



UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

**Escola Superior d'Enginyeries Industrial,
Aeroespacial i Audiovisual de Terrassa**

INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

TRABAJO DE FIN DE GRADO

Torrecilla Matencio, Marc

ESTUDIO DE LAS ETAPAS DE AUTOMATIZACIÓN DE UN PROCESO INDUSTRIAL

ANEXO III

Director: Delgado Prieto, Miguel

Co-director: Fernández Sobrino, Ángel

Convocatoria: Julio, 2020

SUMARIO

Figura 1. Flujo General [1].	1
Figura 2. Flujo General [2].	1
Figura 3. Nodo función GET AUTO CARGADO [1].	2
Figura 4. Nodo función GET AUTO CARGADO [2].	2
Figura 5. Nodo función GET AUTO CARGADO [3].	2
Figura 6. Nodo función GET MATERIAS CARGADAS.	3
Figura 7. Nodo Función GET LOTE CARGADO.	3
Figura 8. Nodo función GET LOTE FINAL.	4
Figura 9. Nodo función GET PROD FINAL [1].	4
Figura 10. Nodo función GET PROD FINAL [2].	4
Figura 11. Nodo función GET PROD FINAL [3].	5
Figura 12. Nodo función GET (materias actuales abastecidas en la línea)	5
Figura 13. Nodo función GET (tipo cargado 1).	5
Figura 14. Node función GET (tipo cargado 2).	5
Figura 15. Nodo función GET (tipo 3 cargado).	6
Figura 16. Nodo función GET (tipo 1 finalizado).	6
Figura 17. Nodo función GET (tipo 2 finalizado).	6
Figura 18. Nodo función GET (tipo 3 finalizado).	6
Figura 19. Nodo función FLUJO TIPO POSICIONES.	7
Figura 20. Nodo función GET POSICION (posición 1).	7
Figura 21. Nodo función GET POSICION (posición 2).	7
Figura 22. Nodo función GET POSICION (posición 3).	8
Figura 23. Nodo función GET POSICION (posición 4).	8
Figura 24. Nodo función GET POSICION (posición 5).	8
Figura 25. Nodo función GET POSICION (posición 6).	8
Figura 26. Nodo función GET POSICION (posición 7).	9
Figura 27. Nodo función GET POSICION (posición 8).	9
Figura 28. Nodo función GET POSICION (posición 9).	9
Figura 29. Nodo función GET POSICION (posición 10).	9
Figura 30. Nodo toJSON (auto cargado).	10
Figura 31. Nodo toJSON (producto finalizado).	10
Figura 32. Nodo función DIRECCIÓN (posición 1).	11
Figura 33. Nodo función DIRECCIÓN (posición 2).	11
Figura 34. Nodo función DIRECCIÓN (posición 3).	11
Figura 35. Nodo función DIRECCIÓN (posición 4).	12
Figura 36. Nodo función DIRECCIÓN (posición 4).	12
Figura 37. Nodo función DIRECCIÓN (posición 5).	12
Figura 38. Nodo función DIRECCIÓN (posición 6).	13
Figura 39. Nodo función DIRECCIÓN (posición 7).	13
Figura 40. Nodo función DIRECCIÓN (posición 8).	13
Figura 41. Nodo función DIRECCIÓN (posición 9).	14
Figura 42. Nodo función DIRECCIÓN (posición 10).	14
Figura 43. Flujo Ciclos.	15
Figura 44. Nodo función STORE EMERGENCY.	16
Figura 45. Nodo función STORE DATA.	16

Figura 46. Nodo función GET LOTE CANTIDAD (tipo 1 lote cantidad)	16
Figura 47. Nodo función GET LOTE CANTIDAD (tipo 2 lote cantidad).	16
Figura 48. Nodo función GET LOTE CANTIDAD (tipo 3 lote cantidad).	17
Figura 49. Nodo función GET (guardado de lote cantidad)	17
Figura 50. Nodo función GET (retenedores ocupados Profibus).	17
Figura 51. Nodo función GET AUTO FIN (tipo 1 auto finalizado)	17
Figura 52. Nodo función GET AUTO FIN (tipo 2 auto finalizado).	18
Figura 53. Nodo función GET AUTO FIN (tipo 3 auto finalizado)	18
Figura 54. Nodo función GET AUTO (led).....	18
Figura 55. Nodo función GET LOTE (led).	18
Figura 56. Nodo función toJSON (auto finalizado).	19
Figura 57. Nodo función SET CLICK (cancelar tipo 1, tipo 2, tipo 3, todos).	19
Figura 58. Nodo función SET CLICK (ciclo automático).	20
Figura 59. Nodo función SET CLICK (ciclo lote).	20
Figura 60. Nodo función SET CLICK (cancelar automático).	20
Figura 61. Nodo función SET DATA (cancelar tipo 1).	21
Figura 62. Nodo función SET DATA (cancelar tipo 2).	21
Figura 63. Nodo función SET DATA (cancelar tipo 3).	22
Figura 64. . Nodo función SET DATA (cancelar todos).	22
Figura 65. Nodo función SET DATA (ciclo automático).	23
Figura 66. Nodo función SET DATA (ciclo lote).	23
Figura 67. Nodo función SET DATA (cancelar automático).	24
Figura 68. Nodo función FIND ERROR.....	24
Figura 69. Nodo función ERROR.	25
Figura 70. Nodo función CORRECT.	25
Figura 71. Nodo función SET CAMBIO.	25
Figura 72. Nodo función GET TIPO (tipo 1).	26
Figura 73. Nodo función GET TIPO (tipo 2).	26
Figura 74. Nodo función GET TIPO (tipo 3).	26
Figura 75. Nodo función SET DATA (cambio flujo de producción).	27
Figura 76. Nodo función SET DATA (finalizado cambio flujo).	27
Figura 77. Nodo función SET DATA (abastecer materias).	28
Figura 78. Nodo función SET DATA (cantidad).	28
Figura 79. Flujo Pulmón.....	29
Figura 80. Nodo inject (json lectura PLC CAN).	29
Figura 81. Nodo función ID BANDEJA DESCARGADA.	29
Figura 82. Nodo función ID BANDEJAS ALMACENADAS.	30
Figura 83. Nodo función RESET PULMÓN.	30
Figura 84. Nodo función DIRECCIÓN (descarga manual serie).	30
Figura 85. Nodo función DIRECCIÓN (bandejas serie).	31
Figura 86. Nodo función PASOS RESET.	31
Figura 87. Nodo función PILA.	31
Figura 88. Nodo template PILA 3 [1].	32
Figura 89. Nodo template PILA 3 [2].	32
Figura 90. Nodo template PILA 3 [3].	33
Figura 91. Nodo template PILA 2 [1].	33
Figura 92. Nodo template PILA 2 [2].	34
Figura 93. Nodo template PILA 2 [3].	34

Figura 94. Nodo template PILA 1 [1].	35
Figura 95. Nodo template PILA 1 [2].	35
Figura 96. Nodo template PILA 1 [3].	36
Figura 97. Flujo Retenedores [1].	37
Figura 98. Flujo Retenedores [2].	37
Figura 99. Flujo Retenedores [3].	38
Figura 100. Flujo Retenedores [3].	38
Figura 101. Flujo Retenedores [4].	39
Figura 102. Nodo función REST DIR03.	39
Figura 103. Nodo función REST DIR04.	39
Figura 104. Nodo función REST DIR05.	39
Figura 105. Nodo función REST DIR06.	40
Figura 106. Nodo función REST DIR07.	40
Figura 107. Nodo función REST DIR08.	40
Figura 108. Nodo función REST DIR09.	40
Figura 109. Nodo función REST DIR20.	41
Figura 110. Nodo función REST DIR21.	41
Figura 111. Nodo función REST PT04.	41
Figura 112. Nodo función REST PT05.	41
Figura 113. Nodo función REST PT06.	42
Figura 114. Nodo función REST PT07.	42
Figura 115. Nodo función REST PT15.	42
Figura 116. Nodo función REST PT16.	42
Figura 117. Nodo función REST PT17.	43
Figura 118. Nodo función READY DIR03.	43
Figura 119. Nodo función READY DIR04.	43
Figura 120. Nodo función READY DIR05.	43
Figura 121. Nodo función READY DIR06.	44
Figura 122. Nodo función READY DIR07.	44
Figura 123. Nodo función READY DIR08.	44
Figura 124. Nodo función READY DIR09.	44
Figura 125. Nodo función READY DIR20.	45
Figura 126. Nodo función READY DIR21.	45
Figura 127. Nodo función READY PT04.	45
Figura 128. Nodo función READY PT05.	45
Figura 129. Nodo función READY PT06.	46
Figura 130. Nodo función READY PT07.	46
Figura 131. Nodo función READY PT15.	46
Figura 132. Nodo función READY PT16.	47
Figura 133. Nodo función READY PT17.	47
Figura 134. Nodo función MOVE DIR03.	47
Figura 135. Nodo función MOVE DIR04.	47
Figura 136. Nodo función MOVE DIR05.	48
Figura 137. Nodo función MOVE DIR06.	48
Figura 138. Nodo función MOVE DIR07.	48
Figura 139. Nodo función MOVE DIR08.	48
Figura 140. Nodo función MOVE DIR09.	49
Figura 141. Nodo función MOVE DIR20.	49

Figura 142. Nodo función MOVE DIR21	49
Figura 143. Nodo función MOVE PT04.....	50
Figura 144. Nodo función MOVE RECTO PT05.....	50
Figura 145. Nodo función MOVE DESVIO PT05.....	50
Figura 146. Nodo función MOVE PT06.....	50
Figura 147. Nodo función MOVE PT07.....	51
Figura 148. Nodo función MOVE PT15.....	51
Figura 149. Nodo función MOVE RECTO PT16.....	51
Figura 150. Nodo función MOVE DESVIO PT16.....	52
Figura 151. Nodo función MOVE PT17.....	52
Figura 152. Nodo función ID DIR03.....	52
Figura 153. Nodo función ID DIR04.....	53
Figura 154. Nodo función ID DIR05.....	53
Figura 155. Nodo función ID DIR06.....	53
Figura 156. Nodo función ID DIR07.....	53
Figura 157. Nodo función ID DIR08.....	54
Figura 158. Nodo función ID DIR09.....	54
Figura 159. Nodo función ID DIR20.....	54
Figura 160. Nodo función ID DIR21.....	54
Figura 161. Nodo función ID PT04.....	55
Figura 162. Nodo función ID PT05.....	55
Figura 163. Nodo función ID PT06.....	55
Figura 164. Nodo función ID PT07.....	55
Figura 165. Nodo función ID PT15.....	56
Figura 166. Nodo función ID PT16.....	56
Figura 167. Nodo función ID PT17.....	56
Figura 168. Nodo función TIPO DIR03.....	56
Figura 169. Nodo función TIPO DIR04.....	57
Figura 170. Nodo función TIPO DIR05.....	57
Figura 171. Nodo función TIPO DIR06.....	57
Figura 172. Nodo función TIPO DIR07.....	57
Figura 173. Nodo función TIPO DIR08.....	58
Figura 174. Nodo función TIPO DIR09.....	58
Figura 175. Nodo función TIPO DIR20.....	58
Figura 176. Nodo función TIPO DIR21.....	58
Figura 177. Nodo función TIPO PT04.....	58
Figura 178. Nodo función TIPO PT05.....	59
Figura 179. Nodo función TIPO PT06.....	59
Figura 180. Nodo función TIPO PT07.....	59
Figura 181. Nodo función TIPO PT15.....	59
Figura 182. Nodo función TIPO PT16.....	59
Figura 183. Nodo función TIPO PT17.....	60
Figura 184. Nodo función BANDEJA DIR03.....	60
Figura 185. Nodo función BANDEJA DIR04.....	60
Figura 186. Nodo función BANDEJA DIR05.....	60
Figura 187. Nodo función BANDEJA DIR06.....	61
Figura 188. Nodo función BANDEJA DIR07.....	61
Figura 189. Nodo función BANDEJA DIR08.....	61

Figura 190. Nodo función BANDEJA DIR09.	61
Figura 191. Nodo función BANDEJA DIR20.	61
Figura 192. Nodo función BANDEJA DIR21.	62
Figura 193. Nodo función BANDEJA PT04.	62
Figura 194. Nodo función BANDEJA PT05.	62
Figura 195. Nodo función BANDEJA PT06.	62
Figura 196. Nodo función BANDEJA PT07.	62
Figura 197. Nodo función BANDEJA PT15.	63
Figura 198. Nodo función BANDEJA PT16.	63
Figura 199. Nodo función BANDEJA PT17.	63
Figura 200. Nodo función ESTADO DIR04.	63
Figura 201. Nodo función ESTADO PT06.	64
Figura 202. Nodo función toJASON (estado movimiento retenedores.)	65
Figura 203. Nodo función toJASON (estado movimiento plataformas).....	66

ANEXO III: SOFTWARE NODE-RED

1. FLUJOS

1.1. FLUJO GENERAL

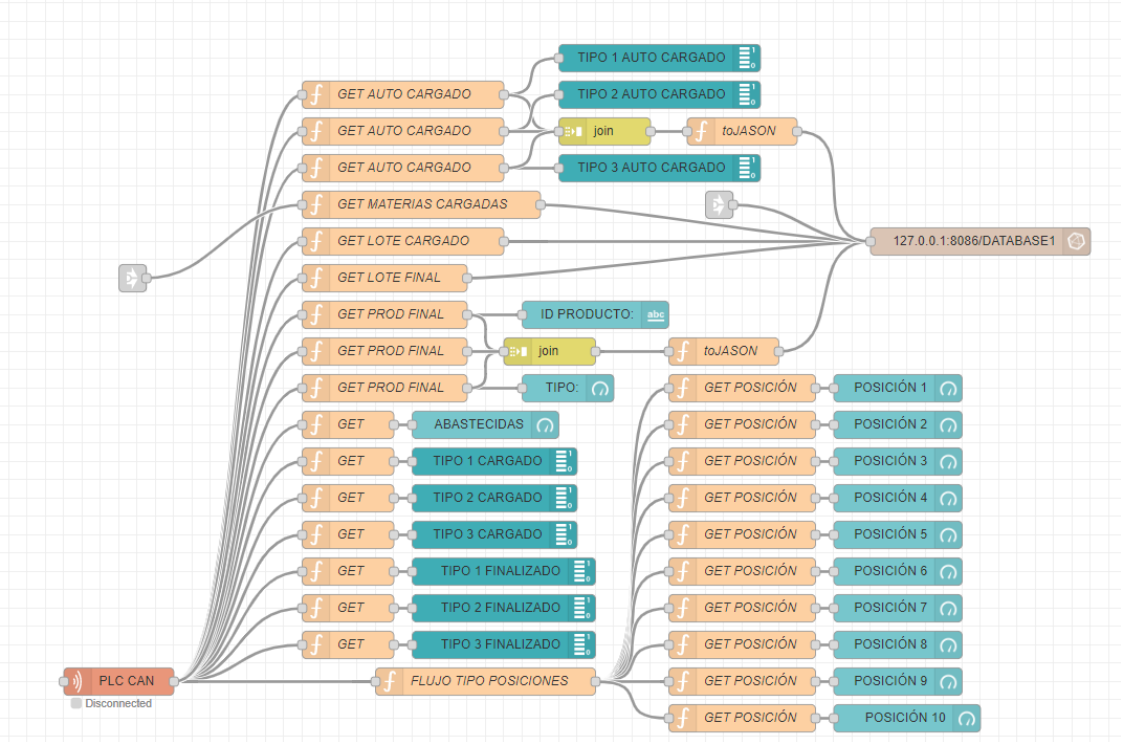


Figura 1. Flujo General [1].

