

# **Informe**

## **Laboratorio 1**

Luis Ángel Vanegas Martínez

Problema #2

Diseño de Circuitos Combinacionales  
Arquitectura de Computadores y Laboratorio

Fredy Alexánder Rivera Vélez

Universidad de Antioquia

2021-1

# Objetivo

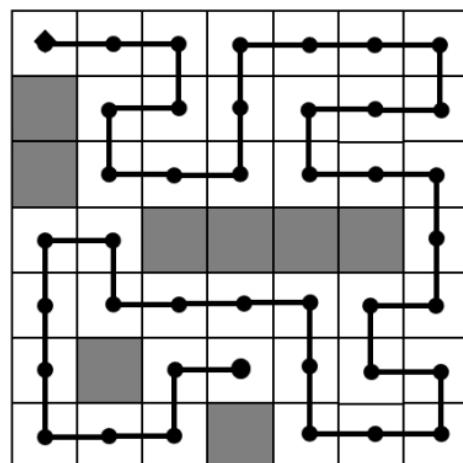
- Emplear los conocimientos teóricos adquiridos en el curso en el proceso de diseño de sistemas digitales combinacionales.
- Emplear herramientas de software para el diseño y la simulación de sistemas digitales.

## Problema Asignado

Problema #2

Acción	Código
Right + Go	00
Left + Go	10
Go	01
Stop	11

11	0	9	55	52	23	25
63	38	7	34	41	36	16
2	20	37	24	19	61	8
3	42	30	62	53	46	29
33	28	4	49	17	40	6
32	51	60	18	22	43	31
12	21	56	5	27	54	13



# Arquitecturas

## Arquitectura Propuesta

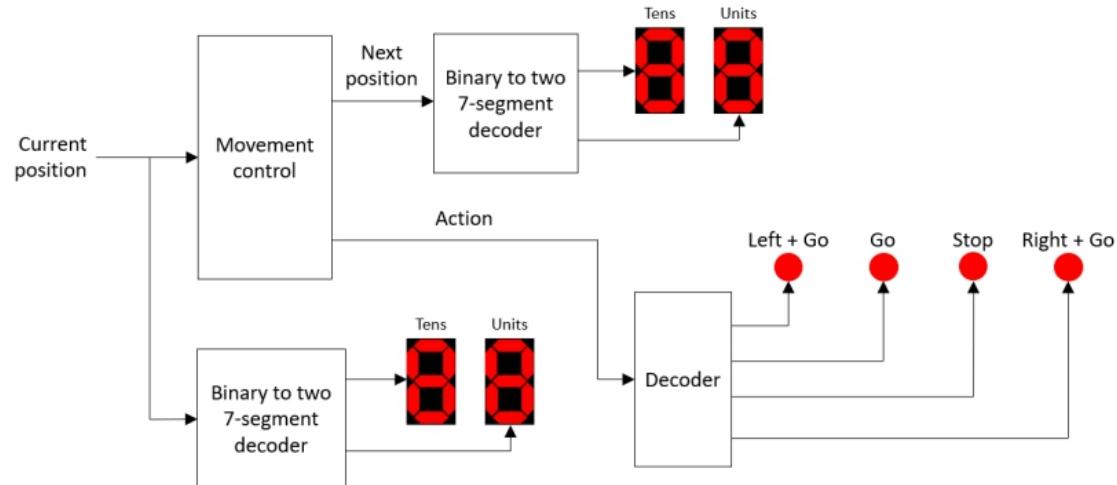
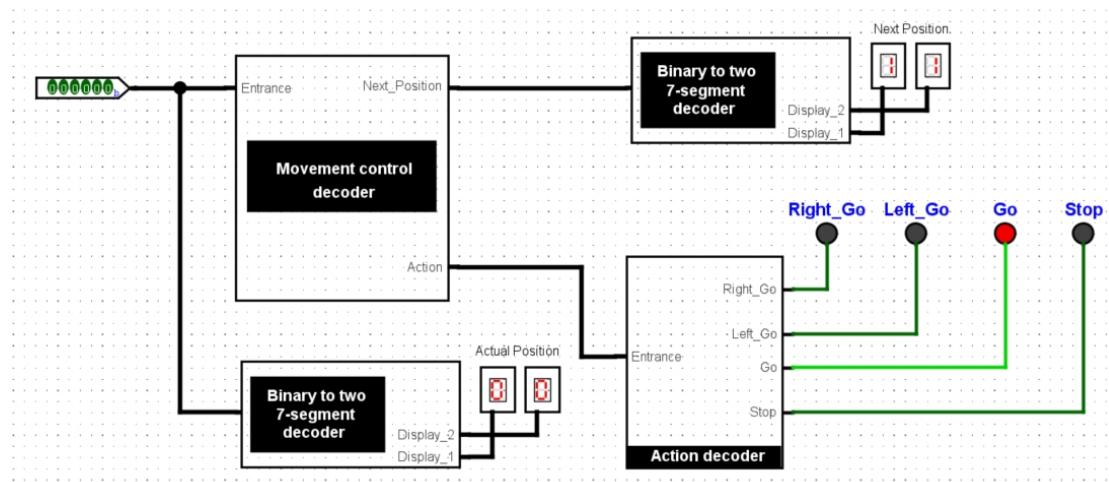
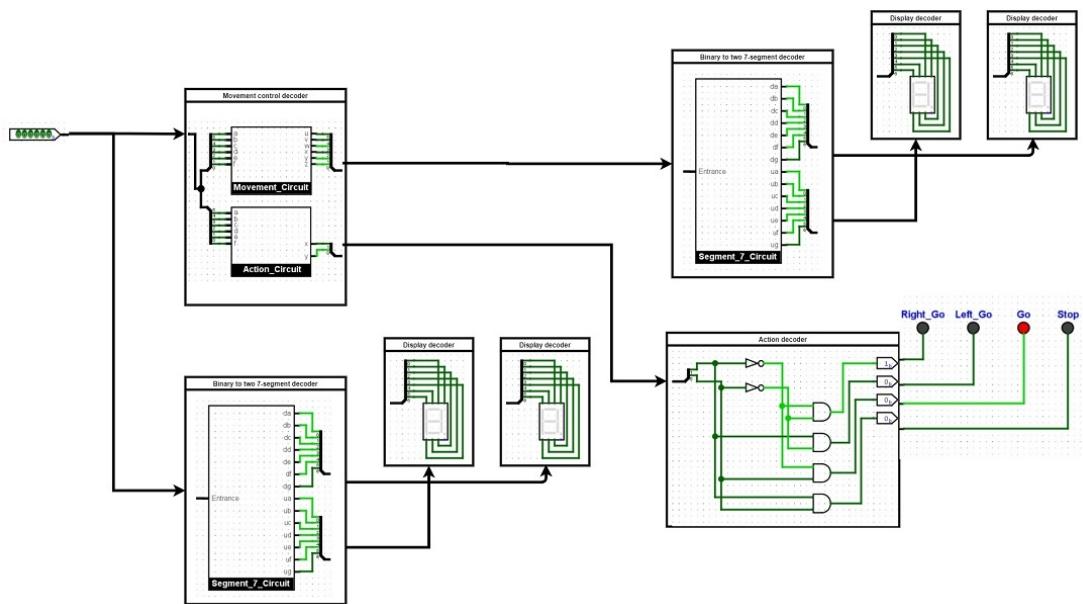


Figura 1. Arquitectura del sistema combinacional

## Arquitectura de la solución



## Arquitectura del Diseño de Circuito



## Decisiones de Diseño

Se ha tomado diferentes decisiones con respecto al diseño del circuito para llegar a la arquitectura propuesta y requerida.

