Robótica – Año 2016

**Desarrollo de un compilador y simulación para el control**

**de un robot guiado diferencial**

Hernán González, Rodrigo Menéndez Marichelar

*Universidad Tecnológica Nacional, Facultad Regional Buenos Aires*

***Abstract* — El presente trabajo describe el desarrollo de un compilador para el control y simulación de un robot guiado diferencial. El mismo genera código para ser utilizado como input de DuinoBot y desarrollar la trayectoria de un ocho en un robot Múltiplo N6. Asimismo, genera código para ser utilizado como input en Matlab para su posterior simulación del robot, utilizando los resultados obtenidos en el estudio cinemático de la estructura.**

***Index Terms* — N6 Múltiplo, Compilador, Antlr, Matlab, guiado diferencial, cinemática directa, Dudek & Jenkin , Robótica.**

1. **Introducción**

En la robótica es una práctica habitual utilizar técnicas que permiten generar instrucciones de alto nivel que son comprensibles por el operador del robot y permiten que el mismo ingrese los parámetros requeridos para una función particular: tipo de movimiento, momento de ejecución, duración del estímulo, ángulo, velocidad, posición, etc.

Dichas funciones requieren ser traducidas a un código que pueda ser comprendido por un procesador (código de máquina). Esto se logra mediante la utilización de un compilador, el cual recibe las instrucciones por parte del operador, las cuales son implementadas en un pseudocódigo de alto nivel comprensible para el programador del robot. A partir de este input, el compilador genera el código comprensible para el procesador utilizando distintas herramientas de análisis y aplicando diferentes reglas propias del robot. Como parte del presente trabajo, el compilador generará dos

Manuscrito elaborado en Septiembre 2016. Dicho trabajo fue apoyado en parte por la Universidad Tecnológica Nacional (UTN), Facultad Regional Buenos Aires (FRBA) bajo la cátedra de Robótica, carrera de Ingeniería Electrónica.

H. G. Estudiante Ingeniería Electrónica FRBA

(e-mail: [hernangonzalez07@gmail.com](mailto:hernangonzalez07@gmail.com)).

R. M. Estudiante Ingeniería Electrónica FRBA

(e-mail: [rodrigo2mzm@gmail.com](mailto:rodrigo2mzm@gmail.com)).

tipos de outputs diferentes, las cuales serán usadas como input del programa desarrollado en C para el robot Multiplo N6 basado en Arduino y del script de simulación de la cinemática directa del robot generado en Matlab.

1. **Compilación**

Para llevar adelante los puntos expresados en la sección anterior, en primer lugar, el compilador deberá verificar que el pseudocódigo definido por el usuario sea válido. Esto lo logra a partir de reglas predefinidas para luego convertir el set de instrucciones de entrada en el correspondiente set de instrucciones de salida (en nuestro caso las entradas para el código de movimiento del robot y para su simulación en Matlab).

El proceso se realiza siguiendo diferentes técnicas:

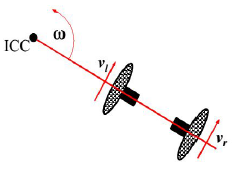
1. **Análisis léxico:** es la técnica mediante la cual se convierte el código fuente en una serie de *tokens*. Se reconocen las palabras reservadas del lenguaje y se evalúan los formatos de los tipos de datos y variables utilizadas en base a las reglas léxicas.
2. **Análisis sintáctico:** los tokens generados previamente son procesados siguiendo a las reglas sintácticas. Las mismas definen el orden de los tokens, lo cual se traduce en las instrucciones del lenguaje.
3. **Generación del código objeto:** una vez verificada la sintaxis, se convierte el set de instrucciones de entrada al set de instrucciones del código de salida. En este punto es dónde se aplica la lógica al compilador, y se establecen por ejemplo límites de diseño basados en el modelo del robot.