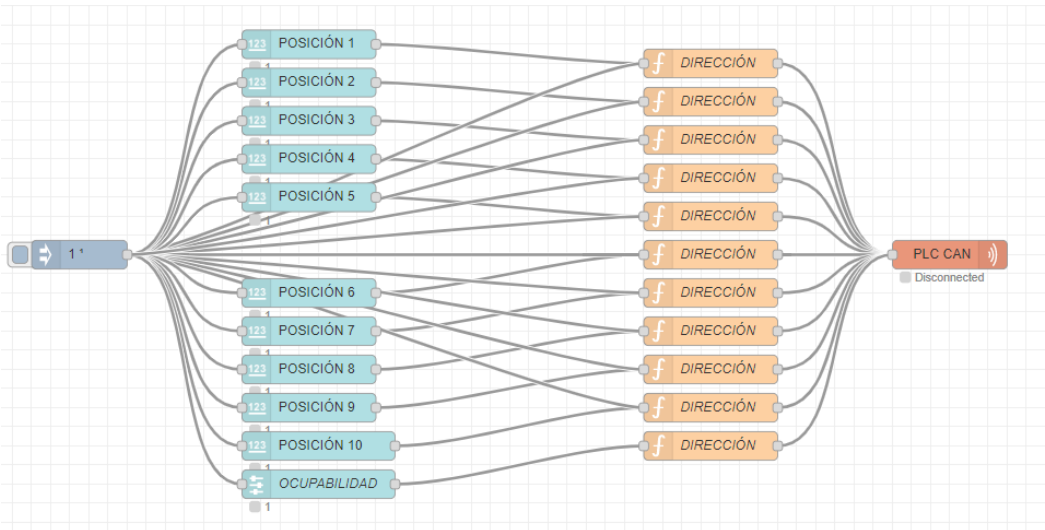


Figura 2. Flujo General [2].

```
1 let word_cantidad=msg.payload;
2 msg.payload=word_cantidad[4];
3 msg.topic="TIPO1";
4 return msg;
5
6
```

Figura 3. Nodo función GET AUTO CARGADO [1].

```
1 let word_cantidad=msg.payload;
2 msg.payload=word_cantidad[5];
3 msg.topic="TIPO2";
4 return msg;
```

Figura 4. Nodo función GET AUTO CARGADO [2].

```
1 let word_cantidad=msg.payload;
2 msg.payload=word_cantidad[6];
3 msg.topic="TIPO3";
4 return msg;
```

Figura 5. Nodo función GET AUTO CARGADO [3].

Name GET MATERIAS CARGADAS

Function

```
1 let cantidad= global.get("MATERIAS");
2 msg.payload=[
3   {
4     measurement: "MateriasCargadas",
5     fields: {
6       Materias: cantidad
7     }
8   }
9 ]
10 return msg;
11
12
```

Figura 6. Nodo función GET MATERIAS CARGADAS.

Name GET LOTE CARGADO

Function

```
1 let lote, estado, tipo1, tipo2, tipo3;
2
3 estado=msg.payload[7];
4 tipo1=msg.payload[8];
5 tipo2=msg.payload[9];
6 tipo3=msg.payload[10];
7
8 if(estado===1){
9   msg.payload=[
10    {
11      measurement: "LoteCargado",
12      fields: {
13        Tipo1:tipo1,
14        Tipo2:tipo2,
15        Tipo3:tipo3
16      }
17    }
18   ];
19
20   return msg;
21 }
22
```

Figura 7. Nodo Función GET LOTE CARGADO.

Name GET LOTE FINAL

Function

```

1 let lote, estado, tipo1, tipo2, tipo3;
2
3 estado=msg.payload[11];
4 tipo1=msg.payload[12];
5 tipo2=msg.payload[13];
6 tipo3=msg.payload[14];
7
8 if(estado===1){
9     msg.payload=[
10        {
11            measurement: "LoteFinalizado",
12            fields: {
13                Tipo1:tipo1,
14                Tipo2:tipo2,
15                Tipo3:tipo3
16            }
17        }
18    ];
19
20     return msg;
21 }
22

```

Figura 8. Nodo función GET LOTE FINAL.

Name GET PROD FINAL

Function

```

1 let word_cantidad=msg.payload;
2 msg.payload=word_cantidad[0];
3 msg.topic= "ID";
4 return msg;

```

Figura 9. Nodo función GET PROD FINAL [1].

Name GET PROD FINAL

Function

```

1 let word_cantidad=msg.payload;
2 msg.payload=word_cantidad[1];
3 msg.topic= "TIPO";
4 return msg;

```

Figura 10. Nodo función GET PROD FINAL [2].



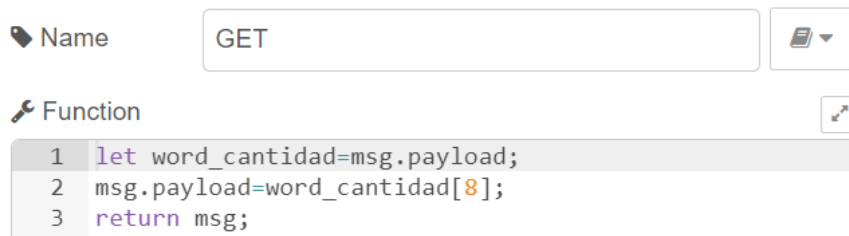
```
1 let word_cantidad=msg.payload;
2 msg.payload=word_cantidad[2];
3 msg.topic= "BANDEJA";
4 return msg;
```

Figura 11. Nodo función GET PROD FINAL [3].



```
1 let word_cantidad=msg.payload;
2 msg.payload=word_cantidad[3];
3 return msg;
4
5
```

Figura 12. Nodo función GET (materias actuales abastecidas en la línea)



```
1 let word_cantidad=msg.payload;
2 msg.payload=word_cantidad[8];
3 return msg;
```

Figura 13. Nodo función GET (tipo cargado 1).



```
1 let word_cantidad=msg.payload;
2 msg.payload=word_cantidad[9];
3 return msg;
```

Figura 14. Node función GET (tipo cargado 2)

```
1 let word_cantidad=msg.payload;
2 msg.payload=word_cantidad[10];
3 return msg;
```

Figura 15. Nodo función GET (tipo 3 cargado).

```
1 let word_cantidad=msg.payload;
2 msg.payload=word_cantidad[12];
3 return msg;
```

Figura 16. Nodo función GET (tipo 1 finalizado).

```
1 let word_cantidad=msg.payload;
2 msg.payload=word_cantidad[13];
3 return msg;
```

Figura 17. Nodo función GET (tipo 2 finalizado).

```
1 let word_cantidad=msg.payload;
2 msg.payload=word_cantidad[14];
3 return msg;
```

Figura 18. Nodo función GET (tipo 3 finalizado)

Name

Function

```
1 let flujo= [];  
2 for(let i=0; i<10; i++){  
3     flujo[i]=msg.payload[i+16];  
4 }  
5 global.set("FLUJO",flujo );  
6 return msg;  
7  
8
```

Figura 19. Nodo función FLUJO TIPO POSICIONES.

Name

Function

```
1 let posicion;  
2 posicion= msg.payload[19];  
3 msg.payload=posicion;  
4 return msg;  
5  
6
```

Figura 20. Nodo función GET POSICION (posición 1)

Name

Function

```
1 let posicion;  
2 posicion= msg.payload[20];  
3 msg.payload=posicion;  
4 return msg;
```

Figura 21. Nodo función GET POSICION (posición 2)



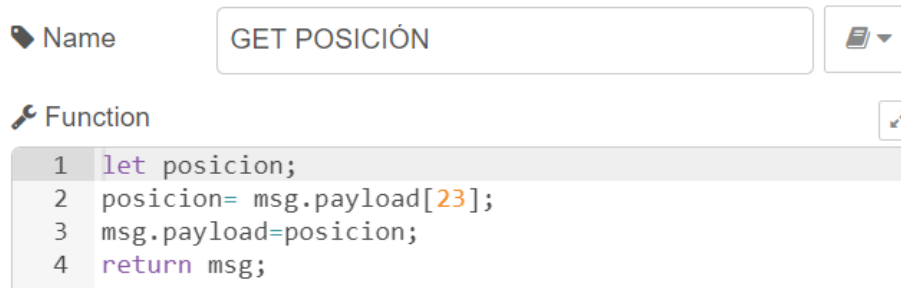
```
1 let posicion;
2 posicion= msg.payload[21];
3 msg.payload=posicion;
4 return msg;
```

Figura 22. Nodo función GET POSICION (posición 3)



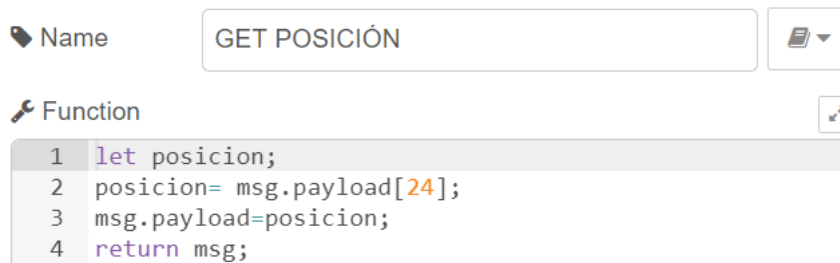
```
1 let posicion;
2 posicion= msg.payload[22];
3 msg.payload=posicion;
4 return msg;
```

Figura 23. Nodo función GET POSICION (posición 4)



```
1 let posicion;
2 posicion= msg.payload[23];
3 msg.payload=posicion;
4 return msg;
```

Figura 24. Nodo función GET POSICION (posición 5).



```
1 let posicion;
2 posicion= msg.payload[24];
3 msg.payload=posicion;
4 return msg;
```

Figura 25. Nodo función GET POSICION (posición 6).

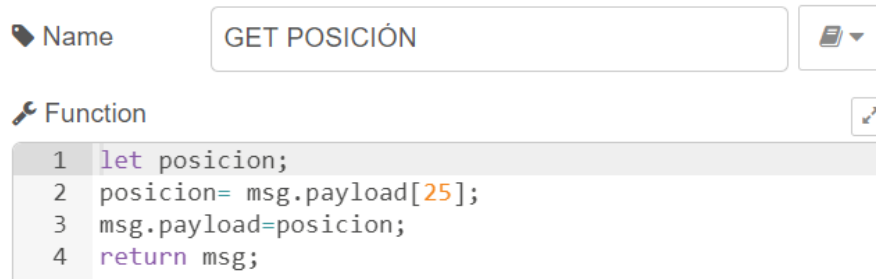


Figura 26. Nodo función GET POSICION (posición 7)

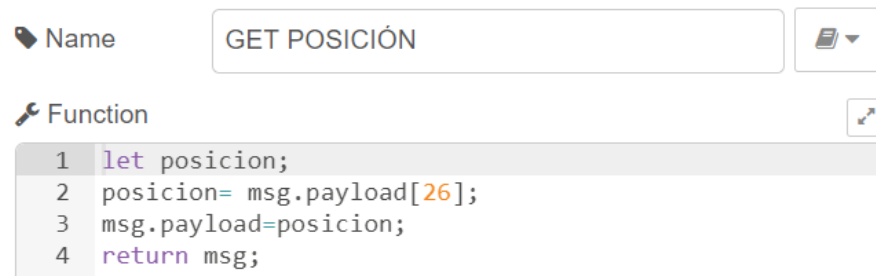


Figura 27. Nodo función GET POSICION (posición 8).

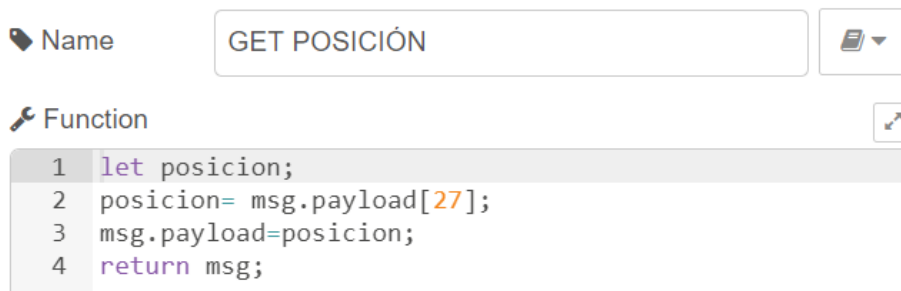


Figura 28. Nodo función GET POSICION (posición 9).

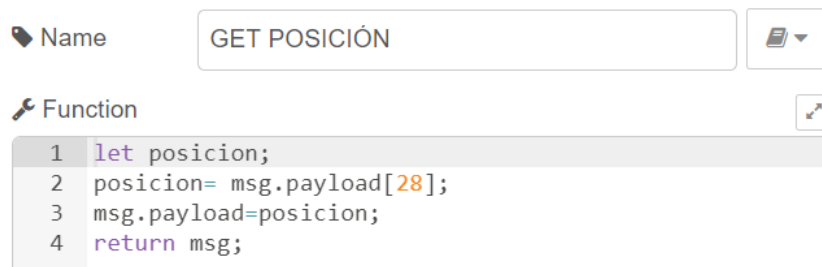


Figura 29. Nodo función GET POSICION (posición 10).

Name

Function

```

1 let tipo1=msg.payload["TIPO1"];
2 let tipo2= msg.payload["TIPO2"];
3 let tipo3=msg.payload["TIPO3"];
4
5 msg.payload=[
6   {
7     measurement: "AutoCargado",
8     fields: {
9       TIPO1: tipo1,
10      TIPO2: tipo2,
11      TIPO3: tipo3
12    }
13  }
14 ]
return msg;

```

Figura 30. Nodo toJSON (auto cargado).

Name

Function

```

1 let id=msg.payload["ID"];
2 let tipo= msg.payload["TIPO"];
3 let bandeja=msg.payload["BANDEJA"];
4 let automatico= global.get("CICLO_AUTOMATICO");
5 let lote= global.get("CICLO_LOTE");
6
7 if(auto automatico || lote){
8   msg.payload=[
9     {
10      measurement: "UltimoProducto",
11      fields: {
12        ID: id,
13        TIPO: tipo,
14        BANDEJA: bandeja
15      }
16    }
17  ]
18 }
return msg;

```

Figura 31. Nodo toJSON (producto finalizado).



```
1 let cambio=global.get("CAMBIO");
2
3 if(cambio===true){
4   msg.datatype=6;
5   msg.address=1213;
6   return msg;
7 }
8
9
```

Figura 32. Nodo función DIRECCIÓN (posición 1)



```
1 letcambio=global.get("CAMBIO");
2 if(cambio===true){
3   msg.datatype=6;
4   msg.address=1214;
5   return msg;
6 }
```

Figura 33. Nodo función DIRECCIÓN (posición 2)



```
1 let cambio=global.get("CAMBIO");
2
3 if(cambio===true){
4   msg.datatype=6;
5   msg.address=1215;
6   return msg;
7 }
8
```

Figura 34. Nodo función DIRECCIÓN (posición 3).

Name DIRECCIÓN

Function

```

1 let cambio=global.get("CAMBIO");
2
3
4 if(cambio===true){
5     msg.datatype=6;
6     msg.address=1216;
7     return msg;
8 }

```

Figura 35. Nodo función DIRECCIÓN (posición 4).

Name DIRECCIÓN

Function

```

1 let cambio=global.get("CAMBIO")
2
3 if(cambio===true){
4     msg.datatype=6;
5     msg.address=1217;
6     return msg;
7 }
8

```

Figura 36. Nodo función DIRECCIÓN (posición 4).

Name DIRECCIÓN

Function

```

1 let cambio=global.get("CAMBIO")
2
3 if(cambio===true){
4     msg.datatype=6;
5     msg.address=1218;
6     return msg;
7 }
8

```

Figura 37. Nodo función DIRECCIÓN (posición 5).



```
1 let cambio=global.get("CAMBIO")
2
3
4 if(cambio===true){
5   msg.datatype=6;
6   msg.address=1219;
7   return msg;
8 }
9
```

Figura 38. Nodo función DIRECCIÓN (posición 6).



```
1 let cambio=global.get("CAMBIO")
2
3 if(cambio===true){
4   msg.datatype=6;
5   msg.address=1220;
6   return msg;
7 }
8
```

Figura 39. Nodo función DIRECCIÓN (posición 7).



```
1 let cambio=global.get("CAMBIO");
2
3 if(cambio===true) {
4   msg.datatype=6;
5   msg.address=1221;
6   return msg;
7 }
8
```

Figura 40. Nodo función DIRECCIÓN (posición 8).



```
1 let cambio=global.get("CAMBIO");
2
3 if(cambio===true){
4   msg.datatype=6;
5   msg.address=1222;
6   return msg;
7 }
8
```

Figura 41. Nodo función DIRECCIÓN (posición 9).



```
1 msg.datatype=6;
2 msg.address=1225;
3 return msg;
4
```

Figura 42. Nodo función DIRECCIÓN (posición 10).

