceso como un digitalizador de posiciones por las que luego pasará a ejecutar la tarea. No es necesaria experiencia por parte del usuario, ya que la definición de su comportamiento se realiza con métodos sencillos:

- \*Moviendo manualmente el robot a la posición deseada.
- \*Mediante un sistema maestro-esclavo por lo que los movimientos realizados por el operador en el maestro son seguidos por el esclavo.
- \*Por medio de un mando que permita definir y registrar posiciones, es la opción más usada.

-Textual: La secuencia de posiciones del robot es definida mediante un programa que define el movimiento del robot y su relación con el entorno (ya que puede obtener información de él). En este caso si es necesaria experiencia en técnicas de programación y realización de estrategias de movimiento basadas en información sensorial. Dentro de la programación textual se pueden distinguir tres tipos:

- \*Nivel robot: orientados a las operaciones y secuencia de ellas que el robot debe realizar para llevar a cabo la tarea (lectura de sensores y movimiento de actuadores).

Ejemplos de lenguajes:

-> ANORAD

-> EMILY

- \*Nivel objeto: las instrucciones se dan en función de los objetos a manejar, las instrucciones a nivel robot se conseguirán consultando una base de datos.

Ejemplos de lenguajes:

-> RAPT y AUTOPASS:

Utilizados para el ensamblaje de piezas, los objetos a utilizar se definían por un conjunto de sus características, empleaban un lenguaje sencillo con palabras inglesas, actualmente la información sobre ellos es muy limitada.

PLACE C<sub>1</sub>

SUCH THAT C<sub>1</sub> BOT CONTACTS C<sub>2</sub>TOP

AND B<sub>1</sub> A<sub>1</sub> IS ALIGNED WITH C<sub>2</sub>A<sub>1</sub>

AND B<sub>1</sub> A<sub>2</sub> IS ALIGNED WITH C<sub>2</sub>A<sub>2</sub>

*1- Superposición y alineación de objetos en Autopass*

\*Nivel tarea: el programador debe establecer cuáles son las acciones que debe ejecutar el robot, pero no tiene necesariamente que detallar cómo hacerlo. Es el sistema el que decide qué movimientos y comprobaciones sensoriales debe realizar, y en qué orden. Las decisiones se toman en función de:

-> Los objetivos propuestos.

-> El estado en cada momento del mundo del robot.

Ejemplo de lenguaje:

-> STRIPS:

Es un lenguaje formal utilizado por el planificador automático con el mismo nombre desarrollado en el SRI (Stanford Research Institute) por Richard Fikes y Nils Nilsson en 1971. STRIPS fue utilizado junto a Lisp por el robot Shakey que fue considerado el primer robot móvil de propósito general en poder razonar sobre sus propias acciones.

Una instancia de STRIPS está compuesta por un estado inicial, un conjunto de estados meta a alcanzar y una serie de acciones con sus precondiciones y postcondiciones.

```
Initial state: At(A), Level(low), BoxAt(C), BananasAt(B)
Goal state:   Have(bananas)
```

Actions:

```
// move from X to Y
_Move(X, Y)_
Preconditions: At(X), Level(low)
Postconditions: not At(X), At(Y)

// climb up on the box
_ClimbUp(Location)_
Preconditions: At(Location), BoxAt(Location), Level(low)
Postconditions: Level(high), not Level(low)

// climb down from the box
_ClimbDown(Location)_
Preconditions: At(Location), BoxAt(Location), Level(high)
Postconditions: Level(low), not Level(high)

// move monkey and box from X to Y
_MoveBox(X, Y)_
Preconditions: At(X), BoxAt(X), Level(low)
Postconditions: BoxAt(Y), not BoxAt(X), At(Y), not At(X)

// take the bananas
_TakeBananas(Location)_
Preconditions: At(Location), BananasAt(Location), Level(high)
Postconditions: Have(bananas)
```

*2- mono subiendo a por platanos con una caja en STRIPS*

### 3-Relacion de los lenguajes de programación de robots y los habituales

Podemos agrupar los lenguajes de programación de robots en dos tipos fundamentales:

-Extensiones de lenguajes habituales:

Se utilizan lenguajes habituales como C++, Java, Python, Matlab..., a los que se les añade la funcionalidad de programar robot mediante la importación de librerías específicas, por lo que son exactamente iguales a su lenguaje habitual y solo se diferencian en las funciones y tipos de datos que añadan las librerías.