Para realizar el compilador, en el presente trabajo se utilizó el IDE ANTLRWorks 1.4.2 en su versión para el sistema operativo Linux. Este IDE combina un editor de gramática con un intérprete, lo que permite fácilmente corregir errores de gramática y contiene tanto el parser como el lexer por lo que no es necesario el uso de otras herramientas.

1. **Cinemática Directa**

Para el análisis del robot de guiado diferencial se utiliza el método de Dudek & Jenkin. Con dicho método se busca determinar la posición del robot, al cual se le ingresan velocidades, respondiendo a un movimiento circular con un determinado radio y punto central.

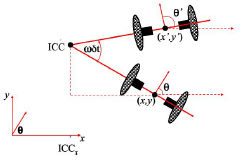
Cómo se ha estudiado en el Trabajo Practico 1 de la presente materia, este modelo considera que las ruedas están girando alrededor de un punto central ICC, ambas sobre un eje en común como se muestra en la Fig. [1]. Luego, variando las velocidades de las ruedas, se tendrán diferentes trayectorias.



**Figura 1: Giro de ruedas alrededor de punto ICC.**

Como se ha estudiado, a partir del radio de giro y la velocidad angular del sistema es posible resolver el problema de cinemática directa. Entónces, si el robot gira con un determinado por un δt alrededor de ICC, se obtiene:

Esto se muestra en la Fig. [2].



**Figura 2: Aumento del ángulo de giro.**

El centro de rotación es

Ahora, con una posición de partida, y con una rotación 2D, se puede calcular la nueva posición que adquiere el robot

Por lo que así se puede conocer la posición y el sentido.

1. **Prueba de código generado**

Para simular el funcionamiento esperado, se implementa una rutina que consiste en generar una instrucción para el robot mediante la cual el mismo se desplazará en forma de ocho. Esta rutina es detallada por el usuario en forma de una instrucción, la cual es interpretada por el compilador para generar la entrada al script que permite implementar dicho movimiento.

En base a las conclusiones obtenidas en el trabajo práctico 1, la instrucción desarrollada para implementar dicho movimiento se detalla a continuación:

**Mover\_ocho(vel\_R,vel\_L,tiempo\_1,tiempo\_2)**

La instrucción traslada el extremo del robot desde su posición inicial realizando una trayectoria con las velocidades ingresadas para cada una de las ruedas durante el tiempo\_1 y luego una trayectoria alternando las velocidades ingresadas para cada una de las ruedas durante un tiempo que resulta de tiempo\_2-tiempo\_1.

Descripción de parámetros:

* Vel\_R: velocidad rueda derecha en valor porcentual. Admite valores entre 0 y 100.
* Vel\_L: velocidad rueda izquierda en valor porcentual. Admite valores entre 0 y 100.
* Tiempo\_1: tiempo durante el cual el robot realizará un movimiento con las velocidades ingresadas en la instrucción. Admite valores de 0 a 30 y equivale a múltiplos de 500ms.
* Tiempo\_2: tiempo en el cual el robot detendrá su movimiento, considerando que inmediatamente después de transcurrido tiempo\_1 el robot invertirá las velocidades ingresadas para rueda derecha y rueda izquierda. Admite valores de 0 a 30 y equivale a múltiplos de 500ms.

En este caso el código de prueba fue el siguiente

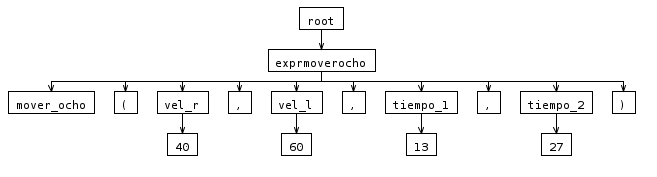
Mover\_ocho( 40 , 60 , 13 , 27 )

Esto indica que el primer tramo de la trayectoria lo hará con velocidades relativas de 40% y 60% de la velocidad nominal en rueda derecha y rueda izquierda respectivamente, y que girará en el primer tramo durante 6.5s y en el segundo tramo durante 7s.