1.2. FLUJO CICLOS

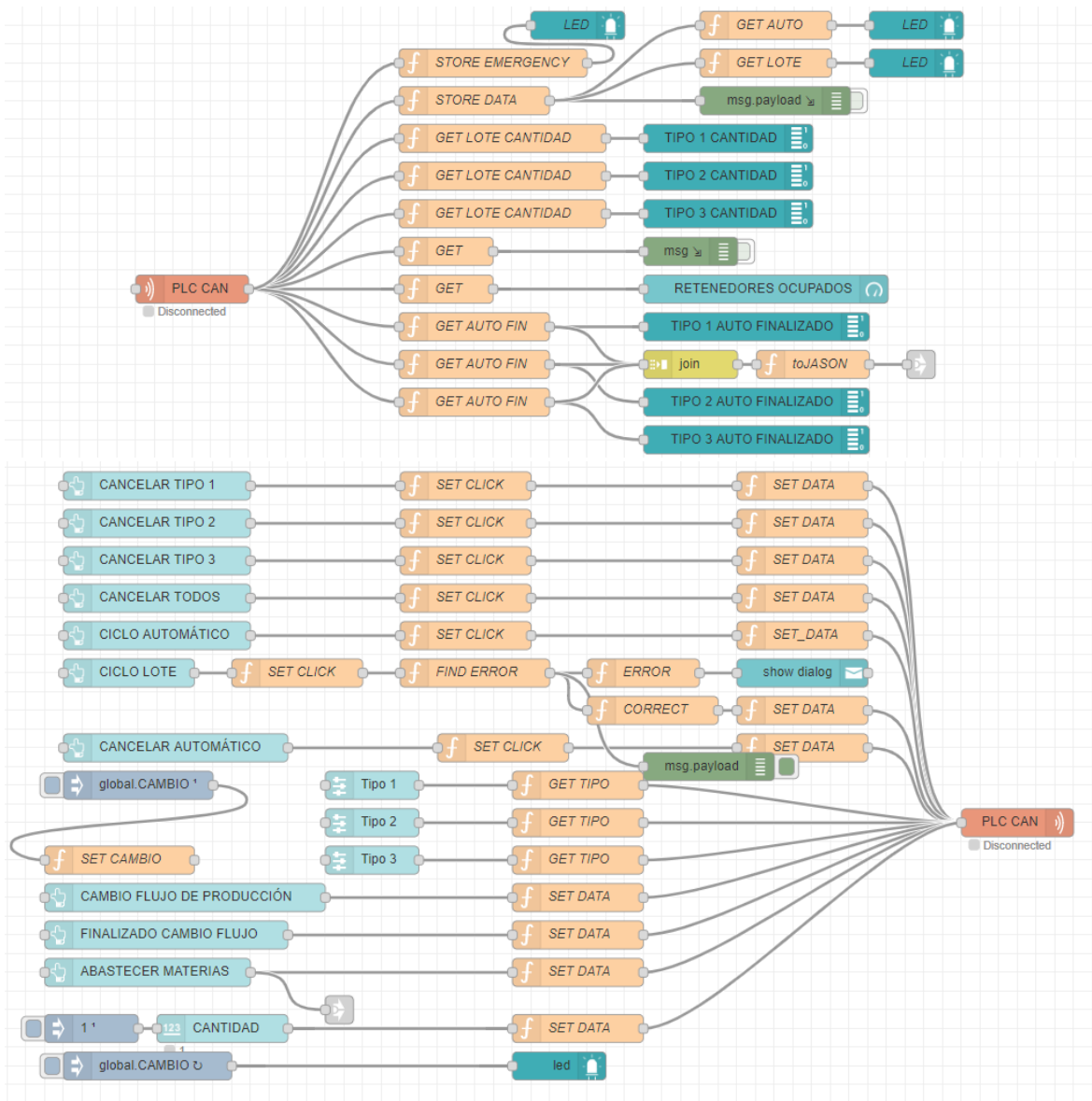


Figura 43. Flujo Ciclos.

Name

Function

```

1 let emergency = msg.payload[11];
2 global.set("EMERGENCY", emergency);
3 msg.payload=emergency;
4 return msg;

```

Figura 44. Nodo función STORE EMERGENCY.

Name

Function

```

1 let recived, word_MW1223=[];
2 recived=msg.payload[0];
3
4 for(let i = 0; i < 16; i++){
5   if((recived & Math.pow(2,i)) >0){
6     word_MW1223[i]=1;}
7   else word_MW1223[i]=0;}
8
9 flow.set("MW1223", word_MW1223);
10 msg.payload = word_MW1223;
11 return msg;

```

Figura 45. Nodo función STORE DATA.

Name

Function

```

1 let lote,word_cantidad=msg.payload;
2 msg.payload=word_cantidad[4];
3 return msg;

```

Figura 46. Nodo función GET LOTE CANTIDAD (tipo 1 lote cantidad)

Name

Function

```

1 let lote, word_cantidad=msg.payload;
2 msg.payload=word_cantidad[5];
3 return msg;

```

Figura 47. Nodo función GET LOTE CANTIDAD (tipo 2 lote cantidad).

Name GET LOTE CANTIDAD

Function

```

1 let lote, word_cantidad=msg.payload;
2 msg.payload=word_cantidad[6];
3 return msg;

```

Figura 48. Nodo función GET LOTE CANTIDAD (tipo 3 lote cantidad).

Name GET

Function

```

1 let lote=[];
2
3 for(let i=0; i<3; i++){
4     lote[i]=msg.payload[4+i];
5 }
6 global.set("LOTE_CANTIDAD", lote);
7
8 msg.payload=lote;
9 return msg;

```

Figura 49. Nodo función GET (guardado de lote cantidad)

Name GET

Function

```

1 let word_cantidad=msg.payload;
2 msg.payload=word_cantidad[7];
3 return msg;

```

Figura 50. Nodo función GET (retenedores ocupados Profibus).

Name GET AUTO FIN

Function

```

1 let word_cantidad=msg.payload;
2 msg.payload=word_cantidad[8];
3 msg.topic="TIPO1";
4 return msg;

```

Figura 51. Nodo función GET AUTO FIN (tipo 1 auto finalizado)



```

1 let word_cantidad=msg.payload;
2 msg.payload=word_cantidad[9];
3 msg.topic="TIPO2";
4 return msg;

```

Figura 52. Nodo función GET AUTO FIN (tipo 2 auto finalizado).




```

1 let word_cantidad=msg.payload;
2 msg.payload=word_cantidad[10];
3 msg.topic="TIPO3";
4 return msg;

```

Figura 53. Nodo función GET AUTO FIN (tipo 3 auto finalizado)

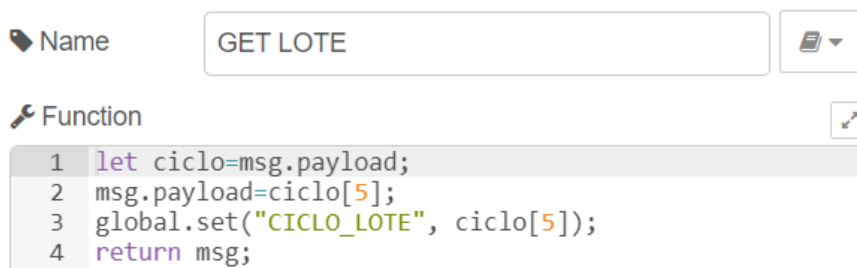


```

1 let ciclo=msg.payload;
2 msg.payload=ciclo[3];
3 global.set("CICLO_AUTOMATICO", ciclo[3]);
4 return msg;

```

Figura 54. Nodo función GET AUTO (led)



```

1 let ciclo=msg.payload;
2 msg.payload=ciclo[5];
3 global.set("CICLO_LOTE", ciclo[5]);
4 return msg;

```

Figura 55. Nodo función GET LOTE (led).

Name

Function

```

1 let tipo1=msg.payload["TIPO1"];
2 let tipo2= msg.payload["TIPO2"];
3 let tipo3=msg.payload["TIPO3"];
4
5 let automatico= global.get("CICLO_AUTOMATICO");
6 let lote= global.get("CICLO_LOTE");
7
8 if(automatico){
9 msg.payload=[
10 {
11     measurement: "AutoFinalizado",
12     fields: {
13         TIPO1: tipo1,
14         TIPO2: tipo2,
15         TIPO3: tipo3
16     }
17 } ]
18 return msg;
19 }

```

Figura 56. Nodo función toJSON (auto finalizado).

Name

Function

```

1 let ciclo;
2 ciclo= flow.get("MW1223")[5];
3 if(ciclo===1){
4     return msg;
5 }

```

Figura 57. Nodo función SET CLICK (cancelar tipo 1, tipo 2, tipo 3, todos).



The screenshot shows a Node-RED function node named 'SET CLICK'. The function code is as follows:

```
1 let ciclo;  
2 ciclo= flow.get("MW1223")[5];  
3 if( ciclo===0){  
4     return msg;  
5 }  
6  
7
```

Figura 58. Nodo función SET CLICK (ciclo automático).



The screenshot shows a Node-RED function node named 'SET CLICK'. The function code is as follows:

```
1 let ciclo;  
2 ciclo= flow.get("MW1223")[3];  
3 if( ciclo===0){  
4     return msg;  
5 }
```

Figura 59. Nodo función SET CLICK (ciclo lote).



The screenshot shows a Node-RED function node named 'SET CLICK'. The function code is as follows:

```
1 let ciclo;  
2 ciclo= flow.get("MW1223")[3];  
3 if(ciclo===1 ){  
4     return msg;  
5 }  
6  
7
```

Figura 60. Nodo función SET CLICK (cancelar automático).

Name

Function

```

1 let word_MW1223, num_MW1223 = 0;
2
3 word_MW1223= flow.get("MW1223");
4 word_MW1223[6]=1;
5
6 flow.set("MW1223",word_MW1223);
7
8 for(let i=0; i<word_MW1223.length; i++){
9     if (word_MW1223[i]===1){
10         num_MW1223+=Math.pow(2,i);}}
11
12 msg.datatype = 6;
13 msg.address = 1223;
14 msg.payload=num_MW1223;
15
16 return msg;

```

Figura 61. Nodo función SET DATA (cancelar tipo 1).

Name

Function

```

1 let word_MW1223, num_MW1223 = 0;
2
3 word_MW1223= flow.get("MW1223");
4
5 word_MW1223[7]=1;
6
7 flow.set("MW1223",word_MW1223);
8
9 for(let i=0; i<word_MW1223.length; i++){
10     if (word_MW1223[i]===1){
11         num_MW1223+=Math.pow(2,i);
12     }
13 }
14
15 msg.datatype = 6;
16 msg.address = 1223;
17 msg.payload=num_MW1223;
18
19 return msg;

```

Figura 62. Nodo función SET DATA (cancelar tipo 2).

Name

Function

```

1 let word_MW1223, num_MW1223=0;
2
3 word_MW1223= flow.get("MW1223");
4 word_MW1223[8]=1;
5 flow.set("MW1223",word_MW1223);
6
7 for(let i=0; i<word_MW1223.length; i++){
8     if (word_MW1223[i]===1){
9         num_MW1223+=Math.pow(2,i);
10    }
11 }
12
13 msg.datatype=6;
14 msg.address=1223;
15 msg.payload=num_MW1223;
16
17 return msg;

```

Figura 63. Nodo función SET DATA (cancelar tipo 3).

Name

Function

```

1 let word_MW1223, num_MW1223=0;
2
3 word_MW1223= flow.get("MW1223");
4 word_MW1223[9]=1;
5 flow.set("MW1223",word_MW1223);
6
7 for(let i=0; i<word_MW1223.length; i++){
8     if (word_MW1223[i]===1){
9         num_MW1223+=Math.pow(2,i);
10    }
11 }
12
13 msg.datatype = 6;
14 msg.address = 1223;
15 msg.payload=num_MW1223;
16
17 return msg;

```

Figura 64. . Nodo función SET DATA (cancelar todos).

Name

Function

```

1 let word_MW1223, num_MW1223 = 0;
2
3 word_MW1223= flow.get("MW1223");
4
5 word_MW1223[3]=1;
6
7 flow.set("MW1223",word_MW1223);
8
9 for(let i=0; i<word_MW1223.length; i++){
10     if (word_MW1223[i]===1){
11         num_MW1223+=Math.pow(2,i);
12     }
13 }
14
15 msg.datatype = 6;
16 msg.address = 1223;
17 msg.payload=num_MW1223;
18
19 return msg;

```

Figura 65. Nodo función SET DATA (ciclo automático).

Name

Function

```

1 let word_MW1223, num_MW1223=0;
2
3 word_MW1223= flow.get("MW1223");
4 word_MW1223[5]=1;
5 flow.set("MW1223",word_MW1223);
6
7 for(let i=0; i<word_MW1223.length; i++){
8     if (word_MW1223[i]===1){
9         num_MW1223+= Math.pow(2,i);
10    }
11 }
12
13 msg.datatype=6;
14 msg.address=1223;
15 msg.payload=num_MW1223;
16
17 return msg;

```

Figura 66. Nodo función SET DATA (ciclo lote).

Name

Function

```

1 let word_MW1223, num_MW1223=0;
2
3 word_MW1223= flow.get("MW1223");
4 word_MW1223[4]=1;
5 flow.set("MW1223",word_MW1223);
6
7 for(let i=0; i<word_MW1223.length; i++){
8     if (word_MW1223[i]==1){
9         num_MW1223+= Math.pow(2,i);
10    }
11 }
12
13 msg.datatype = 6;
14 msg.address = 1223;
15 msg.payload=num_MW1223;
16
17 return msg;

```

Figura 67. Nodo función SET DATA (cancelar automático).

Name

Function

```

1 //Error 0: No hay ningún error.
2 //      1: Falta poner en el Flujo un tipo 1;
3 //      2: Falta poner en el Flujo un tipo 2;
4 //      3: Falta poner en el Flujo un tipo 3;
5 //      4: El lote está vacío;
6
7 let lote=[], flujo=[], error=[0,0,0,0];
8
9 lote= global.get("LOTE_CANTIDAD");
10 flujo=global.get("FLUJO");
11
12 if(lote[0]>0 & !flujo.includes(1)){
13     error[1]=1; error[0]=1}
14 if(lote[1]>0 & !flujo.includes(2) ){
15     error[2]=1;error[0]=1}
16 if(lote[2]>0 & !flujo.includes(3) ){
17     error[3]=1;error[0]=1}
18
19 if(lote[0]===0 & lote[1]===0 & lote[2]===0 ){
20     error[4]=1;
21     error[0]=1;
22 }
23 msg.payload=error;
24 return msg;

```

Figura 68. Nodo función FIND ERROR.

Name

Function

```

1 let msg_error="";
2
3 if(msg.payload[1]===1){
4     msg_error+="Falta Producto Tipo 1 en FLUJOS.<br>";
5 }
6 if(msg.payload[2]===1){
7     msg_error+="Falta Producto Tipo 2 en FLUJOS.<br>";
8 }
9 if(msg.payload[3]===1){
10    msg_error+="Falta Producto Tipo 3 en FLUJOS.<br>";
11 }
12 if(msg.payload[4]===1){
13    msg_error+="El lote está vacío.<br>";
14 }
15
16 msg.payload=msg_error;
17 return msg;

```

Figura 69. Nodo función ERROR.

Name

Function

```

1 if(msg.payload[0]===0){
2
3     msg.payload=true;
4     return msg;}

```

Figura 70. Nodo función CORRECT.

Name

Function

```

1 global.set("CAMBIO", false);
2 return msg;

```

Figura 71. Nodo función SET CAMBIO.

Name GET TIPO

Function

```

1 let ciclo_lote = flow.get("MW1223")[5];
2
3 msg.datatype = 6;
4 msg.address = 1227;
5
6 if(ciclo_lote === 0){
7     return msg;
8 }

```

Figura 72. Nodo función GET TIPO (tipo 1).

Name GET TIPO

Function

```

1 let ciclo_lote = flow.get("MW1223")[5];
2
3 msg.datatype = 6;
4 msg.address = 1228;
5
6 if(ciclo_lote === 0){
7     return msg;
8 }

```

Figura 73. Nodo función GET TIPO (tipo 2).