1. Se usa separadores de bits para enlazar por medio de un único cable los diferentes componentes o decoders tal y como la arquitectura propuesta lo exige.
2. Se crea un componente **Display\_decoder** para instanciar los 4 displays requeridos en la solución y este se crea también teniendo en cuenta el punto 1 anterior.
3. Se crean componentes principales: **Movement control decoder**, **Binary to two 7-segment decoder**, **Action decoder**. Los cuales cada uno instancia sub componentes o circuitos combinacionales previamente diseñados.
4. De acuerdo al problema #2 que me correspondió, se toma el número mayor usado en este problema el cual es el: **63** que en binario corresponde a **111111** igual a **6 bits**. Partiendo de esto, decido que la entrada de datos del robot será de 6 bits.
5. Para tener una consistencia de datos, las entradas de los componentes: **Binary to two 7-segment decoder** y **Movement control decoder** corresponden a 6 bits incluyendo la entrada de los sub circuitos combinacionales que contienen estos.
6. El diseño de los sub circuitos combinacionales se hacen partiendo de tablas de verdad con entradas de 6 bits para todos sus circuitos, excepto para el circuito **Circuit\_Action\_Decoder** (que es el mismo componente **Action Decoder**) debido a que la entrada de este será en 2 bits.

# Proceso de Diseño

Se inicia diseñando las tablas de verdad para cada sub circuito combinacional.

## - Circuit\_Movement

### Análisis

- Se crean entradas de 6 bits para los 64 números posibles a usar por el robot.
- Las entradas de los 64 números se crean en orden de 0 a 63, siendo cada entrada el número correspondiente en binario y la salida la posición siguiente que debería tomar el robot.
- Para los números que no están mapeados en la matriz del problema, como los son: (1,2,5,10,14,15,26,30,35,39,44,45,46,47,48,50,51,53,57,58,59,62,63,64) tendrán como salida el mismo número. Ejemplo del número 1:

a	b	c	d	e	f		u	v	w	x	y	z
0	0	0	0	0	0		0	0	1	0	1	1
0	0	0	0	0	1		0	0	0	0	0	1
0	0	0	0	1	0		0	0	0	0	1	0

- De este circuito la entrada proviene del usuario y la salida va directa a los display que mostrarán la posición siguiente

### Tabla de Verdad

a	b	c	d	e	f		u	v	w	x	y	z
0	0	0	0	0	0		0	0	1	0	1	1
0	0	0	0	0	1		0	0	0	0	0	1
0	0	0	0	1	0		0	0	0	0	1	0
0	0	0	1	0	0		1	0	1	0	1	0
0	0	0	1	0	1		1	1	0	0	1	1
0	0	1	0	0	0		0	0	0	1	0	1
0	0	1	0	0	1		0	0	1	0	0	1
0	0	1	0	1	0		1	1	1	0	1	0
0	0	1	0	1	1		1	1	1	1	0	1
0	0	1	1	0	0		0	1	1	1	0	0
0	0	1	1	0	1		0	1	1	1	1	0
0	0	1	1	1	0		0	0	1	1	1	1
0	0	1	1	1	1		0	1	0	0	1	1
0	1	0	0	0	0		0	1	0	0	1	0
0	1	0	0	0	1		0	1	0	1	1	0
0	1	0	0	1	0		0	1	0	1	1	0
0	1	0	0	1	1		0	1	1	1	1	0
0	1	0	1	0	0		1	1	1	1	0	0
0	1	0	1	0	1		1	1	1	1	1	0
0	1	0	1	1	0		0	1	1	1	1	1
0	1	0	1	1	1		0	1	0	0	1	1
0	1	1	0	0	0		0	0	0	0	0	0
0	1	1	0	0	1		0	0	0	0	0	1
0	1	1	0	1	0		0	0	0	0	1	0
0	1	1	0	1	1		0	0	0	0	1	1
0	1	1	1	0	0		0	0	1	1	1	1
0	1	1	1	0	1		0	0	1	1	1	0
0	1	1	1	1	0		0	0	1	1	1	1
0	1	1	1	1	1		0	0	0	0	0	0

010011		101001
010100		100110
010101		001100
010110		011011
010111		110100
011000		100101
011001		010111
011010		011010
011011		110110
011100		000100
011101		001000
011110		011110
011111		101011
100000		100001
100001		000011
100010		011000
100011		100011
100100		010000
100101		010100
100110		000111
100111		100111
101000		000110
101001		100100
101010		011100
101011		101000
101100		101100
101101		101101
101110		101110
101111		101111
110000		110000
110001		010001
110010		110010
110011		110011
110100		110111
110101		110101
110110		001101
110111		100010
111000		010101
111001		111001
111010		111010
111011		111011
111100		111000
111101		010011
111110		111110
111111		111111

Expresión Minimizada

		e					
0	0	1	0	0	0	0	1
1	0	0	0	0	0	0	1
1	0	1	0	0	1	0	0
0	0	1	1	0	1	0	1
1	0	1	1	0	1	1	1
0	1	1	1	1	1	0	1
0	1	1	0	1	1	1	1
1	0	1	0	0	1	0	0

c·d·e·f+a·b·d·e·f+a·c·d·e·f+b·c·d·e+b·e·f+f+b·c·d·e·f+a·c·d·e·f+a·e·f+f+a·c·d·f+a·b·c·d+a·c·d·f+a·b·d·e+a·b·c·d·e

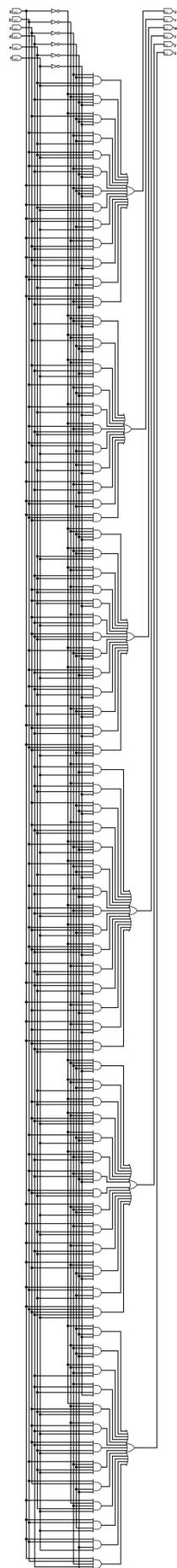
## Expresión de Salida

```

u = c·d·e·f+a·b·d·e·f+a·c·d·e·f+b·c·d·e·f+a·c·d·e·f+a·c·d·f+a·b·c·d+a·c·d·f+a·b·d·e+a·b·c·d·e
v = a·b·c·d·f+a·b·c·d·e·f+a·b·c·d·e·f+b·c·d·e+a·b·e·f+f+a·b·c·d·e+b·c·d·f+a·d·e·f+a·c·d·e+a·b·d+a·b·c
w = a·b·d·e·f+a·c·d·e·f+a·b·d·e+b·c·e+f+b·c·d·e+a·b·c·d·f+a·b·d·e·f+b·d·e·f+a·b·c·d·f+a·b·c·d·f
x = b·d·e·f+b·d·e·f+c·d·e·f+b·c·d·e+a·b·c·e·f+a·b·c·d·e+f+b·c·d·e+a·b·c·d·f+a·b·c·d·f+a·b·c·d·f+a·b·c·f+a·c·d·e
y = a·b·c·d·f+a·b·d·e+a·c·e+a·b·c·d·f+a·b·d·e·f+a·b·c·d·f+f+b·c·e+a·b·c·d·f+a·c·e·f+f+a·b·d·e+a·b·c·d·f
z = a·d·e·f+b·c·d·e+a·b·c·d·c·d·e·f+a·b·c·e·f+f+b·c·d·e·f+b·c·d·e·f+a·b·c·d·e·f+a·b·d·f+a·b·e·f+f+a·b·c·d·e