**Figura 3: parser rule**

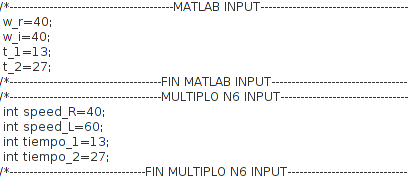


**Figura 3: parser tree**

Para más información, por favor ver Anexo I.

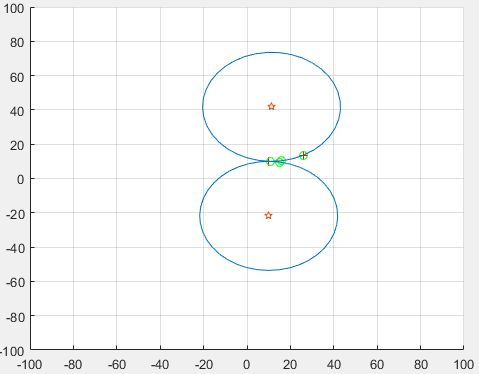
1. **Simulación y código de movimiento**

Para poder visualizar los resultados generados por el compilador se programó el mismo para que genere dos tipos de salidas diferentes las cuales funcionan como input tanto del código en C generado en Duinobot para el movimiento del robot Multiplo N6 y como input del script de Matlab que calcula la cinemática directa en base a los valores de velocidades y tiempo de movimiento en cada una de las ruedas.



**Figura 4: salida del compilador**

Los valores que se obtienen a la salida son ingresados como parámetro tanto a Matlab como al código .C desarrollado para realizar el movimiento de ocho en el robot múltiplo y se obtienen las conclusiones correspondientes a la correlación entre la simulación de la cinemática directa del robot realizada en Matlab y el movimiento real del mismo.



**Figura 5: simulación en Matlab para trayectoria de ocho con velocidades 40% y 60% y tiempo 6,5 segundos en el primer tramo y 7 segundos en el segundo tramo**

1. **Conclusiones**

A lo largo del presente trabajo se pudieron asociar los diferentes conocimientos obtenidos durante la cursada acerca de robótica y aplicarlos al diseño de un compilador. Se puede concluir que el uso de compiladores es una herramienta con gran potencia ya que permite traducir acciones complejas en instrucciones simples para un usuario que puede no contar con conocimientos profundos acerca del funcionamiento de un robot o de su estructura.

Asimismo, se destaca que este trabajo reúne de forma integral los conocimientos utilizados durante los trabajos previos ya que permite asociar la salida del compilador con los desarrollos realizados tanto con duinobot para generar el movimiento del robot diferencial y los realizados utilizando Matlab para implementar el cálculo de la cinemática directa de un robot y su simulación a partir de las variables de entrada. Esto permite abarcar desde diferentes ángulos los aspectos de la robótica y realizar comparaciones que permiten extraer conclusiones interesantes acerca del comportamiento esperado y real del robot.

Al finalizar el trabajo el alumno cuenta con un conocimiento amplio acerca de robótica que le permite desarrollarse en la temática y que abarca diferentes áreas de aplicación como ser, análisis cinemático y dinámico, desarrollo en Arduino, desarrollo en Solidworks, desarrollo de compiladores y vasta experiencia en el uso de Matlab.

1. **Referencias**

[1] Gregory Dudek and Michael Jenkin*, Computational Principles of mobile Robotics SECOND EDITION*, 2010

[2] Barrientos, Peñin, Balaguer y Aracil, “Fundamentos de

Robótica”, Mc Graw Hill, 2001

[3] Hernan Gianetta, Robótica, Topico: “Unidad Temática 01 –

Cinemática del robot”, Universidad Tecnológica

Nacional – Facultad Regional Buenos Aires, Buenos Aires, 14

de Marzo, 2014.

[4] ANTLR, Terrence Parr, University of San Francisco.