Name GET TIPO

Function

```

1 let ciclo_lote = flow.get("MW1223")[5];
2
3 msg.datatype = 6;
4 msg.address = 1229;
5
6 if(ciclo_lote === 0){
7     return msg;
8 }

```

Figura 74. Nodo función GET TIPO (tipo 3).

Name

Function

```

1 let word_MW1223, num_MW1223=0;
2
3 word_MW1223= flow.get("MW1223");
4 word_MW1223[0]=1;
5 flow.set("MW1223",word_MW1223);
6
7 for(let i=0; i<word_MW1223.length; i++){
8     if (word_MW1223[i]==1){
9         num_MW1223+= Math.pow(2,i);
10    }
11 }
12
13 global.set("CAMBIO",true);
14
15 msg.datatype = 6;
16 msg.address = 1223;
17 msg.payload=num_MW1223;
18
19 return msg;

```

Figura 75. Nodo función SET DATA (cambio flujo de producción).

Name

Function

```

1 let word_MW1223, num_MW1223=0;
2
3 word_MW1223= flow.get("MW1223");
4 word_MW1223[1]=1;
5 flow.set("MW1223",word_MW1223);
6 global.set("CAMBIO",false);
7 for(let i=0; i<word_MW1223.length; i++){
8     if (word_MW1223[i]==1){
9         num_MW1223+= Math.pow(2,i);
10    }
11 }
12
13 msg.datatype = 6;
14 msg.address = 1223;
15 msg.payload=num_MW1223;
16
17 return msg;

```

Figura 76. Nodo función SET DATA (finalizado cambio flujo).

Name

Function

```

1 let word_MW1223, num_MW1223=0;
2
3 word_MW1223= flow.get("MW1223");
4 word_MW1223[2]=1;
5 flow.set("MW1223",word_MW1223);
6
7 for(let i=0; i<word_MW1223.length; i++){
8     if (word_MW1223[i]==1){
9         num_MW1223+= Math.pow(2,i);
10    }
11 }
12
13 msg.datatype = 6;
14 msg.address = 1223;
15 msg.payload=num_MW1223;
16
17 return msg;

```

Figura 77. Nodo función SET DATA (abastecer materias).

Name

Function

```

1 msg.datatype = 6;
2 msg.address = 1224;
3 global.set("MATERIAS",msg.payload);
4 return msg;

```

Figura 78. Nodo función SET DATA (cantidad).

1.3. FLUJO PULMÓN

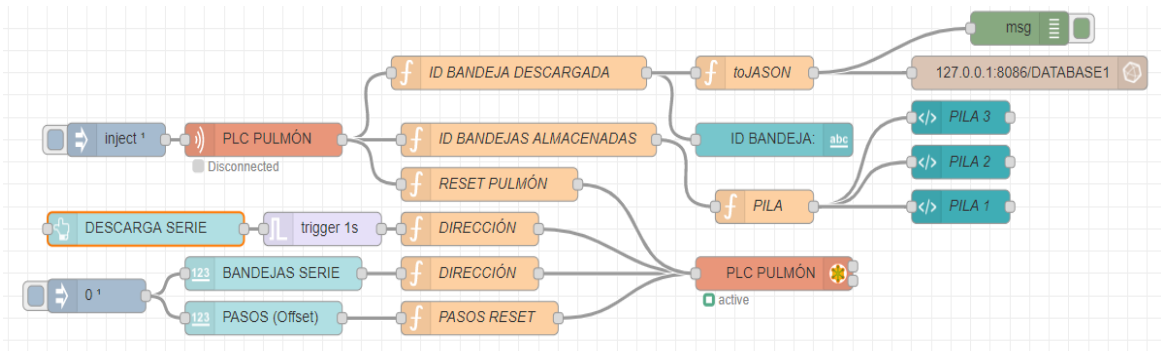


Figura 79. Flujo Pulmón.

Edit JSON
Visual editor

format JSON

```

1 [
2   {
3     "name": "Register1",
4     "address": 52,
5     "quantity": 4,
6     "dataType": "FC3",
7     "interval": 1000
8   },
9   {
10    "name": "Register2",
11    "address": 100,
12    "quantity": 31,
13    "dataType": "FC3",
14    "interval": 1000
15  }
16 ]

```

Figura 80. Nodo inject (json lectura PLC CAN).

Name

Function

```

1 if(msg.settings.name ==="Register1"){
2   msg.payload= msg.payload[0];
3   return msg;
4 }

```

Figura 81. Nodo función ID BANDEJA DESCARGADA.

Name

Function

```

1 if(msg.settings.name ==="Register2"){
2   return msg;
3 }

```

Figura 82. Nodo función ID BANDEJAS ALMACENADAS.

Name

Function

```

1 if(msg.settings.name ==="Register1"){
2   if(msg.payload[1]){
3     let array=[];
4
5     for(let i=0; i<30; i++){
6       array[i]=0;
7     }
8
9     msg.payload = {
10      value:array,
11      'fc':16,
12      'unitid': 1,
13      'address':101,
14      'quantity':30
15    }
16
17    // msg.payload=array
18    return msg;
19  }
20 }

```

Figura 83. Nodo función RESET PULMÓN.

Name

Function

```

1 msg.payload = {
2   value:msg.payload,
3   'fc':6,
4   'unitid': 1,
5   'address':99,
6   'quantity':1
7 }
8 return msg;

```

Figura 84. Nodo función DIRECCIÓN (descarga manual serie).

Name DIRECCIÓN

Function

```

1 msg.payload = {
2   value:msg.payload,
3   'fc':6,
4   'unitid': 1,
5   'address':98,
6   'quantity':1
7 }
8 return msg;

```

Figura 85. Nodo función DIRECCIÓN (bandejas serie).

Name PASOS RESET

Function

```

1 msg.payload = {
2   value:msg.payload,
3   'fc':6,
4   'unitid': 1,
5   'address':97,
6   'quantity':1
7 }
8 return msg;

```

Figura 86 Nodo función PASOS RESET.

Name PILA

Function

```

1 let recived, pila=[];
2 recived= msg.payload;
3
4 for(let i= 0; i<msg.payload.length; i++){
5   pila[i]=recived[i];
6
7   pila.shift();
8   msg.payload= pila;
9   return msg;

```

Figura 87. Nodo función PILA.

```

1 <style>
2   .pos{
3     display: flex;
4     flex-direction: row;
5     justify-content: center;
6     align-items: center;
7
8
9   }
10
11  .box{
12    display: flex;
13    justify-content: center;
14    align-items: center;
15    border: 1px solid rgb(146, 144, 255);
16    border-radius: 2rem;
17    flex:1;
18  }
19
20  #text{
21    flex:1;
22    text-align: center;
23  }
24
25  p{ text-align:center;
26     padding-right:1rem;
27     padding-left:1rem;
28     flex: 0.5;
29  }
30  #almacen{
31    display: flex;
32    flex-direction: column;
33    align-items: stretch;
34    justify-content: center;
35  }
36

```

Figura 88. Nodo template PILA 3 [1].

```

37  #head{
38
39    padding-bottom:0.5rem ;
40  }
41
42
43 </style>
44
45
46 <div id="almacen">
47
48   <div id="head">
49     <div class="pos">
50       <p> <font color="#85b4ff"> POSICIÓN </font> </p>
51       <div id="text"> <font color="#85b4ff"> ID BANDEJAS </font> </div>
52     </div>
53   </div>
54
55   <div class="pos">
56     <p>30: </p>
57     <div class="box" style="{{(msg.payload[29] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}" >{{msg.payload[29]}} </div>
58   </div>
59
60   <div class="pos">
61     <p>29: </p>
62     <div class="box" style="{{(msg.payload[28] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}" >{{msg.payload[28]}} </div>
63   </div>
64
65   <div class="pos">
66     <p>28: </p>
67     <div class="box" style="{{(msg.payload[27] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}" >{{msg.payload[27]}} </div>
68   </div>
69
70   <div class="pos">
71     <p>27: </p>
72     <div class="box" style="{{(msg.payload[26] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}" >{{msg.payload[26]}} </div>
73   </div>

```

Figura 89. Nodo template PILA 3 [2].

```

75 <div class="pos">
76   <p>26: </p>
77   <div class="box" style="{{(msg.payload[25] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}" >{{msg.payload[25]]} </div>
78 </div>
79
80 <div class="pos">
81   <p>25: </p>
82   <div class="box" style="{{(msg.payload[24] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}" >{{msg.payload[24]]} </div>
83 </div>
84
85 <div class="pos">
86   <p>24: </p>
87   <div class="box" style="{{(msg.payload[23] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}" >{{msg.payload[23]]} </div>
88 </div>
89
90 <div class="pos">
91   <p>23: </p>
92   <div class="box" style="{{(msg.payload[22] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}" >{{msg.payload[22]]} </div>
93 </div>
94
95 <div class="pos">
96   <p>22: </p>
97   <div class="box" style="{{(msg.payload[21] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}" >{{msg.payload[21]]} </div>
98 </div>
99
100 <div class="pos">
101   <p>21: </p>
102   <div class="box" style="{{(msg.payload[20] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}" >{{msg.payload[20]]} </div>
103 </div>
104
105 </div>

```

Figura 90. Nodo template PILA 3 [3].

```

1 <style>
2   .pos{
3     display: flex;
4     flex-direction: row;
5     justify-content: center;
6     align-items: center;
7
8
9   }
10
11   .box{
12     display: flex;
13     justify-content: center;
14     align-items: center;
15     border: 1px solid rgb(146, 144, 255);
16     border-radius: 2rem;
17     flex:1;
18   }
19
20   #text{
21     flex:1;
22     text-align: center;
23   }
24
25   p{ text-align:center;
26     padding-right:1rem;
27     padding-left:1rem;
28     flex: 0.5;
29   }
30   #almacen{
31     display: flex;
32     flex-direction: column;
33     align-items: stretch;
34     justify-content: center;
35   }
36

```

Figura 91. Nodo template PILA 2 [1].

```

37 ▾ #head{
38   padding-bottom:0.5rem ;
39 }
40 ▾
41
42 </style>
43
44 <div id="almacen">
45
46   <div id="head">
47     <div class="pos">
48       <p> <font color="#85b4ff"> POSICIÓN </font> </p>
49       <div id="text" <font color="#85b4ff"> ID BANDEJAS </font> </div>
50     </div>
51   </div>
52
53   <div class="pos">
54     <p>20: </p>
55     <div class="box" style="{{(msg.payload[19] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}" >{{msg.payload[19]}} </div>
56   </div>
57
58
59   <div class="pos">
60     <p>19: </p>
61     <div class="box" style="{{(msg.payload[18] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}" >{{msg.payload[18]}} </div>
62   </div>
63
64   <div class="pos">
65     <p>18: </p>
66     <div class="box" style="{{(msg.payload[17] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}" >{{msg.payload[17]}} </div>
67   </div>
68
69   <div class="pos">
70     <p>17: </p>
71     <div class="box" style="{{(msg.payload[16] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}" >{{msg.payload[16]}} </div>
72   </div>
73

```

Figura 92. Nodo template PILA 2 [2].

```

78
79   <div class="pos">
80     <p>15: </p>
81     <div class="box" style="{{(msg.payload[14] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}" >{{msg.payload[14]}} </div>
82   </div>
83
84   <div class="pos">
85     <p>14: </p>
86     <div class="box" style="{{(msg.payload[13] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}" >{{msg.payload[13]}} </div>
87   </div>
88
89   <div class="pos">
90     <p>13: </p>
91     <div class="box" style="{{(msg.payload[12] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}" >{{msg.payload[12]}} </div>
92   </div>
93
94   <div class="pos">
95     <p>12: </p>
96     <div class="box" style="{{(msg.payload[11] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}" >{{msg.payload[11]}} </div>
97   </div>
98
99   <div class="pos">
100     <p>11: </p>
101     <div class="box" style="{{(msg.payload[10] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}" >{{msg.payload[10]}} </div>
102   </div>
103 </div>
104
105

```

Figura 93. Nodo template PILA 2 [3].


```

1 <style>
2   .pos{
3     display: flex;
4     flex-direction: row;
5     justify-content: center;
6     align-items: center;
7
8
9   }
10
11  .box{
12    display: flex;
13    justify-content: center;
14    align-items: center;
15    border: 1px solid rgb(146, 144, 255);
16    border-radius: 2rem;
17    flex:1;
18  }
19
20  #text{
21    flex:1;
22    text-align: center;
23  }
24
25  p{ text-align:center;
26    padding-right:1rem;
27    padding-left:1rem;
28    flex: 0.5;
29  }
30  #almacen{
31    display: flex;
32    flex-direction: column;
33    align-items: stretch;
34    justify-content: center;
35  }
36

```

Figura 94. Nodo template PILA 1 [1].

```

37  #head{
38    padding-bottom:0.5rem ;
39  }
40
41
42
43 </style>
44
45
46 <div id="almacen">
47
48   <div id="head">
49     <div class="pos">
50       <p> <font color="#85b4ff"> POSICIÓN </font> </p>
51       <div id="text"> <font color="#85b4ff"> ID BANDEJAS </font> </div>
52     </div>
53   </div>
54
55   <div class="pos">
56     <p>9: </p>
57     <div class="box" style="{{(msg.payload[9] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}"> {{msg.payload[9]]} </div>
58   </div>
59
60   <div class="pos">
61     <p>9: </p>
62     <div class="box" style="{{(msg.payload[8] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}"> {{msg.payload[8]]} </div>
63   </div>
64
65   <div class="pos">
66     <p>8: </p>
67     <div class="box" style="{{(msg.payload[7] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}"> {{msg.payload[7]]} </div>
68   </div>
69
70   <div class="pos">
71     <p>7: </p>
72     <div class="box" style="{{(msg.payload[6] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}"> {{msg.payload[6]]} </div>
73   </div>

```

Figura 95. Nodo template PILA 1 [2].

```
74 |
75 |   <div class="pos">
76 |     <p>6: </p>
77 |     <div class="box" style="{{(msg.payload[5] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}" >{{msg.payload[5]}} </div>
78 |   </div>
79 |
80 |   <div class="pos">
81 |     <p>5: </p>
82 |     <div class="box" style="{{(msg.payload[4] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}" >{{msg.payload[4]}} </div>
83 |   </div>
84 |
85 |   <div class="pos">
86 |     <p>4: </p>
87 |     <div class="box" style="{{(msg.payload[3] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}" >{{msg.payload[3]}} </div>
88 |   </div>
89 |
90 |   <div class="pos">
91 |     <p>3: </p>
92 |     <div class="box" style="{{(msg.payload[2] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}" >{{msg.payload[2]}} </div>
93 |   </div>
94 |
95 |   <div class="pos">
96 |     <p>2: </p>
97 |     <div class="box" style="{{(msg.payload[1] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}" >{{msg.payload[1]}} </div>
98 |   </div>
99 |
100 |   <div class="pos">
101 |     <p>1: </p>
102 |     <div class="box" style="{{(msg.payload[0] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}" >{{msg.payload[0]}} </div>
103 |   </div>
104 |
105 | </div>
```

Figura 96. Nodo template PILA 1 [3].

1.4. FLUJO RETENEDORES

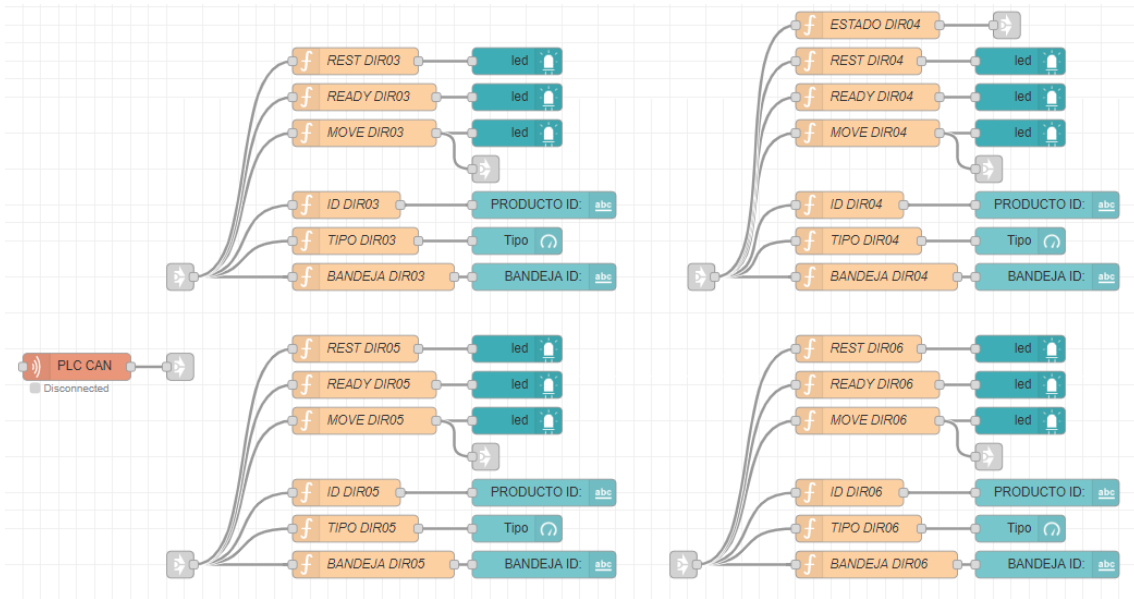


Figura 97. Flujo Retenedores [1].