```

## Circuito



## - Circuit\_Action

### Análisis

- Se crean entradas de 6 bits para los 19 números que usará el robot para hacer el recorrido hasta la posición final.
- Las entradas de los 19 números se crean en orden desde 18 a 11 tal y como indica el camino del problema #2, siendo cada entrada el número correspondiente en binario y la salida el código correspondiente a la acción que ejecutará el robot de acuerdo a la posición donde se encuentra. Código de la acción:

Acción	Código
<i>Right + Go</i>	00
<i>Left + Go</i>	10
<i>Go</i>	01
<i>Stop</i>	11

Ejemplo de la primera acción a ejecutar por el robot (Go):

a	b	c	d	e	f		x	y
0	1	0	0	1	0		0	1
1	1	1	1	0	0		1	0
1	1	1	0	0	0		0	0

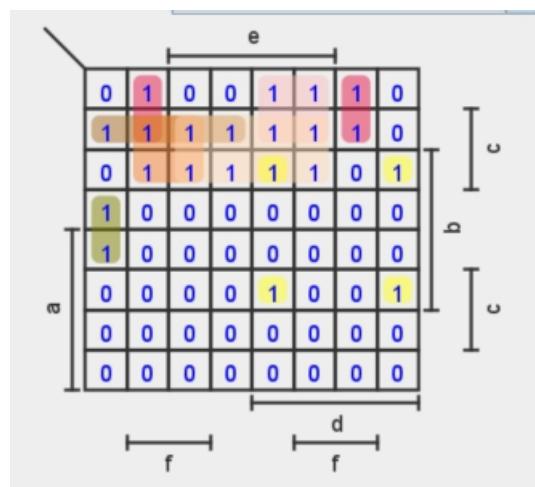
Desde el 18 ejecuta la acción con código 01 para ir al 60.

### Tabla de Verdad

a b c d e f   x y
0 1 0 0 1 0   0 1
1 1 1 1 0 0   1 0
1 1 1 0 0 0   0 0
0 1 0 1 0 1   0 1
0 0 1 1 0 0   0 0
1 0 0 0 0 0   0 1
1 0 0 0 0 1   0 1
0 0 0 0 1 1   0 0
1 0 1 0 1 0   0 0
0 1 1 1 0 0   1 0
0 0 0 1 0 0   0 1
1 1 0 0 0 1   0 1
0 1 0 0 0 1   0 0
0 1 0 1 1 0   0 1
0 1 1 0 1 1   1 0
1 1 0 1 1 0   0 1

001101		10
011111		10
101011		00
101000		00
000110		10
011101		01
001000		10
111101		01
010011		00
101001		00
100100		01
010000		10
011001		10
010111		01
110100		01
110111		00
100010		01
011000		00
100101		01
010100		00
100110		00
000111		10
001001		10
000000		01
001011		11