-Lenguajes diseñados específicamente para programar robots:

Normalmente diseñados por una empresa fabricante de robots para utilizar en sus propias unidades, se diferencian de los lenguajes habituales en la posesión de funciones de movimiento y control de sensores, además de un menor número de tipos de datos, menos estructuras de control y menor complejidad del lenguaje en general, aunque estas últimas diferencias variarán dependiendo de la tarea que se vaya a programar, ya que deberemos elegir un lenguaje u otro, en un determinado robot.

En ambos tipos anteriores la elección del lenguaje dependerá de con cual es posible programar la tarea que deseemos, dependiendo al nivel que pertenezcan de los tres vistos previamente, y el lenguaje soportado por el robot a utilizar.

### 4-Lenguajes en empresas

En el mundo empresarial podemos encontrar muchos lenguajes.

Robotics Business Review (RBR) es la empresa encargada de elaborar la lista anual RBR50 en la que se encuentran las empresas más innovadoras, influyentes y exitosas comercialmente en el ámbito de la robótica, en la lista de este año 2017 podemos encontrar:

-La empresa ABB con el lenguaje RAPID

ABB utilizaba ARLA desde que lo introdujo en 1981 junto con la controladora S2 y posteriormente se utilizó también para la S3 de la compañía, pero en 1994 desarrolló RAPID junto con su nueva controladora S4, este último influenciado por su predecesor ARLA, que dejó de utilizarse tras la introducción de RAPID, y C.

RAPID es un lenguaje de programación textual de alto nivel utilizado actualmente en la controladora IRC5 de ABB. Una aplicación RAPID consta de un programa y una serie de módulos del sistema. El programa es una secuencia de instrucciones que controlan el robot y en general consta de tres partes:

- Una rutina principal (main): Rutina donde se inicia la ejecución.
- Datos de programa: Definen posiciones, valores numéricos, sistemas de coordenadas, etc.
- Subrutinas: Sirven para dividir el programa en partes más pequeñas a fin de obtener un programa modular.

```

%%
VERSION:1
LANGUAGE:ENGLISH
%%
MODULE EJEMPLO
  CONST robtarg A:=[[0,0,0],[0,0,0,0],[0,-1,0,0], [9E+09,...]];

  CONST tooldata pinza:= [TRUE, [[0,0,0],[1,0,0,0]],
                             [0,[0,0,0],[1,0,0,0],0,0,0]];

  PROC cerrar_pinza()
    Set spinza;
  ENDPROC

  PROC coger_pieza()
    MoveJ B1,v100,z5,pinza;
    MoveL B,v80,fine,pinza;
    cerrar_pinza;
  ENDPROC

  PROC main()
    CONST dionum listo:=1;
    abrir_pinza;
    WHILE TRUE DO
      MoveJ A,v100,fine,pinza;
      WaitDI econtrol,listo;
      coger_pieza;
      MoveL B1,v80,z5,pinza;
      MoveJ D,v100,z100,pinza;
      MoveJ C1,v100,z5,pinza;
      MoveL C,v80,fine,pinza;
      abrir_pinza;
      MoveL C1,v80,z5,pinza;
    ENDWHILE
  ENDPROC
ENDMODULE