[5] Hernan Gianetta, Robótica, Topico: “Unidad Temática 07

Compiladores y lenguaje de programación”, Universidad

Tecnológica Nacional – Facultad Regional

**ANEXO I**

***Código compilador***

grammar multiploN6;

tokens

{

MOVE = 'mover\_ocho';

}

/\*PARSER RULES\*/

exprmoverocho : MOVE '(' vel\_r COMA vel\_l COMA tiempo\_1 COMA tiempo\_2 ')'

{

/\*Chequeo si es correcto el rango\*/

boolean check\_velocidades\_tiempo\_OK = false;

if((0<$vel\_r.vel)&&($vel\_r.vel<=100)&&(0<$vel\_l.vel)&&($vel\_l.vel<=100)&&(0<$tiempo\_1.time)&&($tiempo\_1.time<30)&&(0<$tiempo\_2.time)&&($tiempo\_2.time<30))

{

check\_velocidades\_tiempo\_OK = true;

}

if(check\_velocidades\_tiempo\_OK)

{

System.out.println("/\*-----------------------------------------MATLAB INPUT--------------------------------------------\*/");

System.out.println(" w\_r="+$vel\_r.vel+";");

System.out.println(" w\_i="+$vel\_l.vel+";");

System.out.println(" t\_1="+$tiempo\_1.time+";");

System.out.println(" t\_2="+$tiempo\_2.time+";");

System.out.println("/\*--------------------------------------FIN MATLAB INPUT--------------------------------------------\*/");

System.out.println("/\*--------------------------------------MULTIPLO N6 INPUT--------------------------------------------\*/");

System.out.println(" int speed\_R="+$vel\_r.vel+";");

System.out.println(" int speed\_L="+$vel\_l.vel+";");

System.out.println(" int tiempo\_1="+$tiempo\_1.time+";");

System.out.println(" int tiempo\_2="+$tiempo\_2.time+";");

System.out.println("/\*----------------------------------FIN MULTIPLO N6 INPUT--------------------------------------------\*/");

}else

{

System.out.println("/\*Error: alguna de las variables ingresadas esta fuera de los rangos permitidos\*/");

}

};

vel\_r returns [int vel]:INT{$vel = Integer.parseInt($INT.text);};

vel\_l returns [int vel]:INT{$vel = Integer.parseInt($INT.text);};

tiempo\_1 returns [int time]:INT{$time = Integer.parseInt($INT.text);};

tiempo\_2 returns [int time]:INT{$time = Integer.parseInt($INT.text);};

/\*LEXER RULES\*/

ID : ('a'..'z'|'A'..'Z'|'\_') ('a'..'z'|'A'..'Z'|'0'..'9'|'\_')

;

INT : '0'..'9'+;

COMA : ',' ;

LEFT\_PARENT

: '('

;

RIGHT\_PARENT

: ')'

;

COMMENT

: '//' ~('\n'|'\r')\* '\r'? '\n' {$channel=HIDDEN;}

| '/\*' ( options {greedy=false;} : . )\* '\*/' {$channel=HIDDEN;}

;

WS : ( ' '

| '\t'

| '\r'

| '\n'

) {$channel=HIDDEN;}

;

**ANEXO II**

***Código DuinoBot***

#include <DCMotor.h>

#include <servo.h>

#include <TimerOne.h>

#include <EEPROM.h>

DCMotor motor0(M0\_EN,M0\_D0,M0\_D1);

DCMotor motor1(M1\_EN,M1\_D0,M1\_D1);

int giro=0;

int contador\_encoder\_R=0;

int contador\_encoder\_L=0;

int distancia\_L=0;

int distancia\_R=0;

int direccion\_memoria=0;

int segundos=0;

int sensor\_encoder\_R=0;

int sensor\_encoder\_L=0;

int pasos\_L1=0, pasos\_R1=0, pasos\_L2=0, pasos\_R2=0, check\_R=0, check\_L=0;

//Comienzo Input del compilador

int speed\_R=60;

int speed\_L=40;

int tiempo\_1=13;

int tiempo\_2=27; // tiempos en fracciones de 500ms

//Fin input compilador

void setup()