### Expresión Minimizada

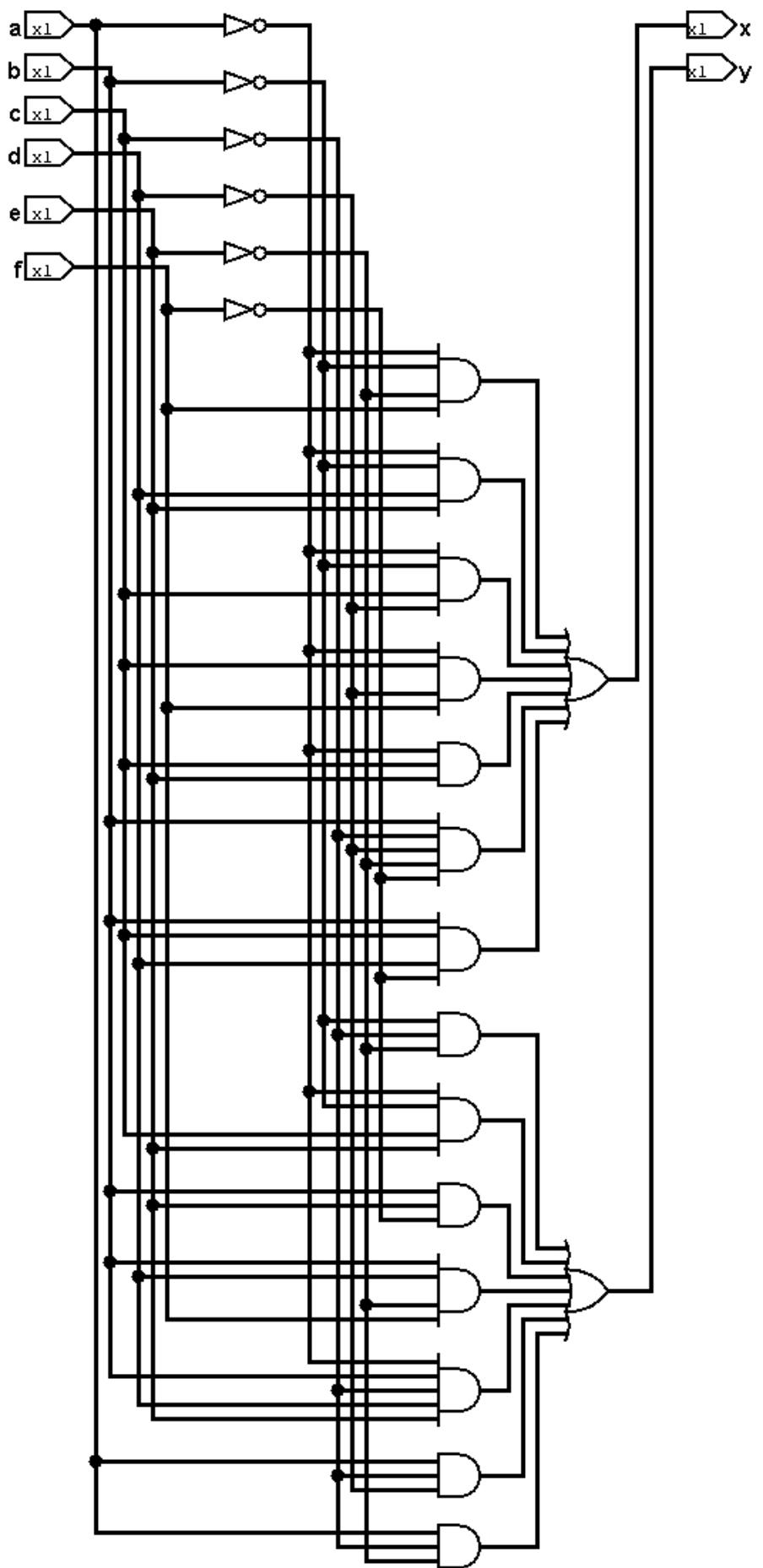


$$x = \overline{a} \cdot \overline{b} \cdot e \cdot f + \overline{a} \cdot \overline{b} \cdot \overline{d} \cdot e + a \cdot \overline{b} \cdot c \cdot d + a \cdot c \cdot \overline{d} \cdot f + a \cdot c \cdot e + b \cdot c \cdot \overline{d} \cdot e + \overline{f} + b \cdot c \cdot d \cdot f \\ y = b \cdot c \cdot e + a \cdot b \cdot c \cdot e + b \cdot \overline{d} \cdot e \cdot f + b \cdot d \cdot e \cdot f + a \cdot b \cdot c \cdot d + a \cdot c \cdot d + a \cdot c \cdot e$$

### Expresión de Salida

$$x = \overline{a} \cdot \overline{b} \cdot e \cdot f + \overline{a} \cdot \overline{b} \cdot \overline{d} \cdot e + a \cdot \overline{b} \cdot c \cdot d + a \cdot c \cdot \overline{d} \cdot f + a \cdot c \cdot e + b \cdot c \cdot \overline{d} \cdot e + \overline{f} + b \cdot c \cdot d \cdot f \\ y = b \cdot c \cdot e + a \cdot b \cdot c \cdot e + b \cdot \overline{d} \cdot e \cdot f + b \cdot d \cdot e \cdot f + a \cdot b \cdot c \cdot d + a \cdot c \cdot d + a \cdot c \cdot e$$

Circuito



## - Action\_Decoder

### Análisis

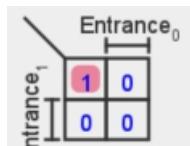
- Se crean entradas de 2 bits para los 4 códigos que usará el robot al ejecutar una acción.
- Las entradas son el código de la acción que ejecutará el robot y la salida es de 1 bit correspondiente al cable que llevará la energía al LED de acuerdo al código de la acción

### Tabla de Verdad

Entrada r: right\_go, Entrada l: left\_go, Entrada g: go, Entrada s: stop

a b		r	l	g	s
0 0		1	0	0	0
1 0		0	1	0	0
0 1		0	0	1	0
1 1		0	0	0	1

### Expresión Minimizada



$$\overline{\text{Entrance}_1} \cdot \overline{\text{Entrance}_0}$$

### Expresión de Salida

```
Right_Go =  $\overline{\text{Entrance}_1} \cdot \overline{\text{Entrance}_0}$ 
Left_Go =  $\text{Entrance}_1 \cdot \overline{\text{Entrance}_0}$ 
Go =  $\text{Entrance}_1 \cdot \text{Entrance}_0$ 
Stop =  $\overline{\text{Entrance}_1} \cdot \text{Entrance}_0$ 
```

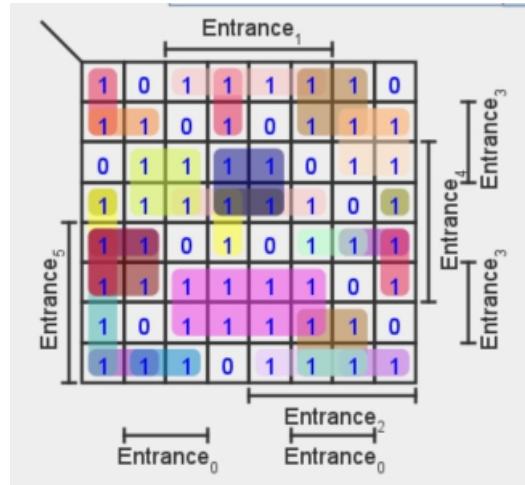
### Circuito



001000	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
001001	1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 0
001010	1 1 1 1 1 1 0 0 1 1 1 0 0 0 0
001011	0 1 1 0 0 0 0 0 1 1 0 0 0 0
001100	1 1 0 1 1 0 1 0 1 1 1 0 0 0 0
001101	1 1 1 1 0 0 1 0 1 1 1 0 0 0 0
001110	0 1 1 0 0 1 1 0 1 1 1 0 0 0 0
001111	1 0 1 1 0 1 1 0 1 1 1 0 0 0 0
010000	1 0 1 1 1 1 1 0 1 1 1 0 0 0 0
010001	1 1 1 0 0 0 0 0 1 1 0 0 0 0
010010	1 1 1 1 1 1 1 0 1 1 1 0 0 0 0
010011	1 1 1 0 0 1 1 0 1 1 1 0 0 0 0
010100	1 1 1 1 1 1 0 1 1 0 1 1 0 1
010101	0 1 1 0 0 0 0 1 1 0 1 1 0 1
010110	1 1 0 1 1 0 1 1 1 0 1 1 0 1
010111	1 1 1 1 0 0 1 1 1 0 1 1 0 1
011000	0 1 1 0 0 1 1 1 1 0 1 1 0 1
011001	1 0 1 1 0 1 1 1 1 0 1 1 0 1
011010	1 0 1 1 1 1 1 1 1 0 1 1 0 1
011011	1 1 1 0 0 0 0 1 1 0 1 1 0 1
011100	1 1 1 1 1 1 1 1 1 0 1 1 0 1
011101	1 1 1 0 0 1 1 1 1 0 1 1 0 1
011110	1 1 1 1 1 1 0 1 1 1 1 0 0 1
011111	0 1 1 0 0 0 0 1 1 1 1 0 0 1
100000	1 1 0 1 1 0 1 1 1 1 1 0 0 1
100001	1 1 1 1 0 0 1 1 1 1 1 0 0 1
100010	0 1 1 0 0 1 1 1 1 1 1 0 0 1
100011	1 0 1 1 0 1 1 1 1 1 1 0 0 1
100100	1 0 1 1 1 1 1 1 1 1 1 0 0 1
100101	1 1 1 0 0 0 0 1 1 1 1 0 0 1
100110	1 1 1 1 1 1 1 1 1 1 1 0 0 1
100111	1 1 1 0 0 1 1 1 1 1 1 0 0 1
101000	1 1 1 1 1 1 0 0 1 1 0 0 1 1
101001	0 1 1 0 0 0 0 0 1 1 0 0 1 1
101010	1 1 1 0 1 1 1 0 1 1 0 0 1 1
101011	1 1 1 1 0 0 1 0 1 1 0 0 1 1
101100	0 1 1 0 0 1 1 0 1 1 0 0 1 1
101101	1 0 1 1 0 1 1 0 1 1 0 0 1 1
101110	1 0 1 1 1 1 1 0 1 1 0 0 1 1
101111	1 1 1 0 0 0 0 0 1 1 0 0 1 1
110000	1 1 1 1 1 1 1 0 1 1 0 0 1 1
110001	1 1 1 0 0 1 1 0 1 1 0 0 1 1
110010	1 1 1 1 1 1 1 0 1 0 1 1 0 1 1
110011	0 1 1 0 0 0 0 1 0 1 1 0 1 1
110100	1 1 0 1 1 0 1 1 0 1 1 0 1 1
110101	1 1 1 1 0 0 1 1 0 1 1 0 1 1
110110	0 1 1 0 0 1 1 1 0 1 1 0 1 1
110111	1 0 1 1 0 1 1 1 0 1 1 0 1 1
111000	1 0 1 1 1 1 1 1 0 1 1 0 1 1
111001	1 1 1 0 0 0 0 1 0 1 1 0 1 1
111010	1 1 1 1 1 1 1 1 0 1 1 0 1 1