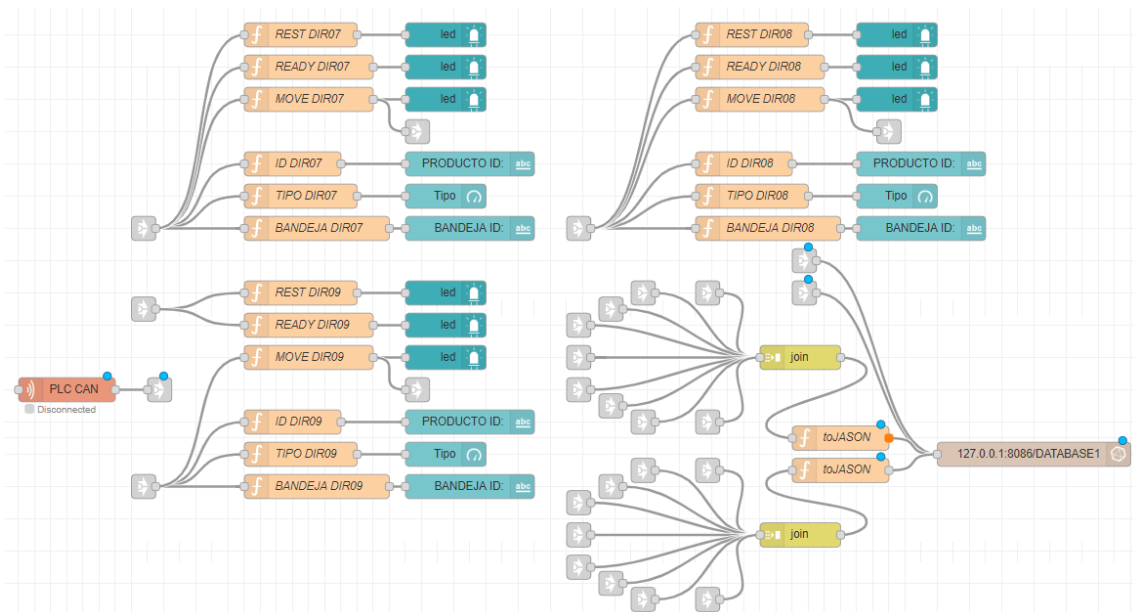


Figura 98. Flujo Retenedores [2].

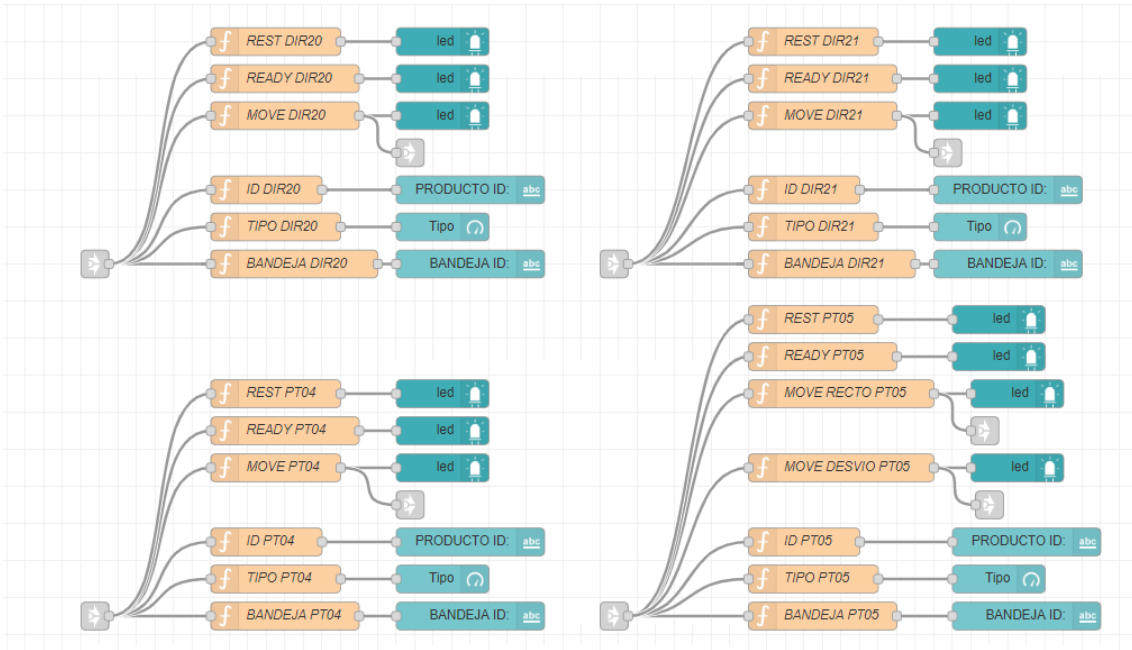


Figura 99. Flujo Retenedores [3].

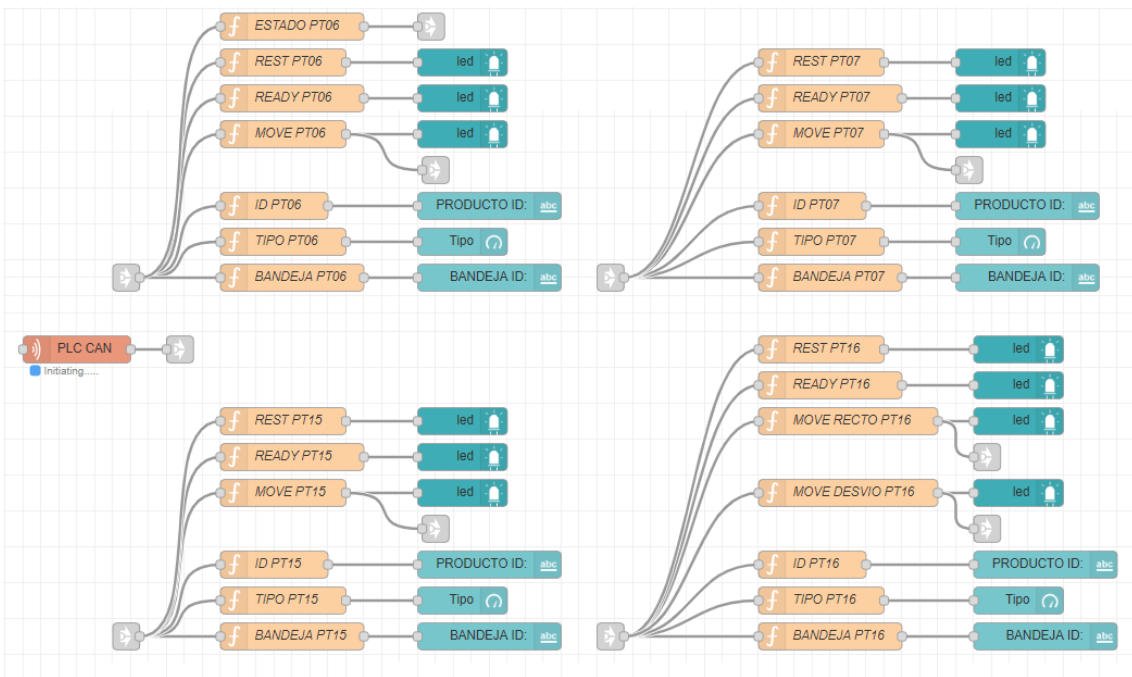


Figura 100. Flujo Retenedores [3].

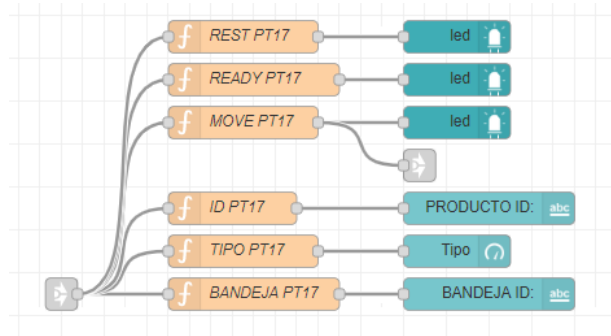


Figura 101. Flujo Retenedores [4].

Name REST DIR03

Function

```

1 let rest=0;
2 rest= msg.payload[0];
3
4 if(rest === 1 || rest === 257){
5   msg.payload=true;
6 }else msg.payload=false;
7
8 return msg;

```

Figura 102. Nodo función REST DIR03.

Name REST DIR04

Function

```

1 let rest=0;
2 rest= msg.payload[8];
3
4 if(rest === 1 || rest === 257){
5   msg.payload=true;
6 }else msg.payload=false;
7
8 return msg;

```

Figura 103. Nodo función REST DIR04.

Name REST DIR05

Function

```

1 let rest=0;
2 rest= msg.payload[17];
3
4 if(rest === 1 || rest === 257){
5   msg.payload=true;
6 }else msg.payload=false;
7
8 return msg;

```

Figura 104. Nodo función REST DIR05.

Name REST DIR06

Function

```
1 let rest=0;
2 rest= msg.payload[25];
3
4 if(rest === 1 || rest === 257){
5   msg.payload=true;
6 }else msg.payload=false;
7
8 return msg;
```

Figura 105. Nodo función REST DIR06.

Name REST DIR07

Function

```
1 let rest=0;
2 rest= msg.payload[33];
3
4 if(rest === 1 || rest === 257){
5   msg.payload=true;
6 }else msg.payload=false;
7
8 return msg;
```

Figura 106. Nodo función REST DIR07.

Name REST DIR08

Function

```
1 let rest=0;
2 rest= msg.payload[41];
3
4 if(rest === 1 || rest === 257){
5   msg.payload=true;
6 }else msg.payload=false;
7
8 return msg;
```

Figura 107. Nodo función REST DIR08.

Name REST DIR09

Function

```
1 let rest=0;
2 rest= msg.payload[49];
3
4 if(rest === 1 || rest === 257){
5   msg.payload=true;
6 }else msg.payload=false;
7
8 return msg;
```

Figura 108. Nodo función REST DIR09.

Name REST DIR20

Function

```
1 let rest=0;
2 rest= msg.payload[15];
3
4 if(rest === 1 || rest === 257){
5   msg.payload=true;
6 }else msg.payload=false;
7
8 return msg;
```

Figura 109. Nodo función REST DIR20.

Name REST DIR21

Function

```
1 let rest=0;
2 rest= msg.payload[23];
3
4 if(rest === 1 || rest === 257){
5   msg.payload=true;
6 }else msg.payload=false;
7
8 return msg;
```

Figura 110. Nodo función REST DIR21.

Name REST PT04

Function

```
1 let rest=0;
2 rest= msg.payload[31];
3
4 if(rest === 1 || rest === 257){
5   msg.payload=true;
6 }else msg.payload=false;
7
8 return msg;
```

Figura 111. Nodo función REST PT04.

Name REST PT05

Function

```
1 let rest=0;
2 rest= msg.payload[39];
3
4 if(rest === 1 || rest === 257){
5   msg.payload=true;
6 }else msg.payload=false;
7
8 return msg;
```

Figura 112. Nodo función REST PT05.

Name REST PT06

Function

```
1 let rest=0;
2 rest= msg.payload[0];
3
4 if(rest === 1 || rest === 257){
5   msg.payload=true;
6 }else msg.payload=false;
7
8 return msg;
```

Figura 113. Nodo función REST PT06.

Name REST PT07

Function

```
1 let rest=0;
2 rest= msg.payload[9];
3
4 if(rest === 1 || rest === 257){
5   msg.payload=true;
6 }else msg.payload=false;
7
8 return msg;
```

Figura 114. Nodo función REST PT07.

Name REST PT15

Function

```
1 let rest=0;
2 rest= msg.payload[17];
3
4 if(rest === 1 || rest === 257){
5   msg.payload=true;
6 }else msg.payload=false;
7
8 return msg;
```

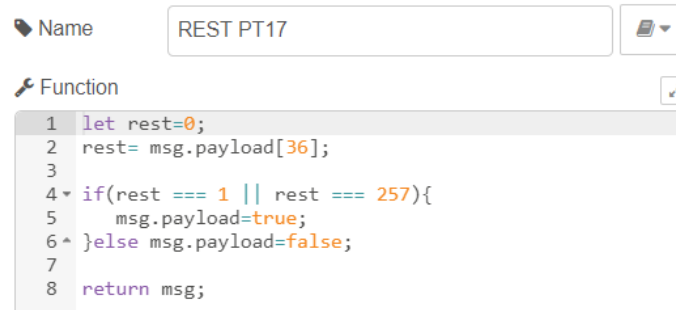
Figura 115. Nodo función REST PT15.

Name REST PT16

Function

```
1 let rest=0;
2 rest= msg.payload[25];
3
4 if(rest === 1 || rest === 257){
5   msg.payload=true;
6 }else msg.payload=false;
7
8 return msg;
```

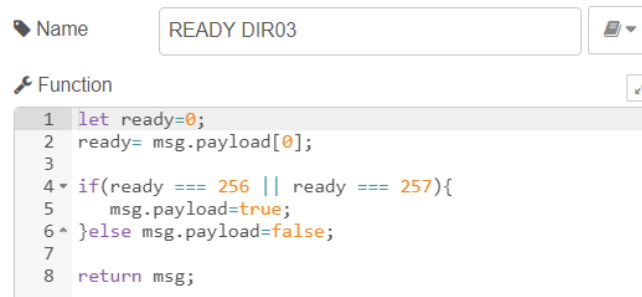
Figura 116. Nodo función REST PT16.



The screenshot shows a Node-RED function node named 'REST PT17'. The code in the function editor is as follows:

```
1 let rest=0;
2 rest= msg.payload[36];
3
4 if(rest === 1 || rest === 257){
5   msg.payload=true;
6 }else msg.payload=false;
7
8 return msg;
```

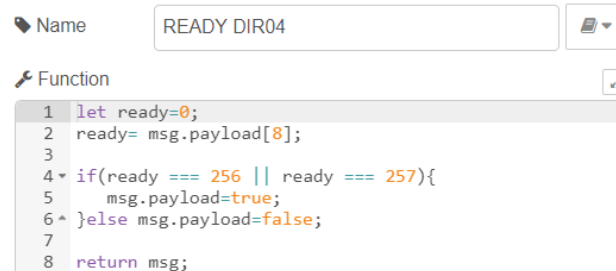
Figura 117. Nodo función REST PT17.



The screenshot shows a Node-RED function node named 'READY DIR03'. The code in the function editor is as follows:

```
1 let ready=0;
2 ready= msg.payload[0];
3
4 if(ready === 256 || ready === 257){
5   msg.payload=true;
6 }else msg.payload=false;
7
8 return msg;
```

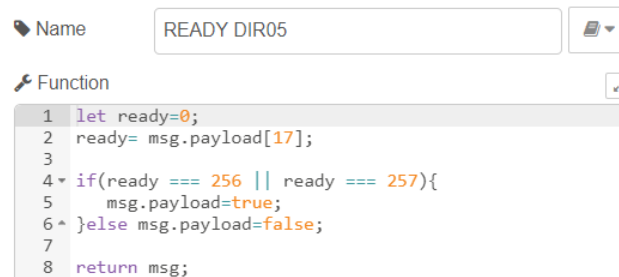
Figura 118. Nodo función READY DIR03.