{

float paso=0, pasos\_tot=16, perimetro=0, pi=3.14, radio\_rueda=0, vueltas\_R=0, vueltas\_L=0,speed\_100=200;

radio\_rueda=3;

perimetro=pi\*radio\_rueda\*2;

paso=perimetro/pasos\_tot;

// encoder\_R=A0;

// encoder\_L=A1;

Serial.begin(9600);

delay (5000);

Serial.println("Pasos\_last\_run\_R1: ");

Serial.println(EEPROMReadInt(0));

Serial.println("Pasos\_last\_run\_L1: ");

Serial.println(EEPROMReadInt(2));

Serial.println("Pasos\_last\_run\_R2: ");

Serial.println(EEPROMReadInt(4));

Serial.println("Pasos\_last\_run\_L2: ");

Serial.println(EEPROMReadInt(6));

// delay(5000);

Timer1.initialize(500000);

Timer1.attachInterrupt(timerISR);

}

void loop()

{

if (giro == 0) //Acaba de arrancar: es el primer tramo

{

motor0.setSpeed(-speed\_R);

motor1.setSpeed(speed\_L);

sensor\_encoder\_R = analogRead(A0);

sensor\_encoder\_L = analogRead(A1);

// Serial.println("sensor\_actual\_R: ");

// Serial.println(sensor\_encoder\_R);

// Serial.println("sensor\_actual\_L: ");

// Serial.println(sensor\_encoder\_L);

if (sensor\_encoder\_R >240)

{

if (check\_R==0)

{

pasos\_R1++;

check\_R=1;

}

}

else

{

if (check\_R==1) check\_R=0;

}

if (sensor\_encoder\_L >240)

{

if (check\_L==0)

{

pasos\_L1++;

check\_L=1;

}

}

else

{

if (check\_L==1) check\_L=0;

}

}

else

{

if (giro==1) //Está en el segundo tramo

{

motor0.setSpeed(-speed\_L);

motor1.setSpeed(speed\_R);

int sensor\_encoder\_R = analogRead(A0);

int sensor\_encoder\_L = analogRead(A1);

if (sensor\_encoder\_R >240)

{

if (check\_R==0)

{

pasos\_R2++;

check\_R=1;

}

}

else

{

if (check\_R==1) check\_R=0;

}

if (sensor\_encoder\_L >240)

{

if (check\_L==0)

{

pasos\_L2++;

check\_L=1;

}

}

else

{

if (check\_L==1) check\_L=0;

}

}

else

{

if (giro==2) //Detener y loguear

{

motor0.brake();

motor1.brake();

EEPROMWriteInt(0, pasos\_R1);

EEPROMWriteInt(2, pasos\_L1);

EEPROMWriteInt(4, pasos\_R2);

EEPROMWriteInt(6, pasos\_L2);

// Serial.println("Pasos\_actual\_R1: ");

// Serial.println(pasos\_R1);

// Serial.println("Pasos\_actual\_L1: ");

// Serial.println(pasos\_L1);

//

// Serial.println("Pasos\_actual\_R2: ");

// Serial.println(pasos\_R2);

// Serial.println("Pasos\_actual\_L2: ");

// Serial.println(pasos\_L2);

Serial.println("Pasos\_actual\_saved\_R1: ");

Serial.println(EEPROMReadInt(0));

Serial.println("Pasos\_actual\_saved\_L1: ");

Serial.println(EEPROMReadInt(2));

Serial.println("Pasos\_actual\_saved\_R2: ");

Serial.println(EEPROMReadInt(4));

Serial.println("Pasos\_actual\_saved\_L2: ");

Serial.println(EEPROMReadInt(6));

giro=3; //salir

}

}

}

}

void timerISR()

{

// Serial.println("Tiempo 1");

segundos ++;

// if (segundos <7) giro=0;

if (segundos ==tiempo\_1)