```

3- Mover una pieza de sitio en RAPID

-La empresa Fanuc con Karel

KAREL es un lenguaje que combina características de alto nivel y bajo nivel, es similar a Pascal. Cuenta con variables fuertemente tipadas, constantes, tipos personalizados, procedimientos, funciones de movimiento, movimiento concurrente etc. KAREL es un lenguaje compilado, el fichero.kl debe traducirse a un fichero.pc antes de cargarse y ejecutarse en el controlador. La carga de ficheros “.pc” a los robots no suele realizarse, y en su lugar los programas se crean con su interfaz propietaria, debido a que activar esta opción en sus controladoras acarrea un desembolso económico.

```

-- hello.kl
PROGRAM hello
  -- this is a comment

  -- translator directive
  %COMMENT = 'Hello'

  -- const declarations
  CONST
    LANG_EN = 1
    LANG_ES = 2
    LANG_FR = 3

  -- custom type
  TYPE
    person_t = STRUCTURE
      name      : STRING[16]
      lang_id   : INTEGER
    ENDSTRUCTURE

  -- program vars
  VAR
    people : ARRAY[3] OF person_t

  -- custom routines
  -- Returns the proper greeting for a given language
  ROUTINE greeting(lang_id : INTEGER) : STRING
  BEGIN
    SELECT lang_id OF
      CASE (LANG_EN):
        RETURN('Hello')
      CASE (LANG_ES):
        RETURN('Hola')
      CASE (LANG_FR):
        RETURN('Bonjour')
    ENDSELECT
  END greeting

  -- Greets a person in their language
  --
  -- Example:
  --   person.name = 'John'
  --   person.lang_id = LANG_EN
  --
  --   greet(person)
  --   # => Hello, John
  ROUTINE greet(person : person_t)
  BEGIN
    WRITE(greeting(person.lang_id), ', ', person.name, CR)
  END greet
BEGIN
  -- setup people[] array
  -- notice KAREL arrays are 1-based, not 0-based.
  people[1].name = 'John'
  people[1].lang_id = LANG_EN

  people[2].name = 'Jose'
  people[2].lang_id = LANG_ES

  people[3].name = 'Jaques'
  people[3].lang_id = LANG_FR

  -- greet each person
  greet(people[1])
  greet(people[2])
  greet(people[3])
END hello

  -- you could also put your routines here

```

4- Programa para saludar a gente en su idioma en Karel

-ADEPT con V+:

La historia de V+ comienza sobre 1959, por aquella época era conocido como VAL (Victor's Assembly Language) y propietario de la empresa Unimation que fue la primera fabricante de robótica del mundo, VAL fue evolucionando en VAL-II y VAL-3 a lo largo del tiempo, en 1983 Adept fue fundada por parte de la división de Unimation accediendo así a los derechos sobre VAL.

V+ es un lenguaje de programación interpretado y estructurado de alto nivel que proporciona inteligibilidad, fiabilidad, adaptabilidad y transportabilidad. En V+ cada línea se interpreta como una instrucción del programa y las diferentes tareas poseen prioridades asignables.

Algoritmo:

```
PROGRAM cogergarra(garra, salir)
LEFTY ; comportamiento de brazo izquierdo
APPRO garra, 100 ; mueve la base a 100mm encima de la garra
DELAY(0.5) ; retardo de 0.5 segundos
OPENI ; abre los enganches de la base
DELAY(0.5)
MOVES garra ; mueve la base hasta entrar en la garra
DELAY(0.5)
CLOSEI ; cierra los enganches de la garra
DELAY(0.5)
MOVES salir ; saca la garra de su plataforma
DELAY(0.5)
DEPARTS 100 ; sube la garra 100mm
```

5- Programa para coger una garra en V+

**\*\*Nota: Los lenguajes anteriormente vistos pertenecen al nivel robot\*\***

La mayor parte de las empresas tienen su propio lenguaje de programación de robots, teniendo en cuenta solo sus necesidades en cuanto a software, y protocolos de comunicación diversos, lo que provoca que el usuario normalmente se quede bloqueado a los productos de un fabricante, ya que el código no es portable debido a los lenguajes propietarios, los fabricantes solo soportan una serie de sensores y periféricos limitados, esto unido a las interfaces mecánicas no estandarizadas acaba implicando una pérdida de tiempo y dinero para el usuario final.