111011	1 1 1 0 0 1 1 1 0 1 1 0 1 1 1
111100	1 1 1 1 1 1 0 1 0 1 1 1 1 1
111101	0 1 1 0 0 0 0 1 0 1 1 1 1 1
111110	1 1 0 1 1 0 1 1 0 1 1 1 1 1
111111	1 1 1 1 0 0 1 1 0 1 1 1 1 1

## Expresión Minimizada

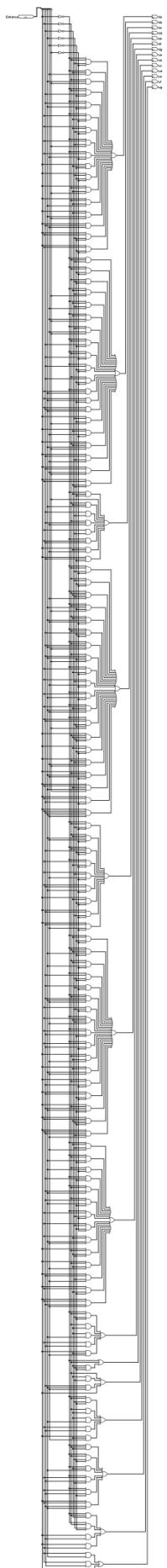


$$\begin{aligned}
 & \overline{\text{Entrance}_5} \text{ } \overline{\text{Entrance}_4} \text{ } \overline{\text{Entrance}_2} \text{ } \overline{\text{Entrance}_0} + \overline{\text{Entrance}_5} \text{ } \overline{\text{Entrance}_4} \text{ } \text{Entrance}_3 + \overline{\text{Entrance}_5} \text{ } \text{Entrance}_4 \text{ } \overline{\text{Entrance}_1} + \\
 & \overline{\text{Entrance}_5} \text{ } \text{Entrance}_3 \text{ } \overline{\text{Entrance}_2} + \overline{\text{Entrance}_5} \text{ } \text{Entrance}_2 \text{ } \overline{\text{Entrance}_1} + \overline{\text{Entrance}_5} \text{ } \text{Entrance}_1 \text{ } \overline{\text{Entrance}_0} + \overline{\text{Entrance}_4} \text{ } \text{Entrance}_3 \text{ } \overline{\text{Entrance}_0} + \\
 & \overline{\text{Entrance}_4} \text{ } \text{Entrance}_3 \text{ } \text{Entrance}_2 + \overline{\text{Entrance}_4} \text{ } \text{Entrance}_3 \text{ } \text{Entrance}_1 + \overline{\text{Entrance}_4} \text{ } \text{Entrance}_2 \text{ } \text{Entrance}_1 + \overline{\text{Entrance}_4} \text{ } \text{Entrance}_1 \text{ } \text{Entrance}_0 + \\
 & \text{Entrance}_3 \text{ } \text{Entrance}_2 \text{ } \text{Entrance}_1 + \text{Entrance}_3 \text{ } \text{Entrance}_2 \text{ } \text{Entrance}_0 + \text{Entrance}_3 \text{ } \text{Entrance}_1 \text{ } \text{Entrance}_0 + \text{Entrance}_2 \text{ } \text{Entrance}_1 \text{ } \text{Entrance}_0
 \end{aligned}$$

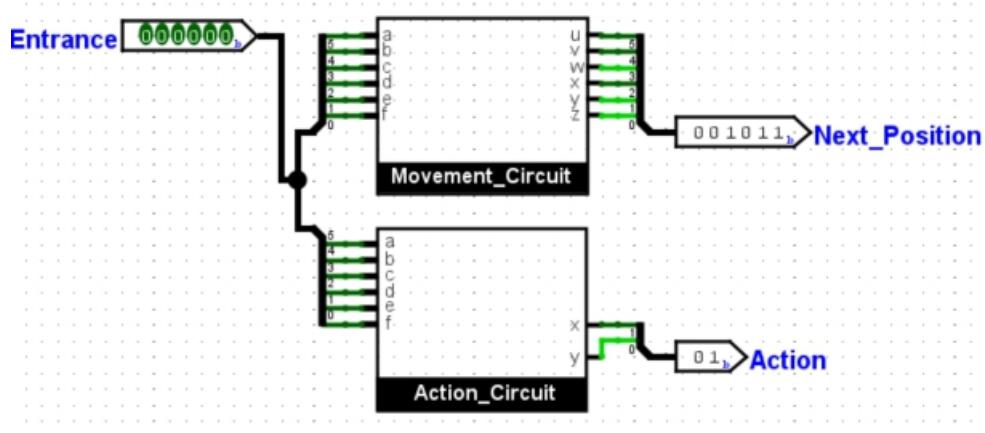
## Expresión de Salida

$$\begin{aligned}
 & \overline{\text{Entrance}_5} \text{ } \overline{\text{Entrance}_4} \text{ } \overline{\text{Entrance}_2} \text{ } \overline{\text{Entrance}_0} + \overline{\text{Entrance}_5} \text{ } \overline{\text{Entrance}_4} \text{ } \text{Entrance}_3 + \overline{\text{Entrance}_5} \text{ } \text{Entrance}_4 \text{ } \overline{\text{Entrance}_1} + \\
 & \overline{\text{Entrance}_5} \text{ } \text{Entrance}_3 \text{ } \overline{\text{Entrance}_2} + \overline{\text{Entrance}_5} \text{ } \text{Entrance}_2 \text{ } \overline{\text{Entrance}_1} + \overline{\text{Entrance}_5} \text{ } \text{Entrance}_1 \text{ } \overline{\text{Entrance}_0} + \overline{\text{Entrance}_4} \text{ } \text{Entrance}_3 \text{ } \overline{\text{Entrance}_0} + \\
 & \overline{\text{Entrance}_4} \text{ } \text{Entrance}_3 \text{ } \text{Entrance}_2 + \overline{\text{Entrance}_4} \text{ } \text{Entrance}_3 \text{ } \text{Entrance}_1 + \overline{\text{Entrance}_4} \text{ } \text{Entrance}_2 \text{ } \text{Entrance}_1 + \overline{\text{Entrance}_4} \text{ } \text{Entrance}_1 \text{ } \text{Entrance}_0 + \\
 & \text{Entrance}_3 \text{ } \text{Entrance}_2 \text{ } \text{Entrance}_1 + \text{Entrance}_3 \text{ } \text{Entrance}_2 \text{ } \text{Entrance}_0 + \text{Entrance}_3 \text{ } \text{Entrance}_1 \text{ } \text{Entrance}_0 + \text{Entrance}_2 \text{ } \text{Entrance}_1 \text{ } \text{Entrance}_0
 \end{aligned}$$