{giro=1;

}

if (segundos ==tiempo\_2) giro=2;

}

unsigned int EEPROMReadInt(int p\_address)

{

byte lowByte = EEPROM.read(p\_address);

byte highByte = EEPROM.read(p\_address + 1);

return ((lowByte << 0) & 0xFF) + ((highByte << 8) & 0xFF00);

}

void EEPROMWriteInt(int p\_address, int p\_value)

{

byte lowByte = ((p\_value >> 0) & 0xFF);

byte highByte = ((p\_value >> 8) & 0xFF);

EEPROM.write(p\_address, lowByte);

EEPROM.write(p\_address + 1, highByte);

}

**ANEXO III**

***Código Simulación Matlab***

clear

%variables:

%Cota de rueda a rueda

l=12.72;

%punto inicial

x(1)=10;

y(1)=10;

%MATLAB INPUT

w\_r=40;

w\_i=60;

t\_1=13;

t\_2=27;

%FIN MATLAB INPUT

%velocidad angular porcentual

wdp=[(w\_r/100) (w\_i/100)];

wip=[(w\_i/100) (w\_r/100)];

%Duración de cada velocidad

dt=[(t\_1/2) ((t\_2-t\_1)/2)];

%vd=[25 25 25 50];

%vi=[50 50 50 50];

%dt=[10 7 8 9 6 10 11]

%dt=[0.4 0.4 0.4 0.4];

%angulo inicial

fi=0\*180/pi;

%variables auxiliares

k=1;

c=1;

tit=0;

hold on

grid on

%marca con un + el punto inicial

plot(x(1),y(1),'-r +')

for j=1:2

%la velocidad tangencial es igual a: velocidad angular (200revoluciones

%por min)\*porcentaje de velocidad angular\*2pi\*radio de la ruaeda/60segundos

vd=wdp(j)\*(200/60)\*2\*pi\*2.945;

vi=wip(j)\*(200/60)\*2\*pi\*2.945;

if vd==vi

x(k+1)=x(k)+vd\*dt(j)\*cos(fi);

y(k+1)=y(k)+vd\*dt(j)\*sin(fi);

k=k+1;

else

%radio con respecto al centro instantaneo de curvatura

r(j)=(l/2)\*(vi+vd)/(vd-vi);

%angulo que adquiria el robot al final del recorrido

%(se divide por 40 para tener subsegmentos para el ploteo)

tit=(vd-vi)/(l\*40);

tit=tit\*dt(j);

for i=1:40

%matriz de rotación y traslación

MAT=[cos(tit) -sin(tit) 0; sin(tit) cos(tit) 0; 0 0 1];

%centro instantaneo de curvatura

ICCx(c)=x(k)-r(j)\*sin(fi);

ICCy(c)=y(k)+r(j)\*cos(fi);

%matriz que se va a multiplicar con MAT

mat2=[x(k)-ICCx(c); y(k)-ICCy(c); fi];

%matriz que se va a sumar con MAT

mat3=[ICCx(c); ICCy(c); tit];

%matriz resultante con x´e y´ (punto final de cada cambio

%de trayectoria y subtrayectoria de ploteo)

MATres=MAT\*mat2+mat3;

%vectores de x e y de trayectorias y subtrayectorias de ploteo

x(k+1)=MATres(1,1);

y(k+1)=MATres(2,1);

k=k+1;

%angulo final que tendra el robot al terminar cada cambio

%de trayectoria y subtrayectoria

fi=fi+tit;

%contador de ICC

if i==1%bandera para determinar cuando una nueva trayectoria se inicia

c=c+1;

plot(x(k),y(k),'-g o')

end

if i==40&&j==2

plot(x(k),y(k),'-r +')

end

end

plot(x(k),y(k),'-g o')

end

end

%vector de cordenadas x e y centro instantaneo de cada cambio de trayectoria

axis([-100 100 -100 100]);

plot(x,y,'-');

plot(ICCx,ICCy,'p');