Ante esta gran cantidad de lenguajes propietarios han surgido proyectos intentando proporcionar un lenguaje estándar, como ROS-Industrial, el cual es de open-source y extiende las capacidades de ROS, Robot Operating System, que es un conjunto de herramientas y librerías para simplificar la tarea de crear un comportamiento complejo y robusto en una gran variedad de robots, con ROS-I ROS adquiere capacidades para la fabricación industrial. ROS-I empezó gracias a la colaboración de Yaskawa Motoman Robotics, Southwest Research Institute, y Willow Garage en el apoyo a utilizar ROS en la producción industrial. El repositorio de software en GitHub fue fundado en enero de 2012 por Shaun Edwards (SwRI). ROS-I posee varios consorcios (ROS-Industrial Consortium Americas, ROS-I Consortium Europe, ROS-Industrial Consortium Asia Pacific), cuya función es apoyar a la comunidad de ROS-I proporcionando entrenamiento, soporte técnico y definiendo la hoja de ruta en el desarrollo de ROS-I.



Los consorcios también favorecen la creación de proyectos empresariales conjuntos para ampliar las capacidades de ROS-I.

ROS-I también tiene eventos como ROSCon y conferencias, de las cuales hay una prevista para el 12 de diciembre de 2017.

Los miembros de consorcios de ROS-I a fecha de octubre de 2017 son:



## 5-Lenguajes en educación

Actualmente los robots se han convertido en una forma educativa de interesar a la juventud en el mundo de la programación, la inteligencia artificial y la robótica.

A diferencia del mundo empresarial, las compañías dedicadas a educación no suelen tener un lenguaje propietario, sino que suelen permitir la programación de sus robots en varios lenguajes habituales, algunas también poseen software propietario para la programación de sus robots, el cual consiste normalmente en una interfaz de programación más visual, pero el programa final seguirá estando en un lenguaje estándar no propietario.

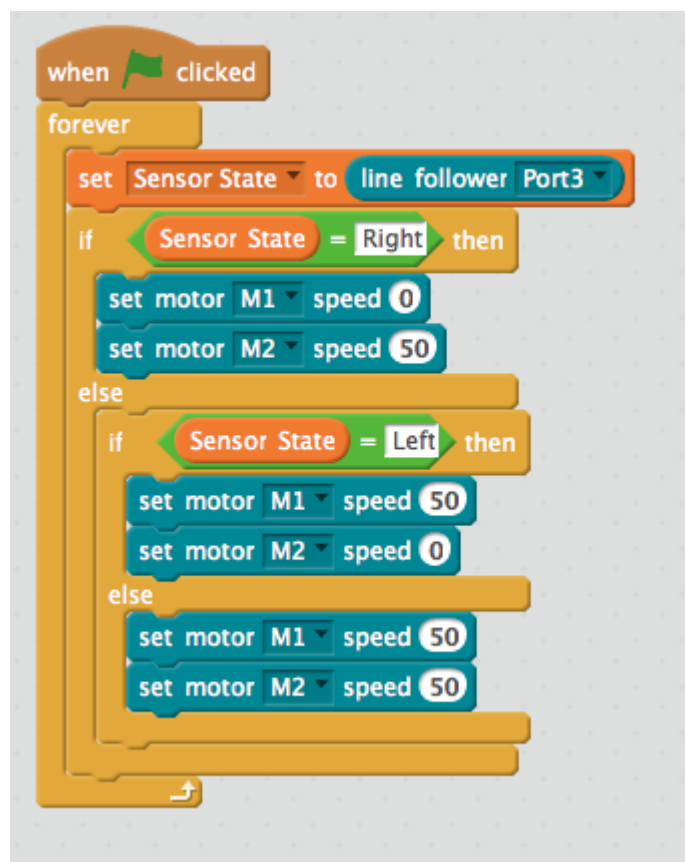
Algunos lenguajes en este ámbito:

- Scratch:

Es un lenguaje de programación visual (arrastrar y soltar bloques) y educativo desarrollado por Lifelong Kindergarten Group en el MIT (Massachusetts Institute of Technology), se clasifica dentro de los paradigmas imperativo y dirigido por eventos, su primera versión fue lanzada en 2003 con el propósito de ayudar a la gente joven a aprender a programar, su segunda versión fue lanzada en mayo de 2013.