## Círcuito



## Componente Movement\_Control\_Decoder

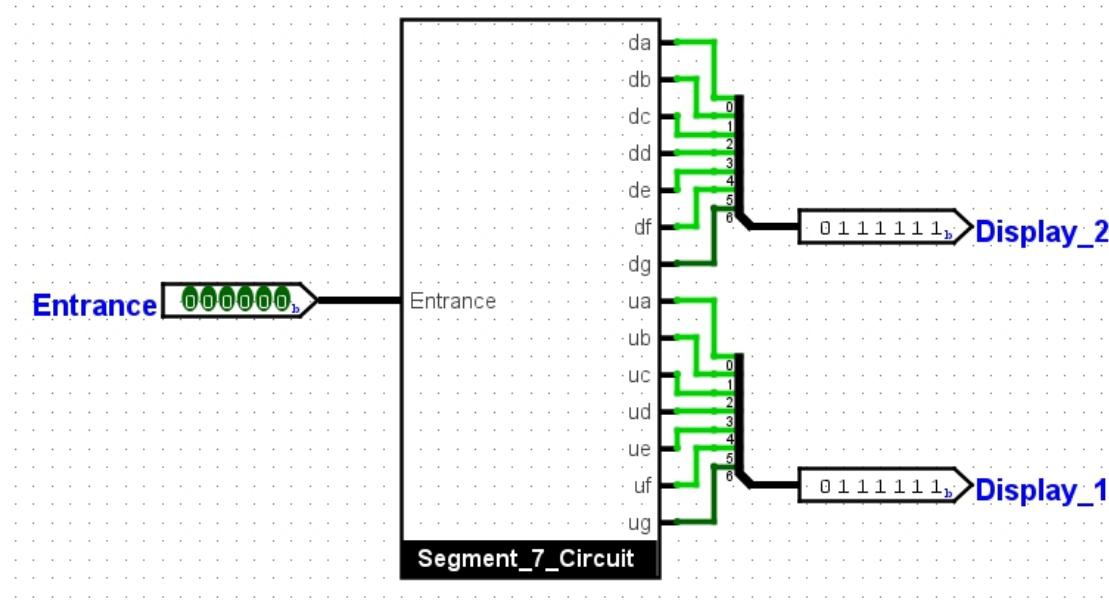


Este componente es creado partiendo de los circuitos **Action\_Circuit** y **Movement\_Circuit**. Los cuales ambos tienen como entradas un número en base binaria de 6 bits.

**Action\_Circuit** El tiene una salida de 2 bits (como describo anteriormente) y este conecta directamente al componente Action\_Decoder.

**Movement\_Circuit** El tiene una salida de 6 bits (como describo anteriormente) y este conecta directamente al componente Binary\_to\_two\_7\_Segment\_Decoder.

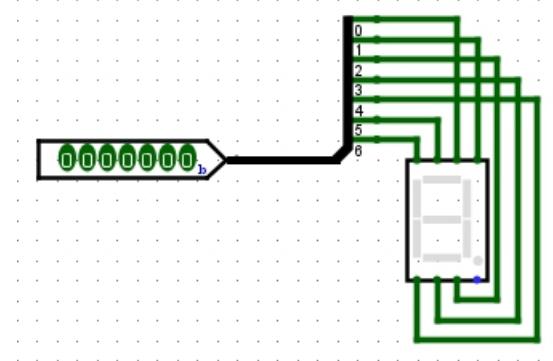
## Componente Binary\_to\_two\_7\_Segment\_Decoder



Este componente es creado partiendo del circuito **Circuit\_two\_7\_Segment**. El cual tiene como entrada un número en base binaria de 6 bits y como salida 14 bits, de los cuales los 7 primeros corresponden al número binario expresado en decimal del número que va en el

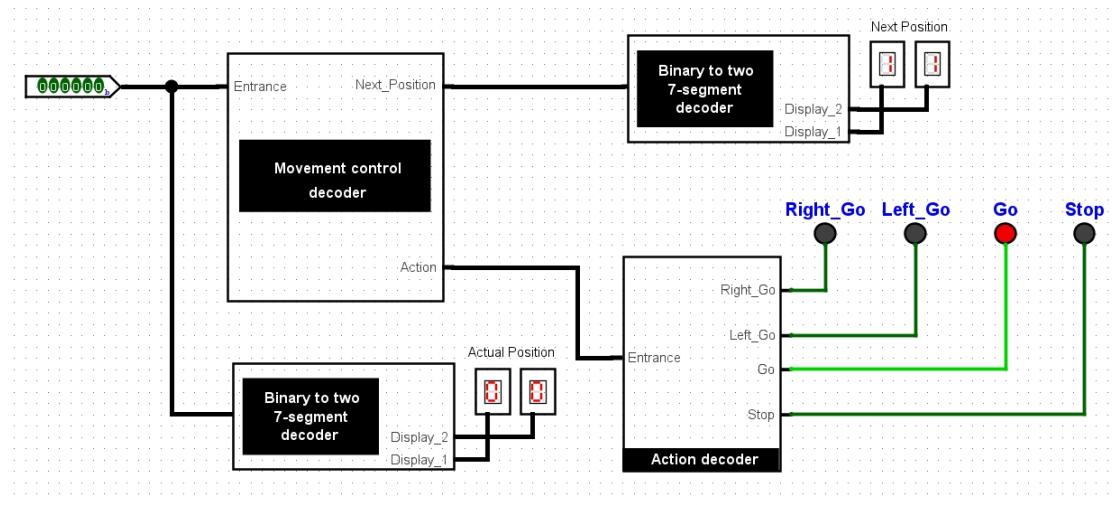
display 2 y los 7 bits restantes corresponden al número binario expresado en decimal del número que va en el display 1.

## Componente Display\_Decoder



Este componente es creado con el fin de incrustar el 7 segmentos de forma ordenada con un separador de bits de 7 y una entrada de 7bits para que posteriormente se conecte al decoder del número a mostrar.