The screenshot shows a Node-RED function node named 'READY DIR04'. The code in the function editor is as follows:

```
1 let ready=0;
2 ready= msg.payload[8];
3
4 if(ready === 256 || ready === 257){
5   msg.payload=true;
6 }else msg.payload=false;
7
8 return msg;
```

Figura 119. Nodo función READY DIR04.



The screenshot shows a Node-RED function node named 'READY DIR05'. The code in the function editor is as follows:

```
1 let ready=0;
2 ready= msg.payload[17];
3
4 if(ready === 256 || ready === 257){
5   msg.payload=true;
6 }else msg.payload=false;
7
8 return msg;
```

Figura 120. Nodo función READY DIR05.

Name: READY DIR06

Function

```

1 let ready=0;
2 ready= msg.payload[25];
3
4 if(ready === 256 || ready === 257){
5   msg.payload=true;
6 }else msg.payload=false;
7
8 return msg;

```

Figura 121. Nodo función READY DIR06.

Name: READY DIR07

Function

```

1 let ready=0;
2 ready= msg.payload[33];
3
4 if(ready === 256 || ready === 257){
5   msg.payload=true;
6 }else msg.payload=false;
7
8 return msg;

```

Figura 122. Nodo función READY DIR07.

Name: READY DIR08

Function

```

1 let ready=0;
2 ready= msg.payload[41];
3
4 if(ready === 256 || ready === 257){
5   msg.payload=true;
6 }else msg.payload=false;
7
8 return msg;

```

Figura 123. Nodo función READY DIR08.

Name: READY DIR09

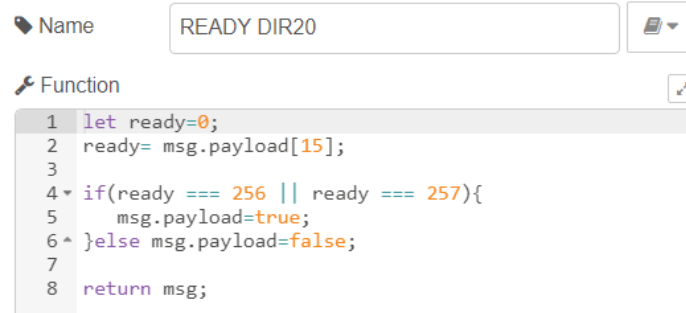
Function

```

1 let ready=0;
2 ready= msg.payload[49];
3
4 if(ready === 256 || ready === 257){
5   msg.payload=true;
6 }else msg.payload=false;
7
8 return msg;

```

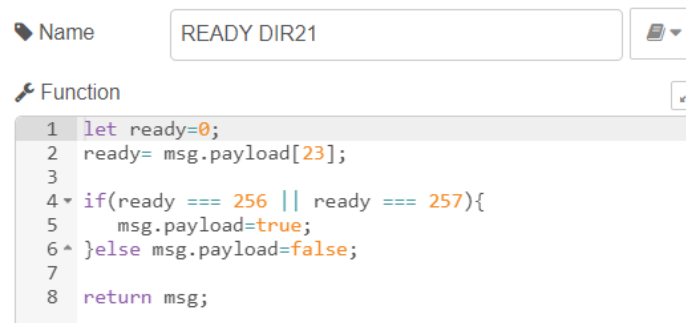
Figura 124. Nodo función READY DIR09.



The screenshot shows a Node-RED function node named 'READY DIR20'. The function code is as follows:

```
1 let ready=0;
2 ready= msg.payload[15];
3
4 if(ready === 256 || ready === 257){
5   msg.payload=true;
6 }else msg.payload=false;
7
8 return msg;
```

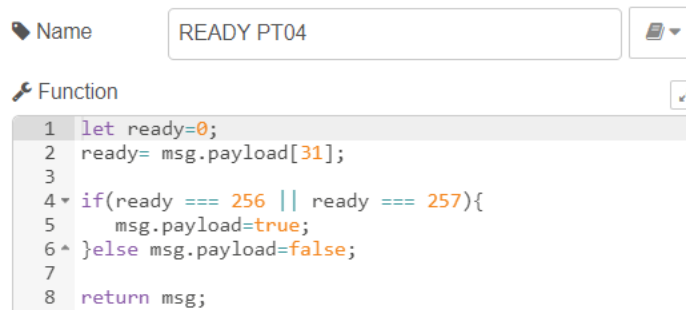
Figura 125. Nodo función READY DIR20.



The screenshot shows a Node-RED function node named 'READY DIR21'. The function code is as follows:

```
1 let ready=0;
2 ready= msg.payload[23];
3
4 if(ready === 256 || ready === 257){
5   msg.payload=true;
6 }else msg.payload=false;
7
8 return msg;
```

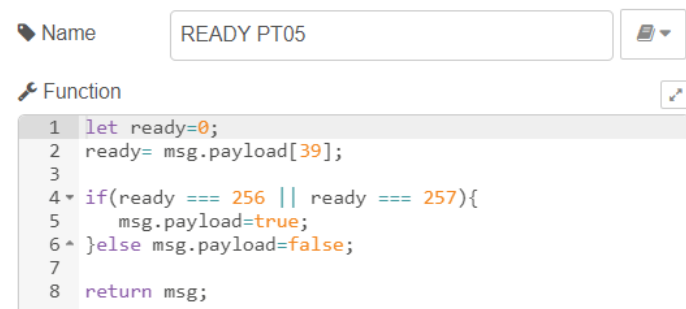
Figura 126. Nodo función READY DIR21.



The screenshot shows a Node-RED function node named 'READY PT04'. The function code is as follows:

```
1 let ready=0;
2 ready= msg.payload[31];
3
4 if(ready === 256 || ready === 257){
5   msg.payload=true;
6 }else msg.payload=false;
7
8 return msg;
```

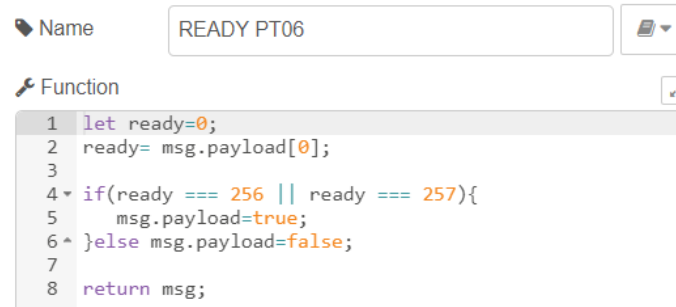
Figura 127. Nodo función READY PT04.



The screenshot shows a Node-RED function node named 'READY PT05'. The function code is as follows:

```
1 let ready=0;
2 ready= msg.payload[39];
3
4 if(ready === 256 || ready === 257){
5   msg.payload=true;
6 }else msg.payload=false;
7
8 return msg;
```

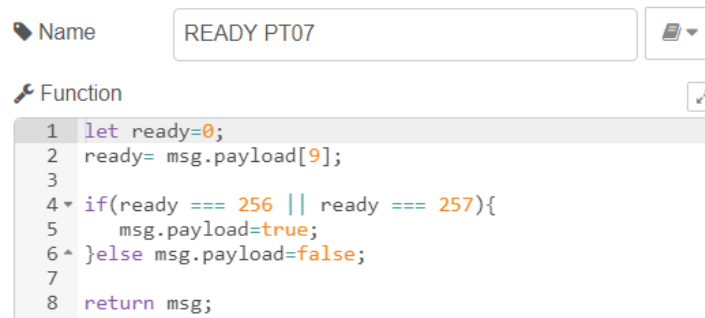
Figura 128. Nodo función READY PT05.



The screenshot shows a Node-RED function node named 'READY PT06'. The 'Name' field contains 'READY PT06'. The 'Function' field contains the following JavaScript code:

```
1 let ready=0;
2 ready= msg.payload[0];
3
4 if(ready === 256 || ready === 257){
5   msg.payload=true;
6 }else msg.payload=false;
7
8 return msg;
```

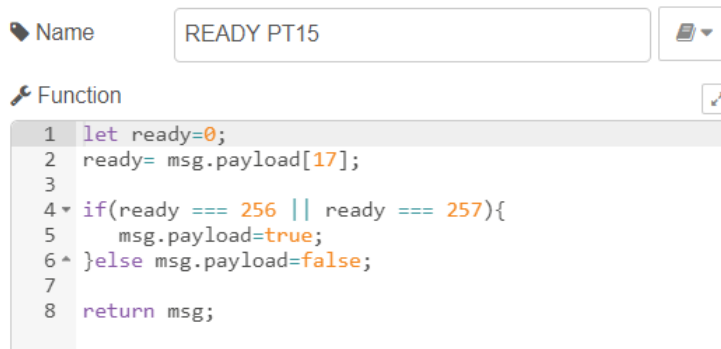
Figura 129. Nodo función READY PT06.



The screenshot shows a Node-RED function node named 'READY PT07'. The 'Name' field contains 'READY PT07'. The 'Function' field contains the following JavaScript code:

```
1 let ready=0;
2 ready= msg.payload[9];
3
4 if(ready === 256 || ready === 257){
5   msg.payload=true;
6 }else msg.payload=false;
7
8 return msg;
```

Figura 130. Nodo función READY PT07.



The screenshot shows a Node-RED function node named 'READY PT15'. The 'Name' field contains 'READY PT15'. The 'Function' field contains the following JavaScript code:

```
1 let ready=0;
2 ready= msg.payload[17];
3
4 if(ready === 256 || ready === 257){
5   msg.payload=true;
6 }else msg.payload=false;
7
8 return msg;
```

Figura 131. Nodo función READY PT15.

Name: READY PT16

Function:

```

1 let ready=0;
2 ready= msg.payload[25];
3
4 if(ready === 256 || ready === 257){
5   msg.payload=true;
6 }else msg.payload=false;
7
8 return msg;

```

Figura 132. Nodo función READY PT16.

Name: READY PT17

Function:

```

1 let ready=0;
2 ready= msg.payload[36];
3
4 if(ready === 256 || ready === 257){
5   msg.payload=true;
6 }else msg.payload=false;
7
8 return msg;

```

Figura 133. Nodo función READY PT17.

Name: MOVE DIR03

Function:

```

1 let ready=0;
2 ready= msg.payload[1];
3
4 if(ready === 1 || ready === 257){
5   msg.payload=1;
6 }else msg.payload=0;
7 msg.topic="DIR03";
8 return msg;

```

Figura 134. Nodo función MOVE DIR03.

Name: MOVE DIR04

Function:

```

1 let ready=0;
2 ready= msg.payload[9];
3
4 if(ready === 1 || ready === 257){
5   msg.payload=1;
6 }else msg.payload=0;
7 msg.topic="DIR04";
8 return msg;

```

Figura 135. Nodo función MOVE DIR04.

Name

Function

```

1 let ready=0;
2 ready= msg.payload[18];
3
4 if(ready === 1 || ready === 257){
5     msg.payload=1;
6 }else msg.payload=0;
7 msg.topic="DIR05";
8 return msg;

```

Figura 136. Nodo función MOVE DIR05.

Name

Function

```

1 let ready=0;
2 ready= msg.payload[26];
3
4 if(ready === 1 || ready === 257){
5     msg.payload=1;
6 }else msg.payload=0;
7 msg.topic="DIR06";
8 return msg;

```

Figura 137. Nodo función MOVE DIR06.

Name

Function

```

1 let ready=0;
2 ready= msg.payload[34];
3
4 if(ready === 1 || ready === 257){
5     msg.payload=1;
6 }else msg.payload=0;
7 msg.topic="DIR07";
8 return msg;

```

Figura 138. Nodo función MOVE DIR07.

Name

Function

```

1 let ready=0;
2 ready= msg.payload[42];
3
4 if(ready === 1 || ready === 257){
5     msg.payload=1;
6 }else msg.payload=0;
7 msg.topic="DIR08";
8 return msg;

```

Figura 139. Nodo función MOVE DIR08.

Name

Function

```
1 let ready=0;
2 ready= msg.payload[0];
3
4 if(ready === 1 || ready === 257){
5   msg.payload=1;
6 }else msg.payload=0;
7 msg.topic="DIR09";
8 return msg;
```

Figura 140. Nodo función MOVE DIR09.

Name

Function

```
1 let ready=0;
2 ready= msg.payload[16];
3
4 if(ready === 1 || ready === 257){
5   msg.payload=1;
6 }else msg.payload=0;
7 msg.topic="DIR20";
8 return msg;
```

Figura 141. Nodo función MOVE DIR20.

Name

Function

```
1 let ready=0;
2 ready= msg.payload[24];
3
4 if(ready === 1 || ready === 257){
5   msg.payload=1;
6 }else msg.payload=0;
7 msg.topic="DIR21";
8 return msg;
```

Figura 142. Nodo función MOVE DIR21.

Name

Function

```

1 let ready=0;
2 ready= msg.payload[32];
3
4 if(ready === 1 || ready === 257){
5   msg.payload=1;
6 }else msg.payload=0;
7 msg.topic="PT04";
8 return msg;

```

Figura 143. Nodo función MOVE PT04.

Name

Function

```

1 let move=0;
2 move= msg.payload[42];
3
4 if(move === 256 || move === 257){
5   msg.payload=1;
6 }else msg.payload=0;
7 msg.topic="PT05_RECTO";
8 return msg;

```

Figura 144. Nodo función MOVE RECTO PT05.

Name

Function

```

1 let move=0;
2 move = msg.payload[43];
3
4 if(move === 1 || move === 257){
5   msg.payload=1;
6 }else msg.payload=0;
7 msg.topic="PT05_DESVIO";
8 return msg;

```

Figura 145. Nodo función MOVE DESVIO PT05.

Name

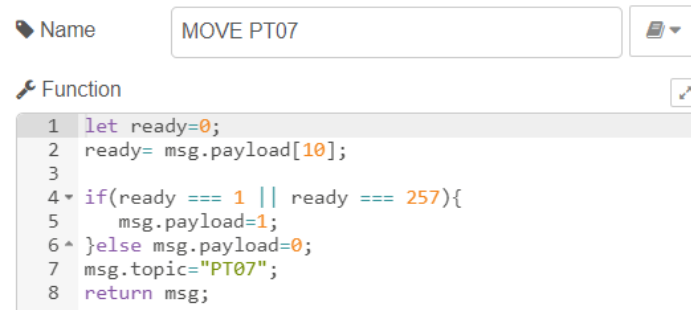
Function

```

1 let ready=0;
2 ready= msg.payload[1];
3
4 if(ready === 1 || ready === 257){
5   msg.payload=1;
6 }else msg.payload=0;
7 msg.topic="PT06";
8 return msg;

```

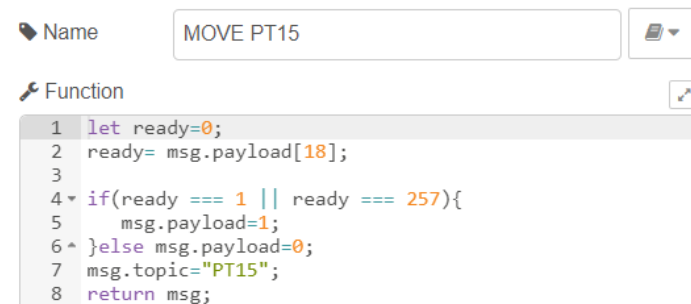
Figura 146. Nodo función MOVE PT06.



The screenshot shows the configuration for a Node-RED function node named "MOVE PT07". The function code is as follows:

```
1 let ready=0;
2 ready= msg.payload[10];
3
4 if(ready === 1 || ready === 257){
5   msg.payload=1;
6 }else msg.payload=0;
7 msg.topic="PT07";
8 return msg;
```

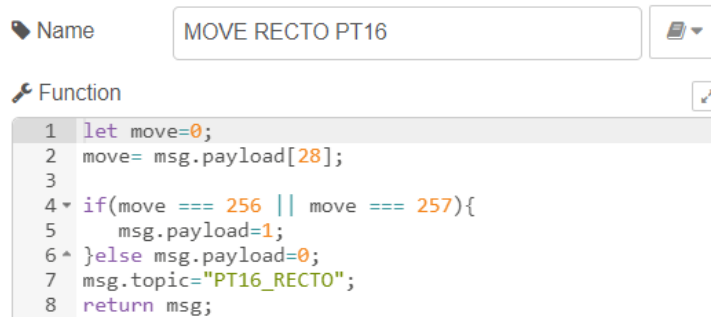
Figura 147. Nodo función MOVE PT07.