Scratch goza de una amplia comunidad con unos 15 millones de usuarios registrados en su página oficial, es utilizado en escuelas, bibliotecas, museos, clases de introducción a la informática (incluyendo a Harvard) además posee su propio día (21 de mayo).

Como ejemplos de empresas que utilizan Scratch tenemos RoboThink, PHIRO..., mientras que otras emplean lenguajes que utilizan la programación visual de Scratch para generar código en otro lenguaje, como es el caso de Scratch4Arduino utilizado por ProtoCREA y mblock utilizado por Makeblock, por último también podemos encontrar otros como Snap! Basados en Scratch y que añaden algunas funciones.



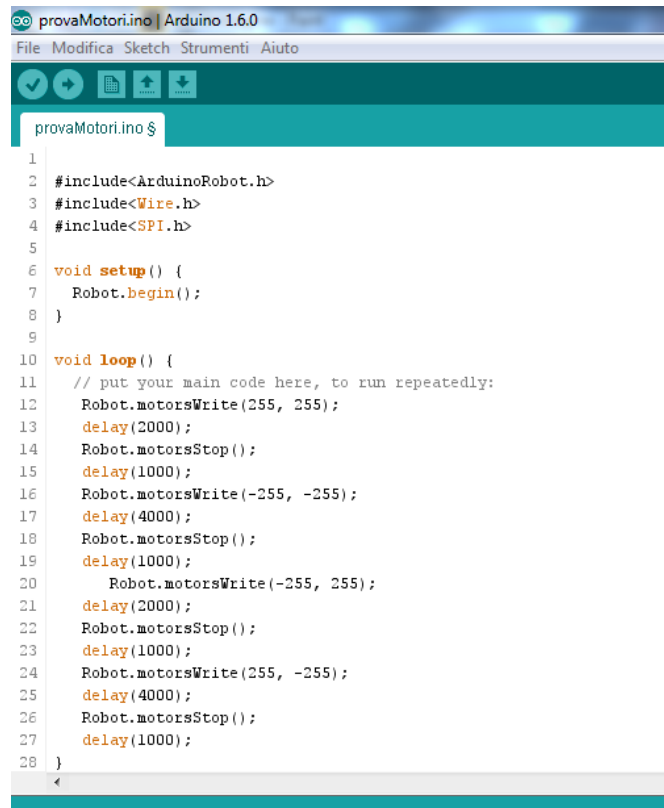
6- Programa para seguir líneas en Scratch

-Arduino:

Arduino es una plataforma electrónica open-source basada en hardware y software sencilla de usar.

El lenguaje Arduino es simplemente el lenguaje C/C++ con librerías añadidas, sirve para programar los diversos modelos de placas Arduino y aunque puede emplearse directamente en la programación

orientada a robots en educación, es habitual ver interfaces visuales que simplifican la programación a este lenguaje, como ya anteriormente se citaron Scratch4Arduino y mblock así como Ardublock entre otros.

The image shows the Arduino IDE interface with a file named 'provaMotori.ino' open. The code is written in C++ and includes libraries for an Arduino robot. The 'setup' function calls 'Robot.begin()'. The 'loop' function contains a sequence of motor control commands: writing values to 'Robot.motorsWrite', followed by 'delay' calls, and 'Robot.motorsStop()' calls. The sequence alternates between positive and negative motor values.

```
1
2 #include<ArduinoRobot.h>
3 #include<Wire.h>
4 #include<SPI.h>
5
6 void setup() {
7   Robot.begin();
8 }
9
10 void loop() {
11   // put your main code here, to run repeatedly:
12   Robot.motorsWrite(255, 255);
13   delay(2000);
14   Robot.motorsStop();
15   delay(1000);
16   Robot.motorsWrite(-255, -255);
17   delay(4000);
18   Robot.motorsStop();
19   delay(1000);
20   Robot.motorsWrite(-255, 255);
21   delay(2000);
22   Robot.motorsStop();
23   delay(1000);
24   Robot.motorsWrite(255, -255);
25   delay(4000);
26   Robot.motorsStop();
27   delay(1000);
28 }
```