## Componente Main



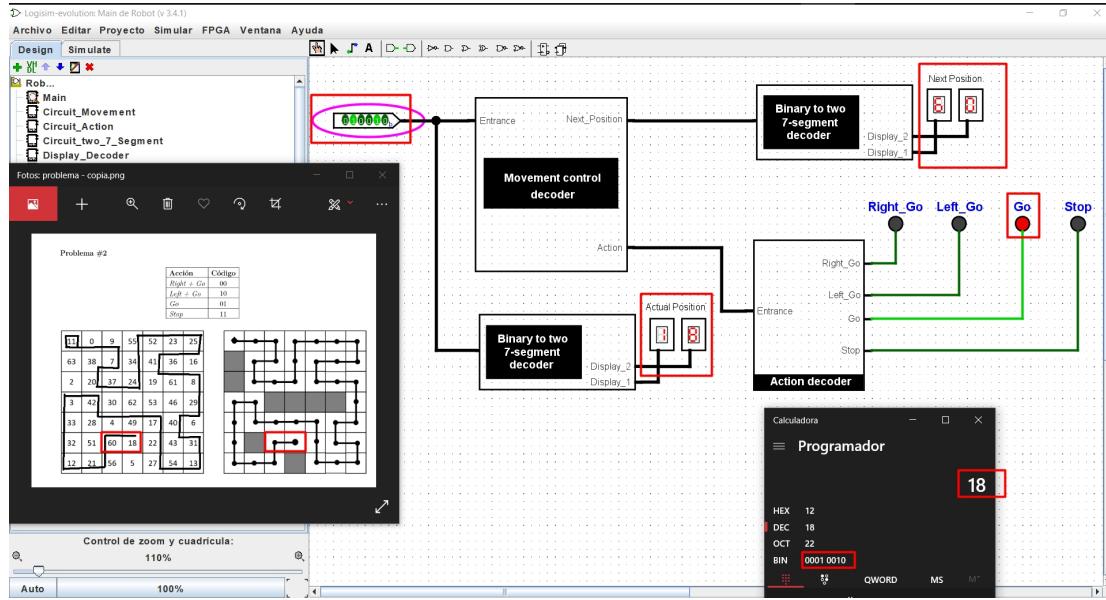
En este componente integro instanciando los demás componentes con el fin de que todos trabajen mancomunadamente para satisfacer las necesidades y problemáticas propuestas, cumpliendo con la arquitectura exigida.

Cada vez que se digite un número entre 0 y 63 en base binaria, ese número viaja a binary to tow 7-segment decoder para mostrar el número ingresado en base decimal en el display de 7 segmentos, ese mismo número digitado viaja a Movement control decoder, donde se obtendrá la acción a realizar y el valor de la siguiente posición. Luego en el Action decoder se toma el código de la acción a realizar y se decodifica en un bit que se encarga de encender el LED correspondiente a dicha acción.

# Simulación

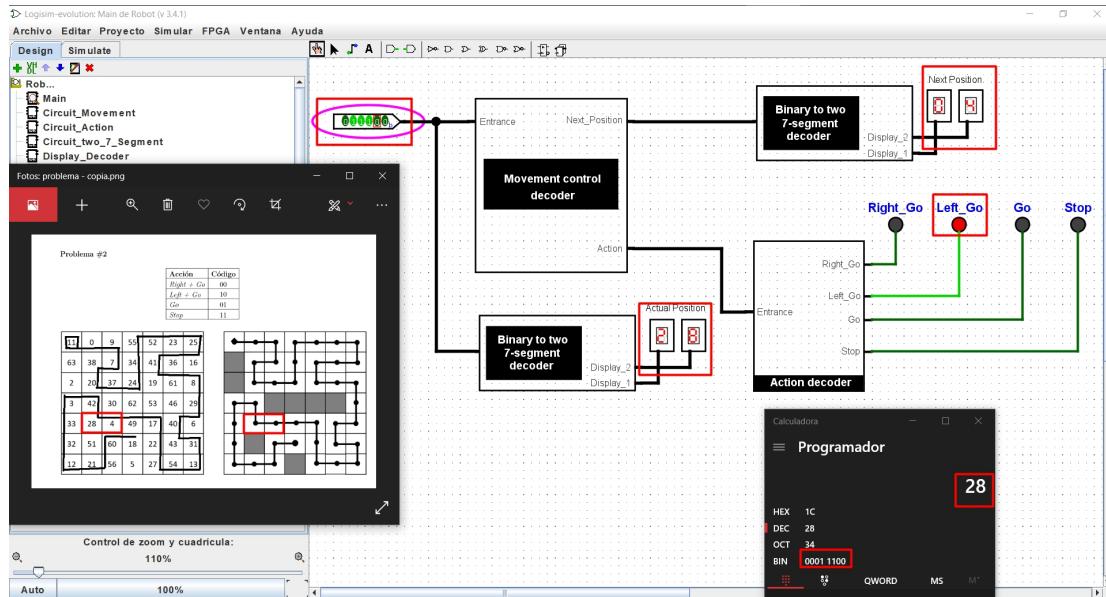
La simulación la haremos con base a distintos casos. Caso real, caso probable, caso extremo.

## Caso real 1:



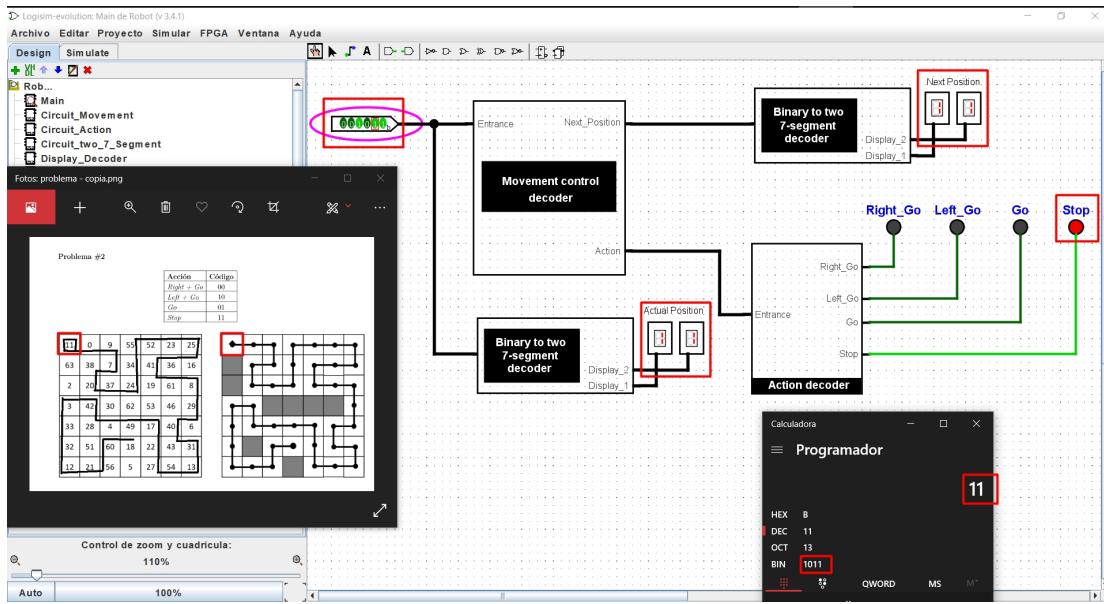
Visualizamos el número 18 en binario como: 010010 en la entrada y como posición actual, el LED indica que la acción es Go y que la siguiente posición es 60; de acuerdo al diagrama, es correcto.