The screenshot shows the configuration for a Node-RED function node named "MOVE PT15". The function code is as follows:

```
1 let ready=0;
2 ready= msg.payload[18];
3
4 if(ready === 1 || ready === 257){
5   msg.payload=1;
6 }else msg.payload=0;
7 msg.topic="PT15";
8 return msg;
```

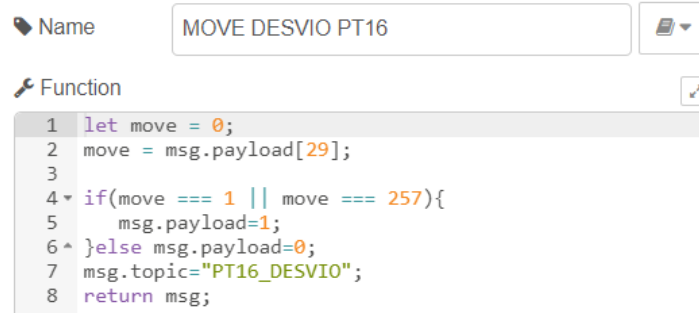
Figura 148. Nodo función MOVE PT15.



The screenshot shows the configuration for a Node-RED function node named "MOVE RECTO PT16". The function code is as follows:

```
1 let move=0;
2 move= msg.payload[28];
3
4 if(move === 256 || move === 257){
5   msg.payload=1;
6 }else msg.payload=0;
7 msg.topic="PT16_RECTO";
8 return msg;
```

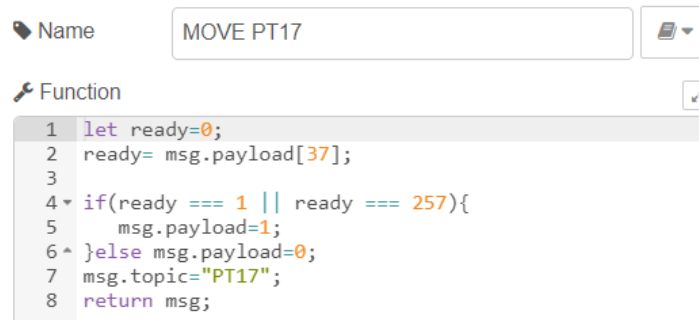
Figura 149. Nodo función MOVE RECTO PT16.



The screenshot shows a Node-RED function node configuration. The 'Name' field contains 'MOVE DESVIO PT16'. The 'Function' field contains the following JavaScript code:

```
1 let move = 0;
2 move = msg.payload[29];
3
4 if(move === 1 || move === 257){
5     msg.payload=1;
6 }else msg.payload=0;
7 msg.topic="PT16_DESVIO";
8 return msg;
```

Figura 150. Nodo función MOVE DESVIO PT16.



The screenshot shows a Node-RED function node configuration. The 'Name' field contains 'MOVE PT17'. The 'Function' field contains the following JavaScript code:

```
1 let ready=0;
2 ready= msg.payload[37];
3
4 if(ready === 1 || ready === 257){
5     msg.payload=1;
6 }else msg.payload=0;
7 msg.topic="PT17";
8 return msg;
```

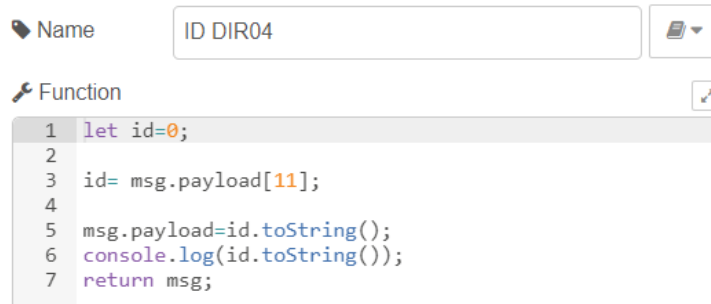
Figura 151. Nodo función MOVE PT17.



The screenshot shows a Node-RED function node configuration. The 'Name' field contains 'ID DIR03'. The 'Function' field contains the following JavaScript code:

```
1 let id=0;
2 id= msg.payload[2];
3 msg.payload=id.toString();
4 return msg;
```

Figura 152. Nodo función ID DIR03.



The screenshot shows a Node-RED function node configuration. The 'Name' field contains 'ID DIR04'. The 'Function' field contains the following JavaScript code:

```
1 let id=0;
2
3 id= msg.payload[11];
4
5 msg.payload=id.toString();
6 console.log(id.toString());
7 return msg;
```

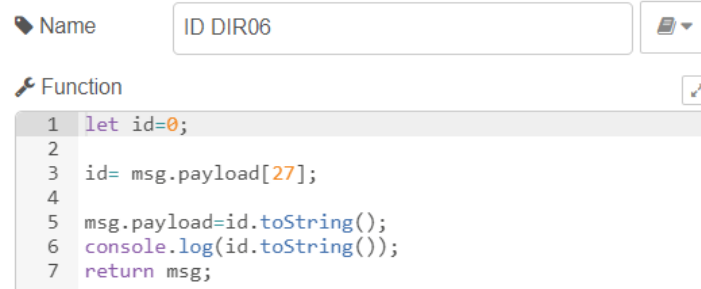
Figura 153. Nodo función ID DIR04.



The screenshot shows a Node-RED function node configuration. The 'Name' field contains 'ID DIR05'. The 'Function' field contains the following JavaScript code:

```
1 let id=0;
2
3 id= msg.payload[19];
4
5 msg.payload=id.toString();
6
7 return msg;
```

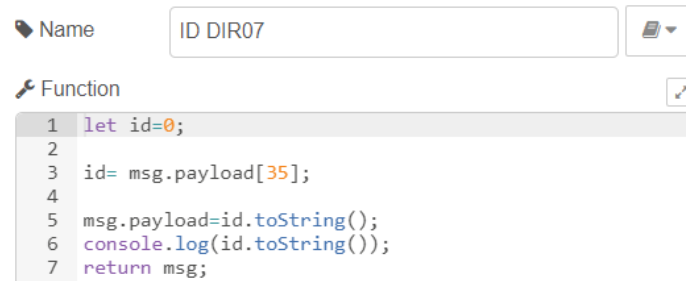
Figura 154. Nodo función ID DIR05.



The screenshot shows a Node-RED function node configuration. The 'Name' field contains 'ID DIR06'. The 'Function' field contains the following JavaScript code:

```
1 let id=0;
2
3 id= msg.payload[27];
4
5 msg.payload=id.toString();
6 console.log(id.toString());
7 return msg;
```

Figura 155. Nodo función ID DIR06.



The screenshot shows a Node-RED function node configuration. The 'Name' field contains 'ID DIR07'. The 'Function' field contains the following JavaScript code:

```
1 let id=0;
2
3 id= msg.payload[35];
4
5 msg.payload=id.toString();
6 console.log(id.toString());
7 return msg;
```

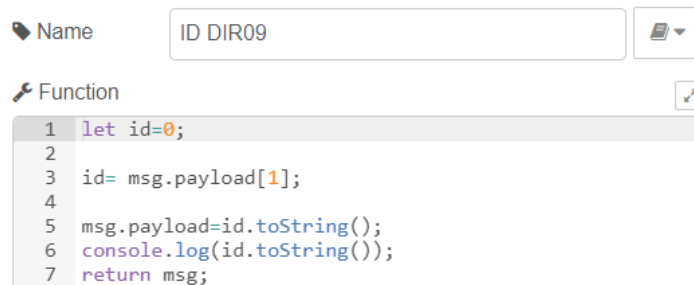
Figura 156. Nodo función ID DIR07.



The screenshot shows a Node-RED function node configuration. The 'Name' field contains 'ID DIR08'. The 'Function' field contains the following JavaScript code:

```
1 let id=0;
2
3 id= msg.payload[43];
4
5 msg.payload=id.toString();
6 console.log(id.toString());
7 return msg;
```

Figura 157. Nodo función ID DIR08.



The screenshot shows a Node-RED function node configuration. The 'Name' field contains 'ID DIR09'. The 'Function' field contains the following JavaScript code:

```
1 let id=0;
2
3 id= msg.payload[1];
4
5 msg.payload=id.toString();
6 console.log(id.toString());
7 return msg;
```

Figura 158. Nodo función ID DIR09.



The screenshot shows a Node-RED function node configuration. The 'Name' field contains 'ID DIR20'. The 'Function' field contains the following JavaScript code:

```
1 let id=0;
2
3 id= msg.payload[17];
4
5 msg.payload=id.toString();
6 console.log(id.toString());
7 return msg;
```

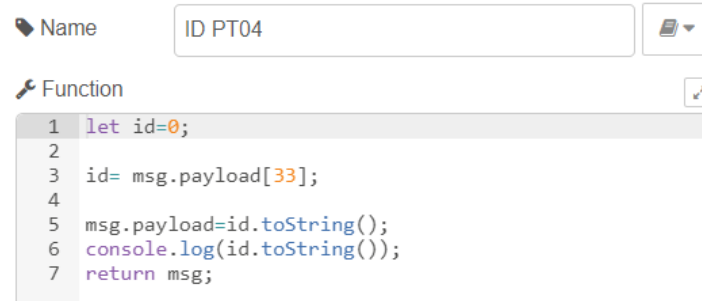
Figura 159. Nodo función ID DIR20.



The screenshot shows a Node-RED function node configuration. The 'Name' field contains 'ID DIR21'. The 'Function' field contains the following JavaScript code:

```
1 let id=0;
2
3 id= msg.payload[25];
4
5 msg.payload=id.toString();
6 console.log(id.toString());
7 return msg;
```

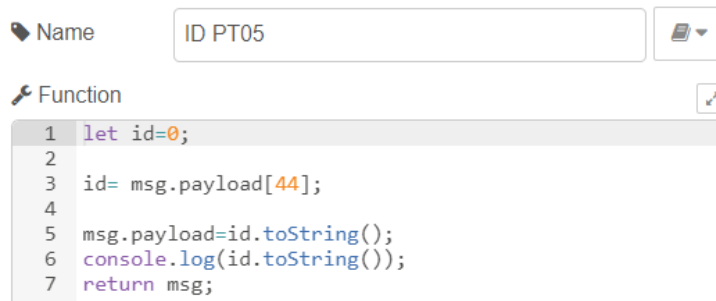
Figura 160. Nodo función ID DIR21.



The screenshot shows a Node-RED function node configuration. The 'Name' field contains 'ID PT04'. The 'Function' field contains the following JavaScript code:

```
1 let id=0;
2
3 id= msg.payload[33];
4
5 msg.payload=id.toString();
6 console.log(id.toString());
7 return msg;
```

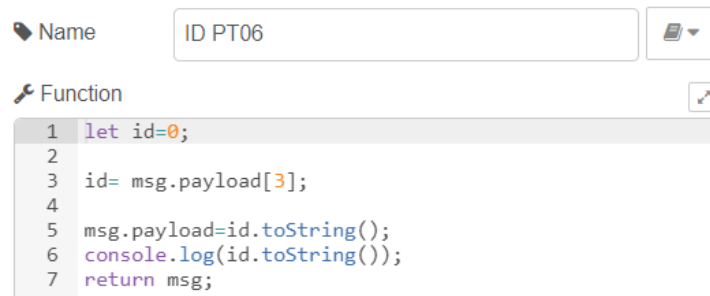
Figura 161. Nodo función ID PT04.



The screenshot shows a Node-RED function node configuration. The 'Name' field contains 'ID PT05'. The 'Function' field contains the following JavaScript code:

```
1 let id=0;
2
3 id= msg.payload[44];
4
5 msg.payload=id.toString();
6 console.log(id.toString());
7 return msg;
```

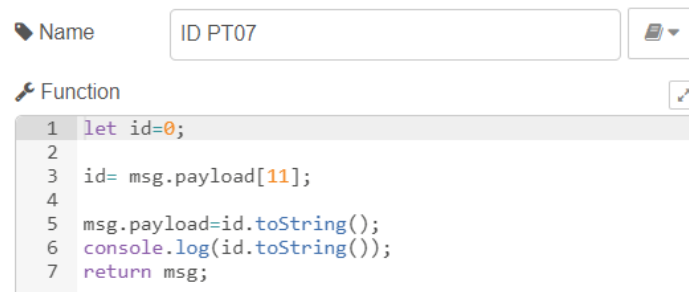
Figura 162. Nodo función ID PT05.



The screenshot shows a Node-RED function node configuration. The 'Name' field contains 'ID PT06'. The 'Function' field contains the following JavaScript code:

```
1 let id=0;
2
3 id= msg.payload[3];
4
5 msg.payload=id.toString();
6 console.log(id.toString());
7 return msg;
```

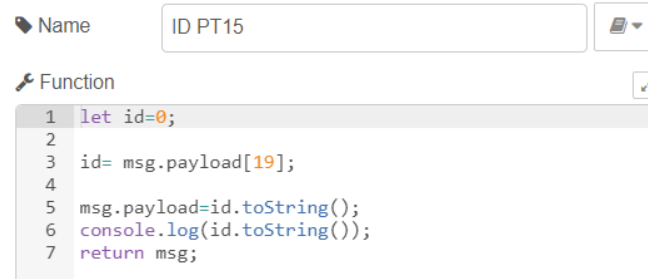
Figura 163. Nodo función ID PT06.



The screenshot shows a Node-RED function node configuration. The 'Name' field contains 'ID PT07'. The 'Function' field contains the following JavaScript code:

```
1 let id=0;
2
3 id= msg.payload[11];
4
5 msg.payload=id.toString();
6 console.log(id.toString());
7 return msg;
```

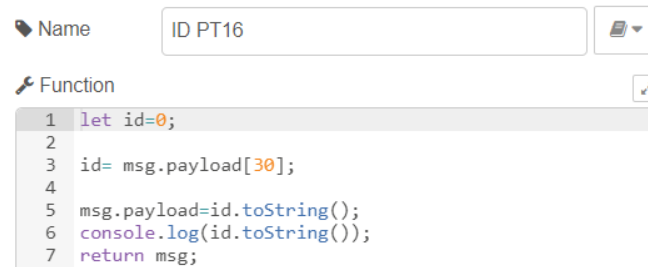
Figura 164. Nodo función ID PT07.



The screenshot shows a Node-RED function node configuration. The 'Name' field is 'ID PT15'. The 'Function' field contains the following JavaScript code:

```
1 let id=0;
2
3 id= msg.payload[19];
4
5 msg.payload=id.toString();
6 console.log(id.toString());
7 return msg;
```

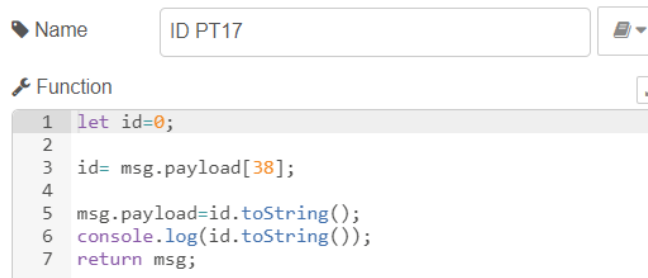
Figura 165. Nodo función ID PT15.



The screenshot shows a Node-RED function node configuration. The 'Name' field is 'ID PT16'. The 'Function' field contains the following JavaScript code:

```
1 let id=0;
2
3 id= msg.payload[30];
4
5 msg.payload=id.toString();
6 console.log(id.toString());
7 return msg;
```

Figura 166. Nodo función ID PT16.



The screenshot shows a Node-RED function node configuration. The 'Name' field is 'ID PT17'. The 'Function' field contains the following JavaScript code:

```
1 let id=0;
2
3 id= msg.payload[38];
4
5 msg.payload=id.toString();
6 console.log(id.toString());
7 return msg;
```

Figura 167. Nodo función ID PT17.



The screenshot shows a Node-RED function node configuration. The 'Name' field is 'TIPO DIR03'. The 'Function' field contains the following JavaScript code:

```
1 let tipo=0;
2 tipo= msg.payload[3];
3 msg.payload=tipo;
4 return msg;
```

Figura 168. Nodo función TIPO DIR03.



Figura 169. Nodo función TIPO DIR04.



Figura 170. Nodo función TIPO DIR05.



Figura 171. Nodo función TIPO DIR06.



Figura 172. Nodo función TIPO DIR07.