7- Programa de prueba de motores en Arduino

Además de los dos ejemplos anteriores, elegidos por su soporte (o de alguna variante) en una gran cantidad de robots educativos, la robótica en el ámbito educativo emplea multitud de lenguajes habituales junto con módulos o librerías que les confieren la capacidad de programar robots, entre los que podemos encontrar Python, C/C++, JavaScript, Matlab...

## 6-Relación de lenguajes en empresa y educación

A pesar de la diferencia de lenguajes entre el mundo empresarial y el educativo, no existe un cambio en la variedad de tareas a realizar en cada uno de ellos, las únicas posibles diferencias serían la eficiencia del código, ya que los lenguajes de empresa trabajan sobre un hardware específico con solo las funciones necesarias y la simplicidad del mismo la cual es más habitual en la empresa ya que las funciones realizables por un robot contarán ya con una función específica.

## 7-Bibliografía

<https://es.wikipedia.org/wiki/Robot>

<https://analisisyprogramacionoop.blogspot.com.es/2014/10/programacion-de-robots.html>

[http://platea.pntic.mec.es/vgonzale/cyr\\_0708/archivos/\\_15/Tema\\_5.6.htm](http://platea.pntic.mec.es/vgonzale/cyr_0708/archivos/_15/Tema_5.6.htm)

<https://blog.robotiq.com/what-is-the-best-programming-language-for-robotics>

[https://en.wikipedia.org/wiki/Educational\\_robotics](https://en.wikipedia.org/wiki/Educational_robotics)

<http://petercorke.com/wordpress/toolboxes/robotics-toolbox>

<https://www.makeblock.es/soporte/mblock/>

<http://codigo21.educacion.navarra.es/autoaprendizaje/primeros-pasos-con-scratch-y-lego-wedo/>

<http://www.onerobotics.com/posts/2013/introduction-to-karel-programming/>

<http://www.monografias.com/trabajos3/progrob/progrob.shtml>

<http://rosindustrial.org/>

[https://www.roboticsbusinessreview.com/companies/?companyType=rbr\\_50](https://www.roboticsbusinessreview.com/companies/?companyType=rbr_50)

<https://en.wikipedia.org/wiki/RAPID>

<http://hopl.info/showlanguage.prx?exp=7482>

[https://wikivisually.com/lang-de/wiki/ARLA\\_\(Programmiersprache\)](https://wikivisually.com/lang-de/wiki/ARLA_(Programmiersprache))

<http://www.robot-forum.com/robotforum/fanuc-robot-forum/karel-load-option-in-r30ib/>

[https://en.wikipedia.org/wiki/Adept\\_Technology](https://en.wikipedia.org/wiki/Adept_Technology)

[https://es.wikipedia.org/wiki/Scratch\\_\(lenguaje\\_de\\_programaci%C3%B3n\)](https://es.wikipedia.org/wiki/Scratch_(lenguaje_de_programaci%C3%B3n))

[https://wiki.scratch.mit.edu/wiki/Scratch#Notable\\_Information](https://wiki.scratch.mit.edu/wiki/Scratch#Notable_Information)

[https://en.wikipedia.org/wiki/Educational\\_robotics](https://en.wikipedia.org/wiki/Educational_robotics)

<https://www.arduino.cc/en/Guide/Introduction>

[https://en.wikipedia.org/wiki/Shakey\\_the\\_robot](https://en.wikipedia.org/wiki/Shakey_the_robot)

<https://es.wikipedia.org/wiki/STRIPS>