## Caso Real 2:



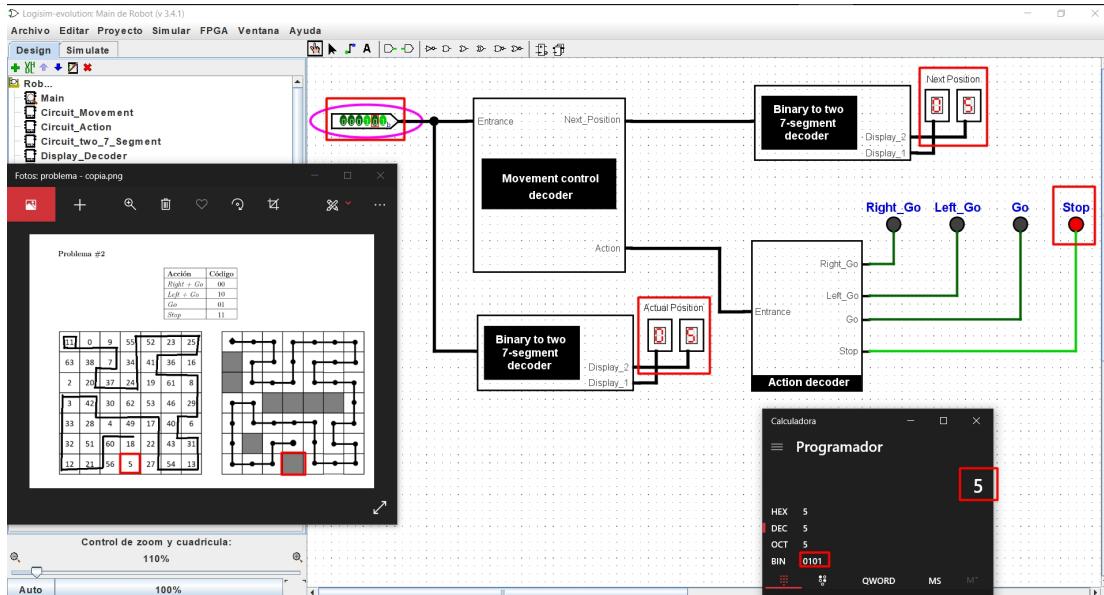
En este caso vemos que la posición actual es 28 denotada en binario como: 011100 de entrada, la acción a ejercer sería ir Left\_Go para ir a la posición siguiente que es 04, la cual se compara con el diagrama y muestra que es correcto.

## Caso Probable 1:



Este caso probable corresponde a que digiten como posición actual el 11. si vemos es el valor donde se quiere llegar y por consiguiente la acción a tomar es Stop y la siguiente posición es el mismo número 11 porque no puede avanzar.

## Caso Extremo 1:

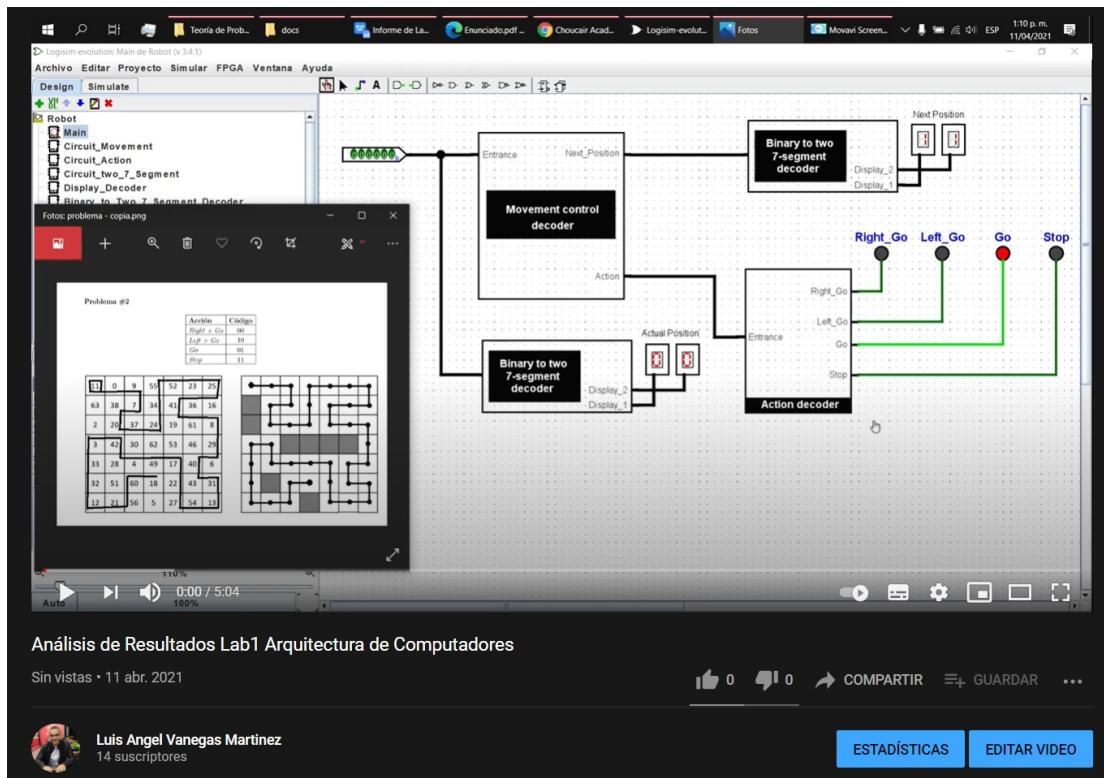


Este caso sucede cuando se pone una posición actual que no existe en la tabla o que este sombreada, mostrará el valor de entrada que se dio y la acción será Stop, adicional la siguiente posición es el mismo valor de entrada porque no puede avanzar.

# Análisis de Resultados

Para un mejor entendimiento plantee el análisis de resultados en forma de vídeo para que vean la interacción de una forma dinámica y orientada a un caso de uso real.

Clic aquí para ver el vídeo: <https://youtu.be/CfGTDqGxXZg>



## **Observaciones**

Todos los componentes han sido diseñados con el fin de que sean reutilizables y escalables en términos de arquitectura. En caso de querer implementar una nueva funcionalidad, basta con instancia un componente requerido y asignarle los valores de entrada requeridos para que de sus valores de salidas.

El robot se encuentra en la nube documentado y con licencia MIT para un futuro uso de acuerdo a las políticas que yo como creador asigno. El repositorio en mención es:  
<https://github.com/xlavm/Logisim-Circuits/tree/main/Lab-1>

## **Conclusión**

Para concluir, Se diseñan circuitos combinacionales que luego son utilizados por componentes de un nivel medio y que se interconectan entre si de acuerdo a la problemática y arquitectura planteada. Estos circuitos son utilizados de forma repetida por los componentes que los requieran; Así mismo estos componentes de nivel medio son instanciados por un componente de alto nivel llamado “Main” este es el componente que se encargará de ejecutar las funcionalidades de los otros componentes de nivel medio y de los circuitos.

Para soluciones más optimas que contribuyan a una mayor escalabilidad podrían emplearse otros componentes como de “Memoria” por ejemplo ROM que registren los valores del problema o problemas propuestos con el fin de que en caso de un cambio de problema, la arquitectura del circuito no cambie. Esta solución la he planteado y se encuentra en el siguiente repositorio: <https://github.com/xlavm/Logisim-Circuits/tree/main/Lab-1-simplified>