The screenshot shows a Node-RED function node named 'TIPO DIR08'. The function code is as follows:

```
1 let tipo=0;
2 tipo= msg.payload[44];
3 msg.payload=tipo;
4 return msg;
```

Figura 173. Nodo función TIPO DIR08.



The screenshot shows a Node-RED function node named 'TIPO DIR09'. The function code is as follows:

```
1 let tipo=0;
2 tipo= msg.payload[2];
3 msg.payload=tipo;
4 return msg;
```

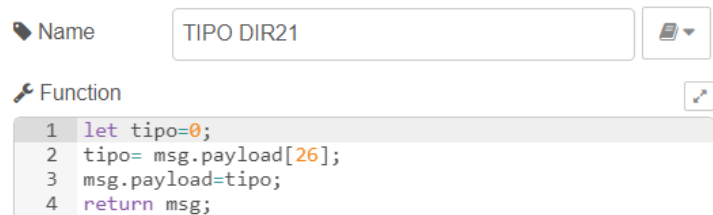
Figura 174. Nodo función TIPO DIR09.



The screenshot shows a Node-RED function node named 'TIPO DIR20'. The function code is as follows:

```
1 let tipo=0;
2 tipo= msg.payload[18];
3 msg.payload=tipo;
4 return msg;
```

Figura 175. Nodo función TIPO DIR20.



The screenshot shows a Node-RED function node named 'TIPO DIR21'. The function code is as follows:

```
1 let tipo=0;
2 tipo= msg.payload[26];
3 msg.payload=tipo;
4 return msg;
```

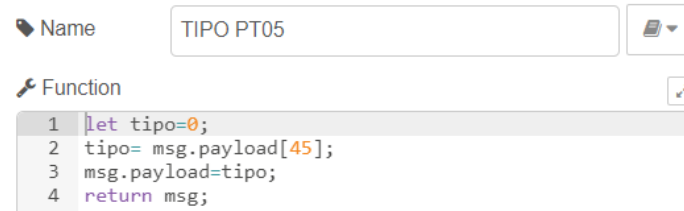
Figura 176. Nodo función TIPO DIR21.



The screenshot shows a Node-RED function node named 'TIPO PT04'. The function code is as follows:

```
1 let tipo=0;
2 tipo= msg.payload[34];
3 msg.payload=tipo;
4 return msg;
```

Figura 177. Nodo función TIPO PT04.



The screenshot shows a Node-RED function node configuration. The 'Name' field contains 'TIPO PT05'. The 'Function' field contains the following JavaScript code:

```
1 let tipo=0;
2 tipo= msg.payload[45];
3 msg.payload=tipo;
4 return msg;
```

Figura 178. Nodo función TIPO PT05.



The screenshot shows a Node-RED function node configuration. The 'Name' field contains 'TIPO PT06'. The 'Function' field contains the following JavaScript code:

```
1 let tipo=0;
2 tipo= msg.payload[4];
3 msg.payload=tipo;
4 return msg;
```

Figura 179. Nodo función TIPO PT06.



The screenshot shows a Node-RED function node configuration. The 'Name' field contains 'TIPO PT07'. The 'Function' field contains the following JavaScript code:

```
1 let tipo=0;
2 tipo= msg.payload[12];
3 msg.payload=tipo;
4 return msg;
```

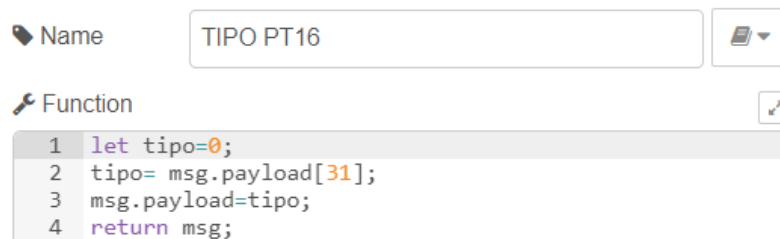
Figura 180. Nodo función TIPO PT07.



The screenshot shows a Node-RED function node configuration. The 'Name' field contains 'TIPO PT15'. The 'Function' field contains the following JavaScript code:

```
1 let tipo=0;
2 tipo= msg.payload[20];
3 msg.payload=tipo;
4 return msg;
```

Figura 181. Nodo función TIPO PT15.



The screenshot shows a Node-RED function node configuration. The 'Name' field contains 'TIPO PT16'. The 'Function' field contains the following JavaScript code:

```
1 let tipo=0;
2 tipo= msg.payload[31];
3 msg.payload=tipo;
4 return msg;
```

Figura 182. Nodo función TIPO PT16.



Figura 183. Nodo función TIPO PT17.



Figura 184. Nodo función BANDEJA DIR03.

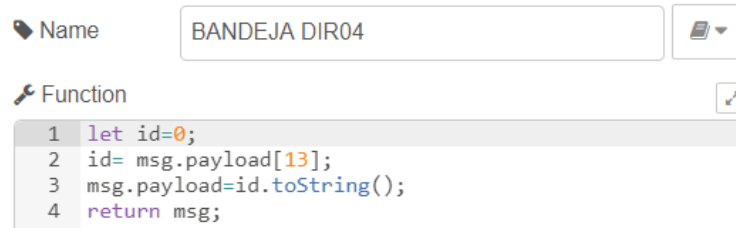


Figura 185. Nodo función BANDEJA DIR04.

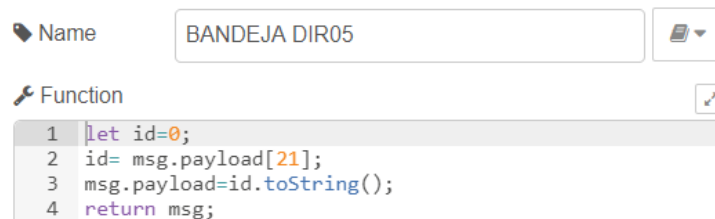


Figura 186. Nodo función BANDEJA DIR05.

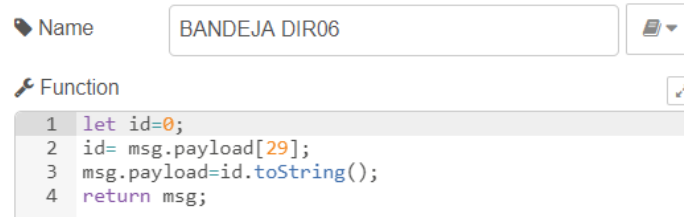


Figura 187. Nodo función BANDEJA DIR06.

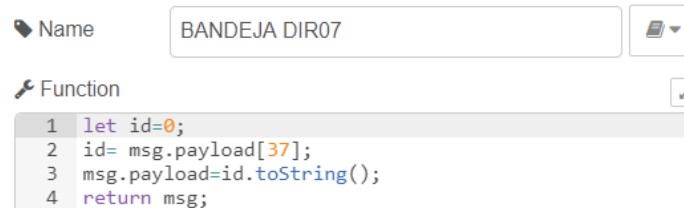


Figura 188. Nodo función BANDEJA DIR07.

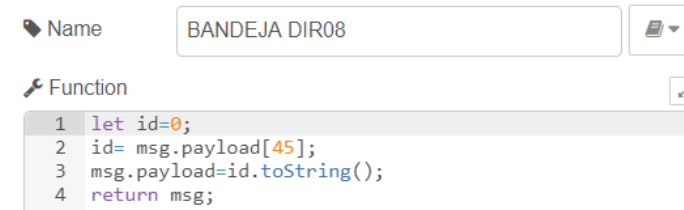


Figura 189. Nodo función BANDEJA DIR08.

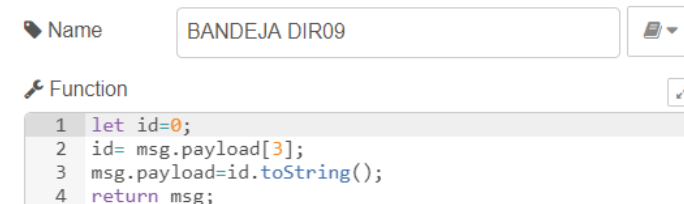


Figura 190. Nodo función BANDEJA DIR09.

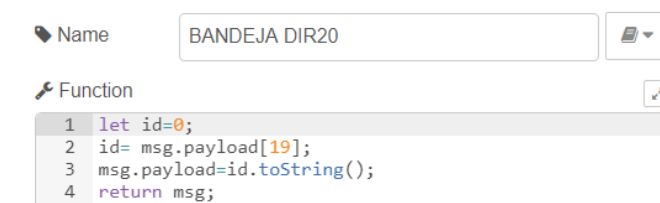
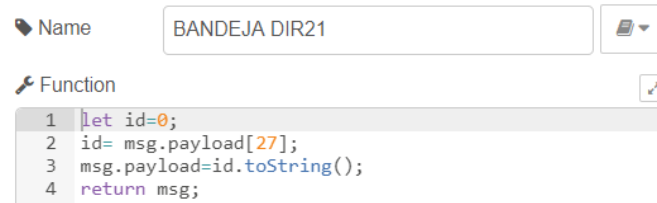


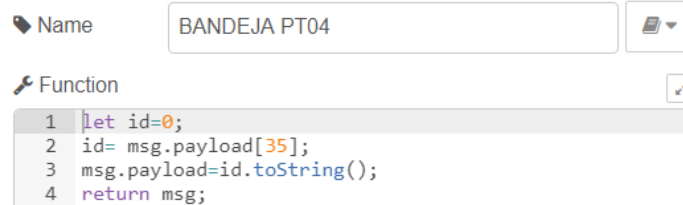
Figura 191. Nodo función BANDEJA DIR20.



The screenshot shows a Node-RED function node configuration. The 'Name' field contains 'BANDEJA DIR21'. The 'Function' field contains the following JavaScript code:

```
1 let id=0;
2 id= msg.payload[27];
3 msg.payload=id.toString();
4 return msg;
```

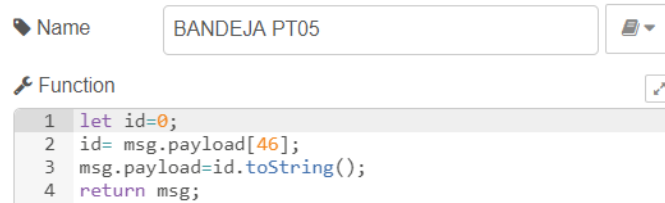
Figura 192. Nodo función BANDEJA DIR21.



The screenshot shows a Node-RED function node configuration. The 'Name' field contains 'BANDEJA PT04'. The 'Function' field contains the following JavaScript code:

```
1 let id=0;
2 id= msg.payload[35];
3 msg.payload=id.toString();
4 return msg;
```

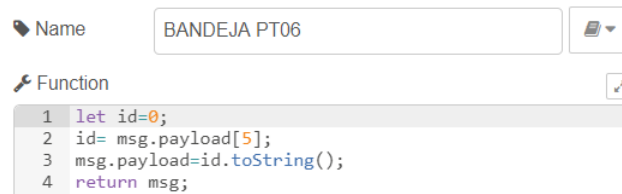
Figura 193. Nodo función BANDEJA PT04.



The screenshot shows a Node-RED function node configuration. The 'Name' field contains 'BANDEJA PT05'. The 'Function' field contains the following JavaScript code:

```
1 let id=0;
2 id= msg.payload[46];
3 msg.payload=id.toString();
4 return msg;
```

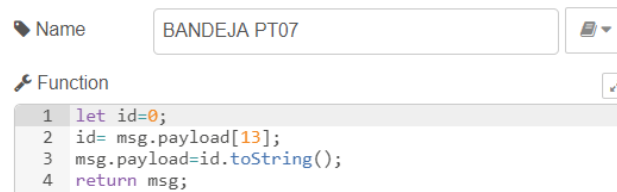
Figura 194. Nodo función BANDEJA PT05.



The screenshot shows a Node-RED function node configuration. The 'Name' field contains 'BANDEJA PT06'. The 'Function' field contains the following JavaScript code:

```
1 let id=0;
2 id= msg.payload[5];
3 msg.payload=id.toString();
4 return msg;
```

Figura 195. Nodo función BANDEJA PT06.



The screenshot shows a Node-RED function node configuration. The 'Name' field contains 'BANDEJA PT07'. The 'Function' field contains the following JavaScript code:

```
1 let id=0;
2 id= msg.payload[13];
3 msg.payload=id.toString();
4 return msg;
```

Figura 196. Nodo función BANDEJA PT07.

Name BANDEJA PT15

Function

```

1 let id=0;
2 id= msg.payload[21];
3 msg.payload=id.toString();
4 return msg;

```

Figura 197. Nodo función BANDEJA PT15.

Name BANDEJA PT16

Function

```

1 let id=0;
2 id= msg.payload[32];
3 msg.payload=id.toString();
4 return msg;

```

Figura 198. Nodo función BANDEJA PT16.

Name BANDEJA PT17

Function

```

1 let id=0;
2 id= msg.payload[40];
3 msg.payload=id.toString();
4 return msg;

```

Figura 199. Nodo función BANDEJA PT17.

Name ESTADO DIR04

Function

```

1 let time=msg.payload[10];
2
3 if(time){
4   msg.payload=[
5     {
6       measurement: "TiempoDIR04",
7       fields: {
8         TIEMPO:time
9       }
10    }
11  ];
12 return msg;
13 }

```

Figura 200. Nodo función ESTADO DIR04.

Name

Function

```
1 let time=msg.payload[2];
2
3 if(time){
4   msg.payload=[
5     {
6       measurement: "TiempoPT06",
7       fields: {
8         TIEMPO:time
9       }
10    }
11  ];
12 return msg;
13 }
```

Figura 201. Nodo función ESTADO PT06.

```
1 let emergency= global.get("EMERGENCY");
2
3 let dir03=msg.payload["DIR03"];
4 let dir04=msg.payload["DIR04"];
5 let dir05=msg.payload["DIR05"];
6 let dir06=msg.payload["DIR06"];
7 let dir07=msg.payload["DIR07"];
8 let dir08=msg.payload["DIR08"];
9 let dir09=msg.payload["DIR09"];
10 let dir20=msg.payload["DIR20"];
11 let dir21=msg.payload["DIR21"];
12
13 let automatico= global.get("CICLO_AUTOMATICO");
14 let lote= global.get("CICLO_LOTE");
15
16 if(emergency & (automatico || lote)){
17
18 msg.payload=[
19 {
20     measurement: "MoveRetenedores",
21     fields: {
22         MOVE_DIR03: dir03,
23         MOVE_DIR04: dir04,
24         MOVE_DIR05: dir05,
25         MOVE_DIR06: dir06,
26         MOVE_DIR07: dir07,
27         MOVE_DIR08: dir08,
28         MOVE_DIR09: dir09,
29         MOVE_DIR20: dir20,
30         MOVE_DIR21: dir21
31     }
32 } ];
33
34 return msg;
35 }
```

Figura 202. Nodo función toJSON (estado movimiento retenedores.)

```
1 let emergency= global.get("EMERGENCY");
2
3 let pt04=msg.payload["PT04"];
4 let pt05_recto=msg.payload["PT05_RECTO"];
5 let pt05_desvio=msg.payload["PT05_DESVIO"];
6 let pt06=msg.payload["PT06"];
7 let pt07=msg.payload["PT07"];
8 let pt15=msg.payload["PT15"];
9 let pt16_recto=msg.payload["PT16_RECTO"];
10 let pt16_desvio=msg.payload["PT16_DESVIO"];
11 let pt17=msg.payload["PT17"];
12
13 let automatico= global.get("CICLO_AUTOMATICO");
14 let lote= global.get("CICLO_LOTE");
15
16 if(emergency & (automatico || lote)){
17
18 msg.payload=[
19 {
20     measurement: "MovePlataformas",
21     fields: {
22         MOVE_PT04: pt04,
23         MOVE_PT05_RECTO: pt05_recto,
24         MOVE_PT05_DESVIO: pt05_desvio,
25         MOVE_PT06: pt06,
26         MOVE_PT07: pt07,
27         MOVE_PT15: pt15,
28         MOVE_PT16_RECTO: pt16_recto,
29         MOVE_PT16_DESVIO: pt16_desvio,
30         MOVE_PT17: pt17
31     }
32 }
33 ];
34
35 return msg;
36 }
37
```

Figura 203. Nodo función toJSON (estado movimiento plataformas).